No. 18-956

IN THE

Supreme Court of the United States

GOOGLE LLC,

Petitioner,

v.

ORACLE AMERICA, INC.,

Respondent.

On Writ of Certiorari to the
U.S. Court of Appeals for the Federal Circuit

JOINT APPENDIX VOLUME 1
PAGES 1-341

Thomas C. Goldstein
GOLDSTEIN & RUSSELL, P.C.
7475 Wisconsin Ave.
Suite 850
Bethesda, MD 20814
(202) 362-0636
tg@goldsteinrussell.com
Counsel of Record for Petitioner

E. Joshua Rosenkranz
ORRICK, HERRINGTON & SUTCLIFFE LLP
51 West 52nd Street
New York, NY 10019
(212) 506-5000
jrosenkranz@orrick.com
Counsel of Record for Respondent

PETITION FOR A WRIT OF CERTIORARI FILED JAN. 24, 2019
CERTIORARI GRANTED NOV. 15, 2019

TABLE OF CONTENTS

VOLUME 1

VOLUME 2

The following decisions have been omitted in printing the joint appendix because they appear in the appendix to the petition for certiorari, beginning on the following pages:

UNITED STATES COURT OF APPEALS
FOR THE FEDERAL CIRCUIT

———

No. 13-1021

———

ORACLE AMERICA, INC.,

Plaintiff-Appellant,

v.

GOOGLE INC.,

Defendant-Cross-Appellant.

———

DOCKET ENTRIES

Date	**#**	**Docket Text**
10/19/2012	1	Appeal docketed. Received: 10/10/2012. [30710] Entry of Appearance due 11/02/2012. Certificate of Interest is due on 11/02/2012. Docketing Statement due 11/02/2012. Appellant/Petitioner's brief due 12/18/2012. [SJ] [Entered: 10/19/2012 12:54 PM]
10/19/2012	2	Note to file: 13-1022 (cross-appeal started 10/19/2012) is consolidated with 13-1021. [30735] [SJ] [Entered: 10/19/2012 01:52 PM]
10/19/2012	3	Amended Notice of Appeal for Google Inc. Service: 10/05/2012 by US mail. [30737] [SJ] [Entered: 10/19/2012 01:59 PM]

* * *

Date	#	Docket Text
12/04/2013	152	Submitted after ORAL ARGUMENT by Mr. E. Joshua Rosenkranz for Oracle America, Inc. and Robert A. Van Nest for Google Inc.. Panel: Judge: O'Malley , Judge: Plager , Judge: Taranto. [120763] [LB] [Entered: 12/04/2013 10:50 AM]
05/09/2014	153	OPINION and JUDGMENT filed. The judgment or decision is: Affirmed-in-part,Reversed-in-part and Remnded. (Precedential Opinion). (For the Court: O'Malley,Circuit Judge; Plager,Circuit Judge and Taranto,Circuit Judge). [153414] [13-1021, 13-1022] [LP] [Entered: 05/09/2014 11:32 AM]
		* * *
06/16/2014	156	Mandate issued to the United States District Court for the Northern District of California. Service: 06/16/2014 by clerk. [161547] [13-1021, 13-1022] [LAJ] [Entered: 06/16/2014 09:28 AM]
		* * *

UNITED STATES COURT OF APPEALS
FOR THE FEDERAL CIRCUIT

No. 17-1118

ORACLE AMERICA, INC.,

Plaintiff-Appellant,

v.

GOOGLE LLC,

Defendant-Cross-Appellant.

DOCKET ENTRIES

Date	#	Docket Text
10/28/2016	1	Appeal docketed. Received: 10/27/2016. [378284] Entry of Appearance due 11/14/2016. Certificate of Interest due 11/14/2016. Docketing Statement due 11/14/2016. Appellant's brief due 12/27/2016. [MJL] [Entered: 10/28/2016 10:43 AM]
		* * *
11/14/2016	25	Note to file: The following cases are consolidated: 17-1118 (Lead) with 17-1202 (Cross-Appeal). FURTHER ENTRIES WILL BE ADDED TO THE LEAD APPEAL ONLY. [382361] [17-1118, 17-1202] [MJL] [Entered: 11/14/2016 10:12 AM]
		* * *

Date	#	Docket Text
12/07/2017	239	Submitted after ORAL ARGUMENT by E. Joshua Rosenkranz for Oracle America, Inc. and Daryl Joseffer for Google Inc. Panel: Judge: O'Malley , Judge: Plager , Judge: Taranto. [481029] [JAB] [Entered: 12/07/2017 03:33 PM]
		* * *
03/27/2018	243	OPINION and JUDGMENT filed. The judgment or decision is: Reversed and Remanded; Cross-Appeal Dismissed. (Precedential Opinion). (For the Court: O'Malley, Circuit Judge; Plager, Circuit Judge and Taranto, Circuit Judge). [508126] [17-1118, 17-1202] [SJ] [Entered: 03/27/2018 09:38 AM]
		* * *
08/28/2018	304	ORDER filed denying [249] petition for en banc rehearing filed by Google LLC. By: En Banc (Per Curiam). Service as of this date by the Clerk of Court. [545797] [SJ] [Entered: 08/28/2018 10:42 AM]
09/04/2018	305	Mandate issued to the United States District Court for the Northern District of California. Service as of this date by the Clerk of Court. [546915] [17-1118, 17-1202] [SJ] [Entered: 09/04/2018 01:09 PM]
		* * *

UNITED STATES DISTRICT COURT
CALIFORNIA NORTHERN DISTRICT
(SAN FRANCISCO)

No. 3:10-cv-03561-WHA

ORACLE AMERICA, INC.,

v.

GOOGLE INC.,

DOCKET ENTRIES

Date	#	Docket Text
8/12/2010	1	COMPLAINT (with jury demand) For Patent and Copyright Infringement against Google Inc. (Filing fee $350, receipt number 54611007901). Filed by Oracle America, Inc. (Attachments: # 1 Civil Cover Sheet)(vlk, COURT STAFF) (Filed on 8/12/2010) Modified on 8/18/2010 (cjl, COURT STAFF). (Entered: 08/17/2010)
		* * *
10/04/2010	32	GOOGLE INC.'S ANSWER to Complaint with Jury Demand, COUNTERCLAIM against Oracle America, Inc. byGoogle Inc.. (Zimmer, Donald) (Filed on 10/4/2010) (Entered: 10/04/2010)
		* * *

Date	#	Docket Text
10/27/2010	36	AMENDED COMPLAINT for patent and copyright infringement against Google Inc.. Filed byOracle America, Inc.. (Attachments: # 1 Exhibit A, # 2 Exhibit B, # 3 Exhibit C, # 4 Exhibit D, # 5 Exhibit E, # 6 Exhibit F, # 7 Exhibit G, # 8 Exhibit H, # 9 Exhibit I, # 10 Exhibit J)(Peters, Marc) (Filed on 10/27/2010) (Entered: 10/27/2010)
		* * *
10/28/2010	41	ANSWER TO COUNTERCLAIM 32 Answer to Complaint, Counterclaim Oracle America, Inc.'s Reply to Defendant Google Inc.'s Answer to Complaint for Patent and Copyright Infringement and Counterclaims byOracle America, Inc.. (Ballinger, Richard) (Filed on 10/28/2010) (Entered: 10/28/2010)
		* * *
11/10/2010	51	Google Inc.'s ANSWER to Amended Complaint for Patent and Copyright Infringement, Amended COUNTERCLAIM against Oracle America, Inc. byGoogle Inc.. (Zimmer, Donald) (Filed on 11/10/2010) (Entered: 11/10/2010)
		* * *
11/29/2010	60	ANSWER TO COUNTERCLAIM 51 Answer to Amended Complaint,

Date	#	Docket Text
		Counterclaim byOracle America, Inc.. (Peters, Marc) (Filed on 11/29/2010) (Entered: 11/29/2010)
		* * *
09/15/2011	433	ORDER PARTIALLY GRANTING AND PARTIALLY DENYING DEFENDANT'S MOTION FOR SUMMARY JUDGMENT ON COPYRIGHT CLAIM by Judge Alsup granting in part and denying in part 260 Motion for Summary Judgment (whalcl, COURT STAFF) (Filed on 9/15/2011) (Entered: 09/15/2011)
		* * *
09/26/2011	461	ORDER GRANTING UNOPPOSED MOTION FOR PARTIAL SUMMARY JUDGMENT REGARDING 35 U.S.C. 271(f) THEORY, Order by Hon. William Alsup granting 409 Motion for Summary Judgment.(whalcl, COURT STAFF) (Filed on 9/26/2011) (Entered: 09/26/2011)
		* * *
11/15/2011	621	ORDER DENYING MOTION FOR PARTIAL SUMMARY JUDGMENT LIMITING DAMAGES BASED ON PATENT–MARKING STATUTE by Judge Alsup denying 552 Motion for Summary Judgment

Date	#	Docket Text
		(whalcl, COURT STAFF) (Filed on 11/15/2011) (Entered: 11/15/2011)
		* * *
04/16/2012	930	Minute Entry: Jury Trial began on 4/16/2012 before William Alsup (Date Filed: 4/16/2012), Jury Selection held on 4/16/2012 before William Alsup (Date Filed: 4/16/2012). (Court Reporter Kathy Sullivan; Debra Pas.) (dt, COURT STAFF) (Date Filed: 4/16/2012) (Entered: 04/16/2012)
		* * *
04/17/2012	936	Minute Entry: Jury Trial held on 4/17/2012 before William Alsup (Date Filed: 4/17/2012). Witnesses called. Further Jury Trial 4/18/2012 7:30 AM. (Court Reporter Kathy Sullivan; Debra Pas.) (dt, COURT STAFF) (Date Filed: 4/17/2012) (Entered: 04/17/2012)
		* * *
04/18/2012	939	Minute Entry: Jury Trial held on 4/18/2012 before William Alsup (Date Filed: 4/18/2012). Witnesses called. Further Jury Trial 4/19/2012 7:30 AM. (Court Reporter Kathy Sullivan; Debra Pas.) (dt, COURT STAFF) (Date Filed: 4/18/2012) (Entered: 04/18/2012)
		* * *

Date	#	Docket Text
04/19/2012	944	Minute Entry: Jury Trial held on 4/19/2012 before William Alsup (Date Filed: 4/19/2012). Witnessess called. Further Jury Trial set for 4/20/2012 7:30 AM. (Court Reporter Kathy Sullivan; Debra Pas.) (dtS, COURT STAFF) (Date Filed: 4/19/2012) (Entered: 04/19/2012)
		* * *
04/20/2012	947	Minute Entry: Jury Trial held on 4/20/2012 before William Alsup (Date Filed: 4/20/2012). Witnesses called. Further Jury Trial 4/23/2012 7:30 AM. (Court Reporter Kathy Sullivan; Debra Pas.) (dt, COURT STAFF) (Date Filed: 4/20/2012) (Entered: 04/20/2012)
		* * *
04/23/2012	975	Minute Entry: Jury Trial held on 4/23/2012 before Judge William Alsup (Date Filed: 4/23/2012). Witnesses called. Further Jury Trial 4/24/2012 7:30 AM. (Court Reporter Kathy Sullivan; Debra Pas.) (dt, COURT STAFF) (Date Filed: 4/23/2012) (Entered: 04/25/2012)
		* * *
04/24/2012	976	Minute Entry: Jury Trial held on 4/24/2012 before Judge William Alsup (Date Filed: 4/24/2012). Witness called. Plaintiff REST – Phase One. Further Jury Trial set

Date	#	Docket Text
		for 4/25/2012 7:30 AM. (Court Reporter Kathy Sullivan; Debra Pas.) (dt, COURT STAFF) (Date Filed: 4/24/2012) (Entered: 04/25/2012)
		* * *
04/25/2012	977	Minute Entry: Jury Trial held on 4/25/2012 before Judge William Alsup (Date Filed: 4/25/2012). Witnesses called. Further Jury Trial set for 4/26/2012 7:30 AM. (Court Reporter Kathy Sullivan; Debra Pas.) (dt, COURT STAFF) (Date Filed: 4/25/2012) (Entered: 04/25/2012)
		* * *
04/26/2012	992	Minute Entry: Jury Trial held on 4/26/2012 before William Alsup (Date Filed: 4/26/2012). Further Jury Trial set for 4/27/2012 7:30 AM. Charging Conference set for 4/27/2012 02:15 PM in Courtroom 8, 19th Floor, San Francisco before Hon. William Alsup. (Court Reporter Kathy Sullivan; Debra Pas.) (dt, COURT STAFF) (Date Filed: 4/26/2012) (Entered: 04/26/2012)
		* * *
04/27/2012	1019	Minute Entry: Jury Trial held on 4/27/2012 before William Alsup (Date Filed: 4/27/2012). Charging Conference held. Further Jury

Date	#	Docket Text
		Trial set for 4/30/2012 7:30 AM. (Court Reporter Debra Pas.) (dt, COURT STAFF) (Date Filed: 4/27/2012) (Entered: 04/30/2012)
		* * *
04/30/2012	1020	Minute Entry: Jury Trial held on 4/30/2012 before William Alsup (Date Filed: 4/30/2012). Closing Arguments held. Jury Instructions read. Deliberations began. Further Jury Trial set for 5/1/2012 8:00 AM. (Court Reporter Kathy Sullivan; Debra Pas.) (dt, COURT STAFF) (Date Filed: 4/30/2012) (Entered: 04/30/2012)
		* * *
05/01/2012	1058	Minute Entry: Jury Trial held on 5/1/2012 before William Alsup (Date Filed: 5/1/2012). Jury Deliberations continued. Further Jury Trial set for 5/2/2012 8:00 AM. (Court Reporter Kathy Sullivan.) (dt, COURT STAFF) (Date Filed: 5/1/2012) (Entered: 05/03/2012)
		* * *
05/02/2012	1059	Minute Entry: Jury Trial held on 5/2/2012 before William Alsup (Date Filed: 5/2/2012). Jury Deliberations continued. Further Jury Trial set for 5/3/2012 8:00 AM. (Court Reporter Debra Pas.) (dt, COURT STAFF) (Date Filed: 5/2/2012)

Date	#	Docket Text
		(Entered: 05/03/2012)
		* * *
05/03/2012	1060	Minute Entry: Jury Trial held on 5/3/2012 before William Alsup (Date Filed: 5/3/2012). Jury Deliberations continued. Further Jury Trial 5/4/2012 8:00 AM. (Court Reporter Kathy Sullivan.) (dt, COURT STAFF) (Date Filed: 5/3/2012) (Entered: 05/03/2012)
		* * *
05/04/2012	1063	Minute Entry: Jury Trial held on 5/4/2012 before William Alsup (Date Filed: 5/4/2012). Jury Deliberations continued. Further Jury Trial set for 5/7/12 at 8:00 AM. (Court Reporter Kathy Sullivan.) (dt, COURT STAFF) (Date Filed: 5/4/2012) (Entered: 05/04/2012)
		* * *
05/07/2012	1089	JURY VERDICT – Phase One. (dt, COURT STAFF) (Filed on 5/7/2012) (Entered: 05/07/2012)
05/07/2012	1090	Minute Entry: Jury Trial held on 5/7/2012 before William Alsup (Date Filed: 5/7/2012). Jury Deliberations Continued. Verdict read. Phase II – Plaintiffs Opening Statement. Further Jury Trial set for 5/8/2012 7:30 AM. (Court Reporter Kathy Sullivan; Debra Pas.) (dt,

Date	#	Docket Text
		COURT STAFF) (Date Filed: 5/7/2012) (Entered: 05/07/2012)
		* * *
05/08/2012	1100	Minute Entry: Jury Trial held on 5/8/2012 before Willliam Alsup (Date Filed: 5/8/2012). Defendant's Opening Statement made. Witnesses called. Further Jury Trial set for 5/9/2012 7:30 AM. Rule 50 Motion Hearing set for 5/9/2012 01:45 PM in Courtroom 8, 19th Floor, San Francisco before Hon. William Alsup. (Court Reporter Kathy Sullivan; Debra Pas.) (dt, COURT STAFF) (Date Filed: 5/8/2012) (Entered: 05/08/2012)
		* * *
05/09/2012	1174	Minute Entry: Jury Trial held on 5/9/2012 before William Alsup (Date Filed: 5/9/2012). Witnesses Called. Rule 50 Motions argued. Further Jury Trial set for 5/10/2012 7:30 AM. (Court Reporter Kathy Sullivan; Debra Pas.) (dt, COURT STAFF) (Date Filed: 5/9/2012) (Entered: 05/18/2012)
		* * *
05/10/2012	1119	ORDER ON MOTIONS FOR JUDGMENT AS A MATTER OF LAW re 1043 Brief, filed by Google Inc., 1045 MOTION for Judgment as a Matter of Law Oracle's

Date	#	Docket Text
		Corrected Rule 50(A) Motion at the Close of Evidence (WITH TABLES) filed by Oracle America, Inc.. Signed by Judge Alsup on May 10, 2012. (whalc 1, COURT STAFF) (Filed on 5/10/2012) (Entered: 05/10/2012)
		* * *
05/10/2012	1175	Minute Entry: Jury Trial held on 5/10/2012 before William Alsup (Date Filed: 5/10/2012). Witnesses called. Plaintiff REST. Charging Conference held. Further Jury Trial set for 5/11/2012 7:30 AM (Court Reporter Kathy Sullivan; Debra Pas.) (dt, COURT STAFF) (Date Filed: 5/10/2012) (Entered: 05/18/2012)
		* * *
05/11/2012	1123	ORDER GRANTING MOTION FOR JUDGMENT AS A MATTER OF LAW ON DECOMPILED FILES re 1045 MOTION for Judgment as a Matter of Law Oracle's Corrected Rule 50(A) Motion at the Close of Evidence (WITH TABLES) filed by Oracle America, Inc.. Signed by Judge Alsup on May 11, 2012. (whalc 1, COURT STAFF) (Filed on 5/11/2012) (Entered: 05/11/2012)
05/11/2012	1176	Minute Entry: Jury Trial held on 5/11/2012 before William Alsup (Date Filed: 5/11/2012). Witnesses

Date	#	Docket Text
		called. Further Jury Trial set for 5/14/2012 7:30 AM. (Court Reporter Kathy Sullivan; Debra Pas.) (dt, COURT STAFF) (Date Filed: 5/11/2012). (Entered: 05/18/2012)
		* * *
05/14/2012	1177	Minute Entry: Jury Trial held on 5/14/2012 before William Alsup (Date Filed: 5/14/2012). Witnesses called. Defendant Rest. Rebuttal witness called. Plaintiff and Defendant Rest. Further Charging Conference held. Further Jury Trial set for 5/15/2012 7:30 AM.(Court Reporter Kathy Sullivan; Debra Pas.) (dt, COURT STAFF) (Date Filed: 5/14/2012) (Entered: 05/18/2012)
		* * *
05/15/2012	1178	Minute Entry: Jury Trial held on 5/15/2012 before William Alsup (Date Filed: 5/15/2012). Closing Arguments Made. Jury Instructions Read. Deliberation Began. Further Jury Trial set for 5/16/2012 8:00 AM. (Court Reporter Kathy Sullivan; Debra Pas.) (dt, COURT STAFF) (Date Filed: 5/15/2012) (Entered: 05/18/2012)
		* * *
05/16/2012	1165	ORDER DENYING MOTION FOR JUDGMENT AS A MATTER

Date	#	Docket Text
		OF LAW REGARDING REGISTRATION AND OWNERSHIP. Signed by Judge Alsup on May 16, 2012. (whalcl, COURT STAFF) (Filed on 5/16/2012) (Entered: 05/16/2012)
		* * *
05/17/2012	1180	Minute Entry: Jury Trial held on 5/17/2012 before William Alsup (Date Filed: 5/17/2012). Jury Deliberations Continued. Further Jury Trial set for 5/18/2012 7:30 AM. (Court Reporter Debra Pas.) (dt, COURT STAFF) (Date Filed: 5/17/2012) (Entered: 05/18/2012)
		* * *
05/18/2012	1179	Minute Entry: Jury Trial held on 5/16/2012 before William Alsup (Date Filed: 5/18/2012). Deliberations continued. Further Jury Trial set for 5/17/2012 8:00 AM. (Court Reporter Kathy Sullivan.) (dt, COURT STAFF) (Date Filed: 5/18/2012). (Entered: 05/18/2012)
05/18/2012	1186	Minute Entry: Jury Trial held on 5/18/2012 before William Alsup (Date Filed: 5/18/2012). Delibertions continued. Further Jury Trial set for 5/21/2012 8:00 AM.(Court Reporter Kathy Sullivan.) (dtS, COURT STAFF) (Date Filed: 5/18/2012) (Entered: 05/22/2012)

Date	#	Docket Text
		* * *
05/21/2012	1187	Minute Entry: Jury Trial held on 5/21/2012 before William Alsup (Date Filed: 5/21/2012). Deliberations continued. Further Jury Trial set for 5/22/2012 8:00 AM.(Court Reporter Kathy Sullivan.) (dtS, COURT STAFF) (Date Filed: 5/21/2012) (Entered: 05/22/2012)
		* * *
05/22/2012	1199	Minute Entry: Jury Trial held on 5/22/2012 before William Alsup (Date Filed: 5/22/2012). Jury Deliberations continued. Further Jury Trial set for 5/23/2012 8:00 AM.(Court Reporter Kathy Sullivan.) (dt, COURT STAFF) (Date Filed: 5/22/2012) (Entered: 05/24/2012)
		* * *
05/23/2012	1190	JURY VERDICT (Phase II). (dt, COURT STAFF) (Filed on 5/23/2012) (Additional attachment(s) added on 2/20/2019: # 1 Special Verdict Form) (amgS, COURT STAFF). (Entered: 05/23/2012)
		* * *
05/23/2012	1200	Minute Entry: Jury Trial (Phase II) completed on 5/23/2012 before William Alsup (Date Filed: 5/23/2012). Jury Verdict read.

Date	#	Docket Text
		Jury polled, thanked and discharged. (Court Reporter Kathy Sullivan.) (dt, COURT STAFF) (Date Filed: 5/23/2012) (Entered: 05/24/2012)
		* * *
05/30/2012	1201	ORDER DENYING ORACLE MOTION FOR JUDGMENT AS A MATTER OF LAW RE PATENT INFRINGEMENT by Hon. William Alsup denying 1168 Motion for Judgment as a Matter of Law.(whalcl, COURT STAFF) (Filed on 5/30/2012) (Entered: 05/30/2012)
05/31/2012	1202	ORDER RE COPYRIGHTABILITY OF CERTAIN REPLICATED ELEMENTS OF THE JAVA APPLICATION PROGRAMMING INTERFACE by Judge William Alsup [granting 984 Motion for Judgment as a Matter of Law; granting 1007 Motion for Judgment as a Matter of Law; finding as moot 1105 Motion for New Trial]. (whasec, COURT STAFF) (Filed on 5/31/2012) (Entered: 05/31/2012)
05/31/2012	1203	FINDINGS OF FACT AND CONCLUSIONS OF LAW ON EQUITABLE DEFENSES re 1049 Proposed Findings of Fact filed by Oracle America, Inc., 1047 Proposed Findings of Fact filed by

Date	#	Docket Text
		Google Inc.. Signed by Judge Alsup on May 31, 2012. (whalcl, COURT STAFF) (Filed on 5/31/2012) (Entered: 05/31/2012)
		* * *
06/20/2012	1211	FINAL JUDGMENT. Signed by Judge Alsup on June 20, 2012. (whalcl, COURT STAFF) (Filed on 6/20/2012) (Entered: 06/20/2012)
		* * *
07/13/2012	1221	ORDER DENYING MOTION FOR JUDGMENT AS A MATTER OF LAW AND NEW TRIAL by Hon. William Alsup denying 1212 Motion for Judgment as a Matter of Law.(whalcl, COURT STAFF) (Filed on 7/13/2012) (Entered: 07/13/2012)
		* * *
09/04/2012	1242	ORDER DENYING MOTION FOR JUDGMENT AS A MATTER OF LAW AND NEW TRIAL by Hon. William Alsup denying 1222 Motion for Judgment as a Matter of Law.(whalcl, COURT STAFF) (Filed on 9/4/2012) (Entered: 09/04/2012)
		* * *
10/03/2012	1243	NOTICE OF APPEAL to the Federal Circuit as to 1211 Judgment by Oracle America, Inc..

Date	#	Docket Text
		Filing fee $ 455, receipt number 0971-7174810. – PLAINTIFF ORACLE AMERICA, INC.'S NOTICE OF APPEAL – Appeal Record due by 11/2/2012. (Jacobs, Michael) (Filed on 10/3/2012) (Entered: 10/03/2012)
		* * *
10/04/2012	1247	Amended NOTICE OF APPEAL to the Federal Circuit as to 1242 Order on Motion for Judgment as a Matter of Law by Google Inc.. See receipt number 0971–7176729 Appeal Record due by 11/5/2012. (Hirsch, Steven) (Filed on 10/4/2012) (Entered: 10/04/2012)
10/04/2012	1248	NOTICE OF CROSS APPEAL as to 1243 Notice of Appeal to the Federal Circuit, by Google Inc.. Filing fee $ 455, receipt number 0971-7177323. Appeal Record due by 11/5/2012. (Hirsch, Steven) (Filed on 10/4/2012) (Entered: 10/04/2012)
		* * *
08/12/2015	1292	COMPLAINT Plaintiff Oracle's Supplemental Complaint for Copyright Infringement against Google Inc.. Filed byOracle America, Inc.. (Bicks, Peter) (Filed on 8/12/2015) (Entered: 08/12/2015)
		* * *
02/05/2016	1479	ORDER RE GOOGLE'S MOTION

Date	#	Docket Text
		TO STRIKE by Hon. William Alsup granting 1454 Motion to Strike.(whalcl, COURT STAFF) (Filed on 2/5/2016) (Entered: 02/05/2016)
		* * *
05/02/2016	1781	MEMORANDUM OPINION RE GOOGLE'S MOTION IN LIMINE NO. 2 REGARDING NEW PRODUCTS by Judge William Alsup [granting 1559 Motion in Limine]. (whasec, COURT STAFF) (Filed on 5/2/2016) (Entered: 05/02/2016)
		* * *
05/09/2016	1856	Minute Entry for proceedings held before Hon. William Alsup: Jury Trial began on 5/9/2016. Jury Selection Completed. Further Jury Trial set for 5/10/2016 7:30 AM. Total Time in Court 6:05. Court Reporter Name Kathy Sullivan; Pam Batalo. Plaintiff Attorney Peter Bicks. et al.Defendant Attorney Robert Van Nest. et al.Attachment minute order.(dl, COURT STAFF) (Date Filed: 5/10/2016) Modified on 5/11/2016 (dl, COURT STAFF). (Entered: 05/10/2016)
		* * *
05/10/2016	1857	Minute Entry for proceedings held before Hon. William Alsup: Jury Trial held on 5/10/2016. Witness

Date	#	Docket Text
		called. Further Jury Trial set for 5/11/2016 7:30 AM. Total Time in Court 5:40. Court Reporter Name Kathy Sullivan; Pam Batalo. Plaintiff Attorney Peter Bicks. Defendant Attorney Robert Van Nest. Attachment minute order.(dl, COURT STAFF) (Date Filed: 5/10/2016) (Entered: 05/10/2016)
		* * *
05/11/2016	1864	Minute Entry for proceedings held before Hon. William Alsup: Jury Trial held on 5/11/2016. Witnesses Called. Further Jury Trial set for 5/12/2016 7:30 AM.Total Time in Court 5:38. Court Reporter Name Kathy Sullivan; Pam Batalo. Plaintiff Attorney Peter Bicks. et al.Defendant Attorney Robert Van Nest. et al.Attachment minute order.(dl, COURT STAFF) (Date Filed: 5/11/2016) (Entered: 05/11/2016)
		* * *
05/12/2016	1879	ORDER IN LIMINE RE ORACLE'S MOTION RE DR. RODERIC CATTELL [re 1824 MOTION in Limine to Exclude Testimony of Roderic Cattell filed by Oracle America, Inc.]. Signed by Judge William Alsup on 5/12/2016. (whasec, COURT STAFF) (Filed on 5/12/2016) (Entered: 05/12/2016)
		* * *
05/12/2016	1882	Minute Entry for proceedings held before Hon. William Alsup: Jury Trial held on 5/12/2016. Witness called. Further Jury Trial set for 5/13/2016 at 7:30 AM; Total Time in Court: 5 hours; 33 minutes. Court Reporter: Pam Batalo. Plaintiff Attorneys: Peter Bicks; Annette Hurst; Matthew Bush; Lisa

Date	#	Docket Text
		Simpson; Gabriel Ramsey; Christina Von Der Ahe; George Saab (Corp Rep). Defendant Attorneys: Robbert Van Nest; Bruce Baber; Christa Anderson; Daniel Purcell; Matthias Kamber; Michael Kwun; Reid Mullen; Maya Karwande; Steven Ragland; Catherine Locavera (Corp Rep). Attachment: Minute Order.(afmS, COURT STAFF) (Date Filed: 5/12/2016) (Entered: 05/12/2016)
		* * *
05/13/2016	1886	Minute Entry for proceedings held before Hon. William Alsup: Jury Trial held on 5/13/2016. Witnesses called. Further Jury Trial set for 5/16/2016 at 7:30 a.m. Total Time in Court: 5:05. Court Reporter: Pam Batalo. Plaintiff Attorneys: Peter Bicks; Annette Hurst; Matthew Bush; Lisa Simpson; Gabriel Ramsey; Christina Von Der Ahe. Defendant Attorney: Robert Van Nest; Bruce Baber; Christa Anderson; Ed Bailey; Daniel Purcell; Matthias Kamber; Michael Kwun; Reid Mullen; Maya Karwande; Steven Ragland. Attachment: Minute Order.(afmS, COURT STAFF) (Date Filed: 5/13/2016) (Entered: 05/13/2016)
		* * *

Date	#	Docket Text
05/16/2016	1902	Minute Entry for proceedings held before Hon. William Alsup: Jury Trial held on 5/16/2016. Witnesses called. Further Jury Trial set for May 17, 2017 at 7:30 A.M. Total Time in Court: 5:26. Court Reporters: Pam Batalo; Kathryn Sullivan. Plaintiff Attorneys: Peter Bicks; Annette Hurst; Matthew Bush; Lisa Simpson; Gabriel Ramsey; Christina Von Der Ahe. Defendant Attorneys: Robert Van Nest; Christa Anderson; Daniel Purcell; Ed Bailey; Eugene Paige; Bruce Baber; Matthias Kamber; Reid Mullen; Michael Kwun; Maya Karwande; Steven Ragland. Interpreter N/A.Attachment: Minute Order.(afmS, COURT STAFF) (Date Filed: 5/16/2016) (Entered: 05/16/2016)
		* * *
05/17/2016	1911	Minute Entry for proceedings held before Hon. William Alsup: Jury Trial held on 5/17/2016. Witnesses called. Further Jury Trial set for 5/18/2016 at 7:30 a.m. Total Time in Court: 5:25. Court Reporters: Pam Batalo/Kathryn Sullivan. Plaintiff Attorneys: Peter Bicks; Annette Hurst; Matthew Bush; Lisa Simpson; Gabriel Ramsey; Christina Von Der Ahe; Alyssa Caridis. Defendant Attorneys:

Date	#	Docket Text
		Robert Van Nest; Bruce Baber; Christa Anderson; Ed Bailey; Daniel Purcell; Matthias Kamber; Eugene Paige; Michael Kwun; Reid Mullen; Maya Karwande; Steven Ragland. Interpreter N/A.Attachment: Minute Order.(afmS, COURT STAFF) (Date Filed: 5/17/2016) (Entered: 05/17/2016)
		* * *
05/18/2016	1922	Minute Entry for proceedings held before Hon. William Alsup: Jury Trial held on 5/18/2016. Witnesses called. Further Jury Trial set for May 19, 2016 at 7:30 a.m. Total Time in Court: 5:37. Court Reporter: Kelly Polvi. Plaintiff Attorneys: Peter Bicks; Annette Hurst; Matthew Bush; Lisa Simpson; Gabriel Ramsey; Christina Von Der Ahe; Alyssa Caridis. Defendant Attorneys: Robert Van Nest; Bruce Baber; Christa Anderson; Ed Bailey; Daniel Purcell; Matthias Kamber; Eugene Paige; Michael Kwun; Reid Mullen; Maya Karwande; Steven Ragland. Attachment: Minute Order.(afmS, COURT STAFF) (Date Filed: 5/18/2016) (Entered: 05/18/2016)
		* * *

Date	#	Docket Text
05/19/2016	1927	AMENDED 1926 MINUTE ENTRY: for proceedings held before Hon. William Alsup: Jury Trial and Charging Conference held on 5/19/2016. Witnesses called. Plaintiff and Defendant RESTS. Further Jury Trial set for May 23, 2016 at 7:30 a.m. Total Time in Court: 5:32. Court Reporters: Pam Batalo; Kathryn Sullivan. Plaintiff Attorneys: Peter Bicks; Annette Hurst; Lisa Simpson; Gabriel Ramsey; Matthew Bush; Christina Von Der Ahe; Alyssa Caridis; Andrew Silverman. Defendant Attorneys: Robert Van Nest; Christa Anderson; Bruce Baber; Ed Bailey; Daniel Purcell; Michael Kwun; Steven Ragland; Reid Mullen; Eugene Paige; Matthias Kamber; Maya Karwande. Interpreter: N/A.Attachment: Minute Order. (afmS, COURT STAFF) (Filed on 5/19/2016) (Entered: 05/19/2016)
		* * *
05/23/2016	1947	Minute Entry for proceedings held before Hon. William Alsup: Jury Trial held on 5/23/2016. Closing Arguments made. Jury Instruction read. Jury Deliberation began. Further Jury Trial set for 5/24/2016 7:45 AM.Total Time in Court 5:09. Court Reporter Name

Date	#	Docket Text
		Kathy Sullivan; Pam Batalo. Plaintiff Attorney Peter Bicks. Defendant Attorney Robert Van Nest. Attachment minute order.(dl, COURT STAFF) (Date Filed: 5/23/2016) (Entered: 05/23/2016)
		* * *
05/24/2016	1969	Minute Entry for proceedings held before Hon. William Alsup: Jury Trial held on 5/24/2016. Deliberation continued. Further Trial set for 5/25/2016 7:30 AM. Total Time in Court 55 minutes. Court Reporter Name Kathy Sullivan. Plaintiff Attorney Peter Bicks. Defendant Attorney Robert Van Nest. Attachment minute order.(dl, COURT STAFF) (Date Filed: 5/24/2016) (Entered: 05/25/2016)
		* * *
05/25/2016	1972	Minute Entry for proceedings held before Hon. William Alsup: Jury Trial held on 5/25/2016. Deliberation Continued. Further Jury Trial set for 5/26/16 7:30 AM.Total Time in Court 1:12. Court Reporter Name Kathy Sullivan. Plaintiff Attorney Peter Bicks. Defendant Attorney Robert Van Nest. Attachment minute order.(dl, COURT STAFF) (Date Filed: 5/25/2016) (Entered: 05/25/2016)

Date	#	Docket Text
		* * *
05/26/2016	1982	JURY VERDICT. (dl, COURT STAFF) (Filed on 5/26/2016) (Entered: 05/26/2016)
05/26/2016	1983	Jury Notes. (dl, COURT STAFF) (Filed on 5/26/2016) (Entered: 05/26/2016)
05/26/2016	1984	Minute Entry for proceedings held before Hon. William Alsup: Jury Trial completed on 5/26/2016. Verdict Reached. Jury polled, thanked and discharged.Total Time in Court 10 minutes. Court Reporter Name Pam Batalo. Plaintiff Attorney Peter Bicks. Defendant Attorney Robert Van Nest. Attachment minute order.(dl, COURT STAFF) (Date Filed: 5/26/2016) (Entered: 05/26/2016)
		* * *
06/08/2016	1988	ORDER DENYING RULE 50 MOTIONS by Judge William Alsup [denying 1914 Motion for Judgment as a Matter of Law; denying 1937 Motion for Judgment as a Matter of Law]. (whasec, COURT STAFF) (Filed on 6/8/2016) (Entered: 06/08/2016)
06/08/2016	1989	FINAL JUDGMENT in favor of defendant Google Inc., and against plaintiff Oracle America, Inc. Signed by Judge William Alsup on

Date	#	Docket Text
		6/8/2016. (whasec, COURT STAFF) (Filed on 6/8/2016) (Entered: 06/08/2016)
		* * *
09/27/2016	2070	ORDER DENYING RENEWED MOTION FOR JUDGMENT AS A MATTER OF LAW AND MOTION FOR A NEW TRIAL by Judge Alsup denying 1993 Motion for Judgment as a Matter of Law; denying 1997 Motion for New Trial. (whalcl, COURT STAFF) (Filed on 9/27/20161 (Entered: 09/27/20161
10/26/2016	2071	NOTICE OF APPEAL to the Federal Circuit by Oracle America, Inc.. Filing fee $ 505, receipt number 0971-10881532. Appeal Record due by 11/25/2016. (Bicks, Peter) (Filed on 10/26/2016) (Entered: 10/26/2016)
		* * *
11/09/2016	2073	NOTICE OF CROSS APPEAL to the Federal Circuit (USCA – Federal Circuit 17-1202) by Google Inc.. Filing fee $ 505, receipt number 0971-10918837. Appeal Record due by 12/9/2016. (Baber, Bruce) (Filed on 11/9/2016) Modified on 11/16/2016 (ecgS, COURT STAFF). (Entered: 11/09/2016)
		* * *

[648] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOOGLE, INC.,

Defendant.

San Francisco, California
April 19, 2012

Before the Honorable William H. Alsup

TRANSCRIPT OF JURY TRIAL PROCEEDINGS

---

* * *

[683] the two circles?

A. That intersection would be is a lot smaller than it's shown. It was shown a little bigger because otherwise it would be very hard to see.

Q. And we're looking, for the record, at Slide 20.

THE COURT: Can I ask a question? Now, the 37 APIs that we're most concerned about in this case, are they all within that little shaded area or are they somewhere else?

THE WITNESS: Your Honor, most of them are – the bulk of them are outside of that little shaded area.

THE COURT: In the purple area?

THE WITNESS: Correct, sir.

BY MR. JACOBS:

Q. Without any kind of class library at all – I want you to assume no class library – what kind of programming could you do with the Java programming language itself?

A. With no class library at all, you could do very little.

Q. What would be the first increment you would need to put in to be able to do something meaningful?

A. To do something meaningful you would need a way to get results out of the program. As the Java – take the Java language just by itself. It can – it can take some parameters to a single method from the command line. If you're typing at the computer, you can type Java, then the name of your program, and then some words. And those words are passed in to the [684] program as strings. So there is limited input there, but there is no way for the program to do output. It can compute. It can do a lot of computation inside, but it can never communicate the results of that computation.

Q. Now, changing the assumption a little bit, you are told you can create a class library, but you can't copy the Java platform API specification for any APIs.

Can you create your own class library to perform the functions that you just described?

MR. PURCELL: Objection. It's a hypothetical.

THE COURT: Sustained.

BY MR. JACOBS:

Q. Does the Java programming language give you the capability to write your own APIs and class libraries?

A. Yes, it does.

Q. Does one need to use Sun's class libraries, Oracle's class libraries, Sun's APIs, Oracle's APIs in order to program in the Java programming language?

A. At the very least you need to use the few that are tightly related to the Java programming language.

THE COURT: When you say that, you're pointing at something. What are you pointing at?

THE WITNESS: Sir, I'm pointing to the Exhibit 1062.

THE COURT: And so the few are everything on that page?

* * *

[762] implementing the same kind of collections on distributed systems, where the data was stored on multiple machine machines so that if one of the machines crashed you wouldn't lose any of your data. You could keep using the data on the other machines. That's called fault tolerance.

And I had to write APIs to use these distributed collections, much as the Java collections framework is a set of APIs to use collections.

Q. Dr. Bloch, is it fair to say you've been involved with APIs for your entire 30 years of education and professional life?

A. That's a fair statement.

Q. Dr. Bloch, how did you learn the Java language?

A. I learned it on an airplane, reading a copy of a book called *Java In A Nutshell*, by a guy named David Flanagan.

Q. Do you recall approximately when that was?

A. Yeah. It was on my way to the job interview for the Java job at Sun. So that would be in 1996, around, you know, May or something. I don't know.

Q. And when you learned the Java language from the *Java In A Nutshell* book, did that include any discussion of APIs?

A. Yeah. In fact, the entire API set of the platform is summarized in that book.

Q. And, now, Mr. Jacobs asked you about some of your writings. You have written a book, yourself, about the Java

* * *

[769] And, also, it's a specification. That API should hopefully be precise enough that it allows other people to do an independent lead limitation of the same API.

Q. Dr. Bloch, is an API like a blueprint in any way?

A. It's not – not really like a blueprint because a blueprint tells you how to build something. And you can build something in many different ways that implements the same API.

The API, as I said, it tells you how to talk to something rather than how to build it. So I don't think of an API as being like a blueprint.

Q. Does an API itself tell you how to build anything?

A. Not really.

Q. Okay. Do you believe that an API serves as a blueprint even for the libraries that house the APIs?

A. Once again, you know, it's a stretch. It's not really like a blueprint because of the fact that a blueprint tells you how to implement something. It basically says use this timber here, and this length should be this.

API specifications, APIs don't do that. They are requirements. They tell you what this thing that you're building has to do. How to talk to it. How to write a program on top of it. But they don't tell you how to build it.

Q. In the Java language, Dr. Bloch, what is it that determines the organization of the code libraries that implement the APIs?

[770] A. The names of the methods basically determine that, because names in Java, they have three parts. It's called a fully-qualified name. It consists of the class – actually, the package, the class, and then the method or field.

So the whole name of something might be something like Java.lang.math.cos. So it's not math. It's a package. And then – sorry Java.lang is a package. Math is a class. And then cos is the cosine function. And so the name determines the organization.

Q. And is this notion of a fully-qualified name something that's a part of the *Java Language Specification*?

A. Yes. I'm certain that term is in there.

Q. If you could give us just an example. You just described in general terms package.class.method would be the format of – at least a start of part of a fully-qualified name. Correct?

A. Uh-huh.

Q. Use using the example you just gave, what would be the fully-qualified name under the *Java Language Specification* for the math function you just described?

A. All right. So Java.lang is the name of the package.

Q. For the fully-qualified name, would there be anything before the words "Java"?

A. No. And it has to be lower case, by the way. Lower case Java period l-a-n-g period. And then we have an upper case M- – because the class names are upper case – a-t-h. And [771] then a period. And then the method name, which I believe is cos, c-o-s. I don't do a lot of trigonometry.

Q. So that is the name of the specific method, and that's what would be considered to be a fully-qualified name under the language specification (indicating)?

A. Yes.

THE COURT: Where's the API part on all that?

THE WITNESS: That is part of the Java SE APIs.

So this is one small element in the API. When you say the API, I think I know what you mean, as in, yes, Java.lang is the package. You if you talk about – sometimes people here in these trials in form we talk about a number, like 37 APIs. They mean 37 packages. And Java.lang would be one of those packages.

THE COURT: Can you do this. Circle the word package.

(Counsel complies.)

THE COURT: Is that what you're saying, that is, one API would correspond to that package?

THE WITNESS: So, honestly, that isn't a terminology that I heard before this trial. You know, quantizing APIs is kind of hard.

But in this trial people have been using the number of APIs to be the number of packages. When they say, oh, yes, this contains 43 APIs, they really mean 43 packages. So, yes.

[772] THE COURT: But your own document that you did in Holland had the word "API." You've heard of that word before.

THE WITNESS: I have. But what I'm saying is, to me, a package is a part of an API. A class is a part of an API. A method is a part of an API.

And I don't talk about a number of APIs any more than I would talk about a number of meat. You know, meat is measured in pounds.

There's no good metric for APIs, really.

THE COURT: Well, look. We are trying to just get the terminology down. It says package.class.method. Now, is the package the API?

Help us understand what more you need to put up there so that we know what API is versus package, or whether they are the same. So you describe it however you want. We're trying to understand.

THE WITNESS: So think of it as, you know, your city, your state, and your street. They are all part of

your address. So, you know, the APIs here are the addresses, and the package is the city. The class is the street. And method is the house number, if you want. It's not a great analogy, but you get the idea.

THE COURT: Yes, but where is the API part?

THE WITNESS: All three of them are parts of it.

THE COURT: And could, in the same API, there be the [773] same package but a different class, and yet a different method?

THE WITNESS: Yes. So, for example, in Java.util we have that collections framework that I wrote. And there's one class for lists, and another class for sets, which can contain duplicates. So those are two different classes in the same package.

THE COURT: And so it sounds like the API is at the package level. No? Yes?

THE WITNESS: You know, every language has its own way of breaking these things up. But, actually, class is somewhat more fundamental to Java than package, simply because the Java – at runtime, you know, entire classes are loaded at once.

But, you know, as I say, I think the address metaphor is pretty good. Which is a more important part of your address, the street or the city?

THE COURT: So you didn't answer my question.

THE WITNESS: I tried.

THE COURT: Either say yes, no, or it's impossible to answer yes or no.

The question was, it sounds like the API is at the package level.

THE WITNESS: Yes, but it's also at the class and method levels.

THE COURT: All right. Thank you.

[774] I'm sorry for the interruption. Please continue.

MR. BABER: Thank you, Your Honor.

BY MR. BABER:

Q. You were talking about, Dr. Bloch, the concept of a fully-qualified name in the Java language. Once a fully-qualified name for a specific method has been chosen, how does that name dictate the organization or where that piece of code will reside in the libraries?

A. Well, if it's a public API, then you – you have to put – an entire class goes in one file. And the file has to have the name of the class.

So, you know, in the case of our Java.lang.math class there actually is a file called math.Java. And at the top of that file there's a line that says "Package Java.lang." And that means that class, Math, is inside the package Java.lang.

Q. So just to be clear, the class name is Java.lang.-math, correct?

A. Correct. And that – by the way, a fully-qualified class name is the class including the package.

Q. And then once the name for an individual method or some other element of the APIs has been chosen, does that name then tell you where you can find that code?

A. Yes, it does.

Q. Okay. And does the language specification for the Java programming language give you any choice as to how to compose a [775] fully-qualified name?

A. I'm not sure what you mean, sorry, by "compose."

Q. Does it have rules about what fully-qualified names have to look like?

A. Yeah, it has conventions.

Q. And prior to this lawsuit, Dr. Bloch, have you ever heard any discussion about the structure, sequence, and organization of APIs?

A. Actually, no. This is the first time I've heard that term.

Q. Dr. Bloch, how do the APIs and the language relate to each other (indicating)?

A. Uhm, well, in a couple of ways. Language lets you write your own APIs.

MR. JACOBS: Objection, Your Honor. So the record is clear, Mr. Baber is holding up the *Java Language Specification* to the witness.

THE WITNESS: Okay.

THE COURT: I'm not sure. I was maybe two sentences back. So you may have a good objection and I just don't understand it. He's –

MR. JACOBS: Objection, the question was vague. And on the record he's holding up the *Java Language Specification*. The witness may be answering two different questions.

THE COURT: That's a good point.

* * *

[904] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOOGLE, INC.,

Defendant.

San Francisco, California
April 20, 2012

Before the Honorable William H. Alsup

TRANSCRIPT OF JURY TRIAL PROCEEDINGS

---

* * *

[961] Q. And did you have occasion to use Java programming language in your work at Danger?

A. Umm, I wrote some of the Danger library code and a little bit of application code in Java.

Q. How did you actually learn how to program in Java?

A. Umm, I think my first encounter with Java again was in 1995 when the language was first released by Sun.

It was interesting at the time because it was this new programming language. And I think I mostly learned by, you know, just tinkering with the compiler, writing little programs and reading some tutorials that had been posted online about the language.

Q. Did you ever read any books about Java to help learn how to program in the language?

A. I think at one point I borrowed or purchased a copy of the *Java Programming Language*, which was an introductory book in Java.

Q. All right. And did those materials you reviewed to learn how to program in Java include discussion of Java APIs?

MR. NORTON: Objection, leading.

THE COURT: What extent, if at all. Remember magic words.

BY MS. ANDERSON:

Q. To what extent, if at all, did the materials you reviewed include discussion of Java APIs?

[962] A. Pretty much all examples of how to write programs in the language involved use of sort of standard Java libraries and APIs.

Q. As a programmer in Java over the years, have you had an understanding as to whether or not the language is free for use?

A. My understanding like all, you know, other similarly – and, actually, I'm not aware of programming languages that aren't free for use.

Q. And how about Java APIs? Have you had an understanding over the years as a Java programmer as to whether Java APIs were free for use?

A. My understanding is they would be. Otherwise, how could you write meaningful programs without, you know. . .

Q. You were asked some questions earlier about the time you spent working at the company called Danger related to development of the Hiptop; do you recall generally those questions?

A. I do recall.

Q. You also testified that you had learned that Danger took a license from Sun eventually, is that right?

A. That was my understanding, yes.

Q. Did you have an understanding as a Danger employee as to why it took that license?

A. My understanding was that we felt that the company –

* * *

[1018] A. Sure. I have a bachelor's degree in physics and computer science from Clarkson University. And I have a master's degree in computer science from Western Polytechnic Institute.

Q. And are you familiar with the Java programming language?

A. Yeah.

Q. When did you first learn to program in the Java language?

A. First learned as an undergraduate, I believe in the – my last two years of college, which I think would have been '97-'98.

Q. How did you learn to program in the Java language?

A. I found a online tutorial like a web page that kind of taught you how to program. So I sort of self-taught myself from a website.

Q. And did that tutorial include any instruction about the Java APIs?

A. Yeah, definitely.

Q. And in your experience – so how many years have you been programming in Java?

A. Off and on, probably about 10 to 12, I guess.

Q. And in your 10 to 12 years of programming in the Java programming language, have you ever written a Java program that didn't use the Java APIs?

A. Uhm, no.

Q. You testified a little bit about compatibility and in particular about the Compatibility Definition Document and the

* * *

[1646] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOOGLE, INC.,

Defendant.

San Francisco, California
April 25, 2012

Before the Honorable William H. Alsup

TRANSCRIPT OF JURY TRIAL PROCEEDINGS

---

* * *

[1687] Are you talking about programs that were previously written in Java or that were not previously written in Java?

THE WITNESS: That were not previously written in Java.

THE COURT: Then the transcript was correct. All right.

THE WITNESS: I'll slow down a little bit.

THE COURT: Okay.

BY MR. VAN NEST:

Q. How many engineers at Google worked on the Android platform during this period from '05 to '08?

A. Well, we started with – you know, we were only eight people when we got acquired, and I think four or five of those were actual engineers. And then over the process of the development of the platform, we hired a bunch of people. By the time we shipped Version 1.0 in 2008, there were about 85 or 90 engineers on the project.

Q. And did that number continue to grow from 2008 on?

A. Yes. It continues to grow.

Q. Can you give the jury an estimate of how many lines of source code were written or contributed to make up the whole stack?

A. So, yeah. I mean, it's not done. It's always evolving and it's always getting bigger. Every time somebody comes into work, they are adding functionality to the platform.

* * *

[1769] already in evidence in this case. Do you recognize this?

A. I recognize the cover, certainly.

Q. And what does it look like to you?

A. This looks like the *Java Language Specification*, the book that I just mentioned.

Q. Okay. And was this one of the books you read to learn how to program in Java?

A. It was.

Q. All right. Did you actually read this book?

A. Uhm, cover to cover, actually.

Q. Did the documentation you reviewed to learn how to program in Java include discussion of Java APIs?

A. Uhm, I'm sorry, say again please.

Q. Sure. Did the documents you reviewed, the treatises and things that you reviewed to learn how to program in Java include any discussion of Java APIs?

A. Oh, yeah, absolutely.

Q. And throughout the course of your history as a Java programmer, did you consider yourself free to use the Java programming language?

A. Oh, yeah, I did.

Q. And also during your career in working in Java programming, have you had an expectation as to whether or not you could use Java APIs as part of programming in the language?

A. It's really impossible to use the language productively

* * *

[1781] THE COURT: Is an API the same thing as a package? If not, what is the difference?

THE WITNESS: Well, you can kind of slice and dice APIs in many ways. Sometimes an API – and it kind of depends on context.

You might – sometimes you might talk about an API being just a single method. So you could say, well, Math.max, that's an API, but you could also talk about the API of Java.lang or, you know, the API – or you

could even say the API of a set of – of a set of packages. And it's all kind of contextual.

THE COURT: Is there a "yes" or "no" answer to my question?

THE WITNESS: I'm sorry.

THE COURT: I asked: Is an API the same thing as a package?

THE WITNESS: Not necessarily, no.

THE COURT: Is it ever the same thing?

THE WITNESS: It could be, yes.

BY MS. ANDERSON:

Q. Let's talk about the core library and the work that you were doing. Did you have a role in determining which packages would be implemented as part of Android's core library?

A. I did.

Q. And what was your role?

[1782] A. My role was to, using my sort of experience and taste, figure out a set of – a set of packages that are associated – that would be associated with the Java programming language, to figure out that set of packages that made sense to implement in the context of Android.

Q. Did your determination of what packages would be implemented in the core library have anything to do with what you thought were expectations of Java language programmers?

A. Yes, absolutely.

Q. And what did it have to do with that?

A. Well, you know, I was talking a bit about, like, what's in people's heads as part of what it means to be an API. And so as a Java programmer, as, say, a typical Java programmer, there's certain of these APIs which you just sort of, like, fundamentally think of as kind of part of – part of the system that you can just use without really having to think too much about it.

And there are other – there are other packages where it might – you know, it might not be necessary, but it would be surprising to not find them. And, you know, there's kind of various – you know, you can sort of, like, make various determinations about, say, any given API, any given package, any given method, any given class to figure out – you know, to at least sort of have a best guess at, well, if I were a programmer programming on this system, would I – would I [1783] expect that to be there? Would I want it to be there? Or would I just not miss it?

And my job was sort of to kind of sift through all of that and come up with a nice and consistent set of APIs that we have would then implement and provide to developers.

Q. Was it your goal to assure that the packages that you think Java language programmers would expect to see there, to be able to use the Java language, be present in the core library?

A. Yes.

Q. Are you familiar generally with the idea that there are certain Java Platforms out there; Java ME, Java SE, Java EE? Have you heard about those generally?

A. I recognize those names. Yes.

Q. Did Android implement all the API packages present in any particular Java Platform?

A. No.

Q. All right. And why not?

A. That wasn't a goal of the project. The goal of the project was to provide something that was familiar to developers. It wasn't to provide any particular preexisting set of packages.

Q. Did you make any judgments in deciding what packages would be implemented in the core library based on whether or not certain APIs are even appropriate for a mobile platform?

[1784] A. Yeah, absolutely.

Q. And, please, explain what you mean by that?

A. Well, if you look at the – say, the universe of packages that have been made for the Java language in general, some of them just don't really apply in – or, you know, didn't apply in the case of what we were doing with Android.

And you can remember that, again, the point was to be a good mobile platform and there are certain constraints that that makes. You know, you can assume that the thing that you're running on is running on a battery, and that's – that's a particular limitation. You can know that there's going to be less memory available than, say, on a desktop or a server. You'll know that, say, the typical CUP speed is going to be slower than you would find on a – you know, on a desktop or server. And you'll also know that – just the sorts of things that you would do as a mobile application are going to be different than the things that you would do if you're, say, sitting in a data center running a web server, for example.

It's just – there's different – there are different needs. And so to the extent that some of those needs are represented in potential Java packages, those are Java packages that we wouldn't necessarily – you know, or, we wouldn't really want to – to have an implementation for, especially in that it takes – implementation takes up space and, you know, storage space on a mobile device an also limited.

[1785] Q. So did you, in fact, exclude from the API packages in the core library certain Java language APIs because you believed they were not appropriate for the mobile platform?

A. That's right.

Q. With respect to the API packages that Google actually did implement as part of the core library, did Google use Sun's source code for that implementation?

A. No.

Q. All right. Where did the source code implementation come from for the API packages that are implemented as part of the core library?

A. There are a few different sources. Would you like me to enumerate?

Q. Yes. Please explain to the jury, generally speaking, where the source code came from?

A. Okay. One of the bigger sources of code that we used was a project called Apache Harmony. Apache Harmony is itself a large body of code written in Java and other languages, and it – it, itself, is an implementation of a Java language platform.

And Harmony itself wasn't appropriate to use entirely, but as an Open Source project it was – it was actually a, you know, sort of expected use of its code for it

to be incorporated into other Open Source projects as Android was and, in fact, the license was exactly compatible. So that [1786] was one major source.

Another source was actually a project called Bouncy Castle, and it's an implementation of a bunch of APIs in Java that made sense to include in Android.

There are a few other Open Source libraries that we used. And, in addition, we wrote a lot of code from scratch ourselves and we had, we had a third-party contractor also write a bunch of the – a bunch of the code.

Q. And with respect to the 37 packages that we're talking about in this case, did any of the code for the 37 purely come from Apache or purely come from Google?

A. I think if you look at any given package, it was – it was always – it always ended up being a mix of code from Google and code from, you know, one of the other sources.

Q. Okay.

THE COURT: May I ask a question? Can I ask a question on this subject?

MS. ANDERSON: Of course, your Honor. Thank you.

THE COURT: I'm trying to understand, and maybe the jury is trying to understand, and they are probably way ahead of me, but what was and was not the same as between Java on the one hand and Android on the other?

Here is what I hear you saying, but you tell me if this is right. Here is the part that was the same, the name; true?

[1787] THE WITNESS: You mean, of the – say, of a method or a class?

THE COURT: Or – yeah, the name. I guess – what goes into a name? You tell me.

THE WITNESS: So maybe – maybe this is what you're trying to get at. So let's just talk about -

THE COURT: Go ahead.

THE WITNESS: So let's just narrow and talk about Math. max for a second.

THE COURT: Okay.

THE WITNESS: The name of the class Math or in the case of Java it's going to be Java.lang.Math, that is going to have to be represented in implementation for – of – of that API.

The name "max" is going to have to show up in the source code as well.

THE COURT: It's going to be the same on the Java version and the Android version?

THE WITNESS: That's right.

THE COURT: All right. And as I understand it, you, in fact, wanted to do that so that the programming community would feel comfortable using the same terminology?

THE WITNESS: Yeah. And, actually, not even just a matter of comfort, but there's a lot of source code out there that wasn't – you know, wasn't written by – well, that was [1788] written by lots of people that already existed that could potentially work just fine on Android. And if we went and changed all the names of things, then that source code wouldn't just work –

THE COURT: All right. So the names are the same literally symbol-by-symbol.

THE WITNESS: Uh-huh.

THE COURT: Java versus Android, true?

THE WITNESS: If I understand you correctly, yes.

THE COURT: All right. And then the declarations, the same, right?

THE WITNESS: Umm, the – there's a little – a little bit of leeway in the declarations in that there are – there are variable names that are kind of built into declarations in the Java programming language, and there would be leeway in those, but in the rest of the declaration this would be none.

THE COURT: So like in the example of max where you can put in two – compare two numbers, you might have X and Y and they may have A and B.

THE WITNESS: That's right.

THE COURT: But otherwise it's the same?

THE WITNESS: That's right.

THE COURT: All right. But then those are the parts that are the same.

* * *

[1879] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOOGLE, INC.,

Defendant.

San Francisco, California
April 26, 2012

Before the Honorable William H. Alsup

TRANSCRIPT OF JURY TRIAL PROCEEDINGS

---

* * *

[1961] just walk up to it and say, "Beep." You would need to know, okay, how do I generate sound? And what are the different functions that are available for me to generate sound? And now how do I write an instruction in such a way that I can use the underlying sound library to cause a beep?

THE COURT: From memory do you know what the–

THE WITNESS: I do not.

THE COURT: You would have to look it up somewhere?

THE WITNESS: I'd have to look it up. I'd have to look at the specs to know that.

THE COURT: Okay. Go ahead.

BY MR. VAN NEST:

Q. Mr. Schwartz, did Sun promote the Java language APIs along with the language?

A. Absolutely. We had to, if you wanted to see that language be broadly accepted.

So it's insufficient to just give you a language because what do you do with it? I mean, how do you now write an application?

So those APIs enabled people to write really full, complete applications that leveraged all the technology that was underlying the platform. So the combination of the language and the APIs, the distribution of those across the world, is what enabled the effect we were seeking, which is broad scale adoption of the platform that would allow us to [1962] bypass Microsoft Windows.

Q. So were the APIs simply marketed along with the language? In other words, free and available for everyone?

A. Yes. Absolutely. We talked about open APIs, and then you compete on implementations. And what that means is we all had the same set of APIs, but we would then create products, the virtual machine specifically or the technology that underlies the language, to go off and perform – I'm doing a bad job of explaining.

Q. Let me ask this question, Mr. Schwartz. You're doing a fine job.

Were the APIs ever sold or licensed separately from the language?

A. No, of course not.

Q. And they were considered free and available as part of the language?

A. As part of the platform, yes.

Q. Now, you were talking about implementations being separate.

A. Yes.

Q. Can you explain to the jurors what you mean by that? What do you mean by a separate implementation of a program?

A. So just because you've written an application to make a beep – you know, if I write it, you can write it on your note pad right now. It's not going to do anything. You actually

* * *

[2091] better language for beginners and for people that didn't want their programs to crash.

Q. And have you taught courses specifically in the Java language?

A. I have taught many courses that use Java, yes.

Q. And when you taught those – have you taught introductory courses in the Java language?

A. I have taught introductory courses.

Q. Have you ever said anything about APIs in any of your introductory level courses in the Java language?

A. We discuss with all our students that we want to write programs that actually do something. And when programs do something, they need to use APIs.

So we absolutely talk about APIs so that our programs can do something interesting and useful.

Q. And are there any sort of standard reference materials relating to the Java programming language that you're familiar with?

A. There are. There are books on the Java programming language. We've seen some of those in court, is my understanding.

Q. We've seen a book in court called the *Java Programming Language Specification*. You're familiar with that?

A. Yes, I am.

MR. BABER: May I approach, your Honor?

* * *

[2128] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOOGLE, INC.,

Defendant.

San Francisco, California
April 27, 2012

Before the Honorable William H. Alsup

TRANSCRIPT OF JURY TRIAL PROCEEDINGS

---

* * *

[2162] Q. Would it be possible for you to write a program or not?

A. I could write a very simple program, yes.

MR. BABER: Your honor, may the professor step down?

THE COURT: Of course.

(Witness steps down)

THE COURT: Do we have a good bold magic marker?

MR. BABER: We have a brand-new black magic marker, and we have a red one.

THE COURT: Okay. You can come up, and remember to keep your voice loud and clear.

BY MR. BABER

Q. Professor, is there a program that you could write that would show how programmers would use an API?

A. There are many simple programs. I think one program that might resonate what people do is a program that would go over the internet, grab a web page and print it, since most people have used a browser to be online.

Using the Java APIs, I can specify a location on the internet, read the contents of that website, and then cause them to be displayed on my screen.

Q. And how many steps would be in that program that you're talking about?

A. Those three steps that I just talked about, specify where to go on the internet – actually, make a connection to that location on the internet, and then read it, and display it on [2163] the screen. Those are the three steps that I'd write in my program to just illustrate how to call the libraries that are provided with android and Java.

Q. Okay. Could you do so?

A. Yes. We've heard already that the components of an API in a class in Java include the package and the class and the method. So if I'm writing a program, I have to – I have to write it in a class. That's what Java language requires, that I write a class. So I'm going to specify the class, and I specify that by writing kind of a class signature. I will say, "public class webReader."

So I have to specify that it's a class, and I have to give it a name. I'm going to implement it, so I'm going to put a curly brace there and a curly brace there because that's where my code is going to be.

And Java requires that all classes be in a package. So I'm going to name my package at the very top. I'm going to write package. And I'm using the name simple, because this is a simple example. So I'm in the class webReader, in the package Simple, which means using – well, we've heard this phrase, Java.lang. This would be simple.webReader. That's where my code is going to be.

(Witness writing on demonstrative.)

A. And now I need to write a method –

Q. Can I ask my question first, Professor Astrachan. Just so [2164] we're clear, even though it has a name of simple.webReader, is what you're about to write, does that become a part of the API?

A. No, this would not be part of the API. This is my own code I'm going to call into the API. This is not part of the API. This is –

I'm a client programmer. I might have been hired to write a program to go online. So this is not part of the API. I'm going to use the power of the API in that library to read a web page and display it, but this is not part of the API. This package simple is not one of the 37 packages that we have heard about at all.

Q. Please continue.

A. Because I'm a careful programmer, I'd like to make this my work. I'll put a comment in it so that we'll know, and anybody using this code will know, that it's me. I will put a stylized comment indicating that this is my work. I will say "@author." And I'll just

write my "OLA," that's my abbreviation for my name so that I don't have to spend time writing it.

This is the comment, and it indicates that I'm the author of this class and this method. And that way people that see this code will know that's me. That's not part of the code. That's just a comment for people reading this to know that I've wrote it.

Now, I need to make – this is a program, and I'm sure you've heard of computer programs before. And it turns [2165] out that in Java to make a program, as opposed to a library of classes a program runs, the library is code that's called when a program runs. And in Java you have to have a special name of the method that's executed when a program runs.

So I would write that. And I know, because I've written a lot of Java programs, what it – what this method signature looks like. It's the method signature that every program needs. APIs don't need this method signature because they are not programs. They are code that I call.

So I just know I'm going to have to write "public static void main, parenthesis, string, bracket, bracket, r, parenthesis, curly brace, curly brace."

That's kind of a mouthful, but once you have written a lot of programs, you just kind of remember that. And sometimes it might be typed automatically if you've got lucky by the programming environment you're in.

My programming environment is this easel, so I'm writing it out by hand, "public static void main." This is the method signature for my class so that this code will run on my computer. It allows Java to run this program.

And I still haven't called the API to do the three things I said it was going to do, which is specify the location on the internet, open a connection to that location, and then print the contents of that we object page.

So now I'm going to get ready to do that.

[2166] Q. What would come next in your code?

A. I need to specify the location on the internet. That's called a URL. I know, because I have some experience, that network stuff in Java is in the Java.net package. So I know what I need is Java.net.url. That's the class that I'm going to use, Java.net.url.

I'm going to specify a website. And the way you create a new location on the internet is to say "new Java.net.url." And I have to tell it where to go on the internet. I'm going to go to cnn.com, so I write "http://cnn.com."

So I've specified the location on the internet. I made a URL. That's what it's called when you use a web browser. And I used the Java.net.url class. I created a new Java.net.url object. I gave it a name "site." That's a website. And now I'm going to use that.

So I've done one step, specify a location on the internet by using the Java.net.url class in the Java.net package, and I know that that class exists because I've written code before.

Q. What would be the second step?

A. I need to open a location to that website to be able to read it. And since I'm going to read it, I know – again, because I'm experienced – that the location on the internet, I need a connection to it, and it's going to stream information [2167] to me. So I know what I need is a Java.io.inputStream, and that's the source of my information. So the class is inputStream in the

package Java.io. And I get that by saying "open-Stream."

So I've called a method in the URL class. It says "site.openstream." OpenStream is the method. I'm going to go over the internet, make a connection to cnn.com, and I've got that connection, and now I'm ready to read it and print it on my screen.

Q. Is there another step for that?

A. One more step to print it on the screen: system.out.print. I'm going to take the source, and I'm going to read it. So I call the method read, which is in the Java.io.inputStream. It allows me to read the stream.

This is a lot of steps. It's specify location, make the connection to it, read it, and print it on my screen. Those are the three steps that I need to specify location, make the connection to it, and cause it to, the contents of that website to be displayed on the screen.

I took one small liberty. This actually prints just one character from the website. It would be a little more complicated to read all the characters from the website. That would be just one other method, but this illustrates how to call the libraries in the packages that we have been hearing about.

[2168] Q. Professor, you testified that in order to use the prewritten code in the libraries, the programmer has to use the method signature that's defined under the specification exactly.

Do you recall that testimony?

A. Yes. I talked about that.

Q. In this program you've just written– first of all, is this program now complete?

A. That program is complete.

Q. In this program, these three lines of code, did you, in fact, include any method signatures that would call for something, prewritten code, in any of the libraries?

A. Yes. I used essentially four different method signatures, four.

Q. Can you, using a red marker, just identify the method signatures that you used to invoke the code in the libraries?

A. I called the constructor for the URL class – we heard some testimony, I think from dr. Bloch, that constructors are like methods. They are pretty similar. Sometimes people talk about them as different, but they are almost the same.

So I called the URL constructor right there. That's code in the Java libraries.

I called the openStream method. That's a method that's in the URL class. And I know that I don't send it anything, and it returns a stream, an input stream. I have to [2169] understand what to pass to it, which is nothing, and what I get back, which is an input stream.

Then I called the read method, and it gives me back a character from that stream.

And I called the print method to print it on my screen.

So those are the four methods I called. All of those are in the API packages that we have heard about.

Q. Are they in the same package or different packages that you just – if I – quickly, how do you know what package –

A. Java.net.url, that's the class in the package. Java.net is the package, URL is the class.

site.openStream, that's in the URL class. The method is openStream.

Read is in the inputStream class of Java.io.

And print is in the outputStream class of system.out, which is actually in Java.io, also.

So I have Java.url, Java.net.url, Java.net.url, Java.io, Java.io.

Q. How many packages have you called on, in just writing those three lines of code?

A. I called on Java.lang, which I left out. Everything uses Java.lang. So Java.lang, Java.net, and Java.io. Three packages.

Q. And now are these actual, real signatures, what you've [2170] written on the board, or were they just for illustrative purposes, examples?

A. These are exactly the method signatures that you would need to call.

Q. And how do you know that those are exactly the method signatures?

A. I know these are the method signatures because I have had some experience writing programs, and that's how I know that these are the right method signatures.

Q. Have you memorized these method signatures?

A. I haven't sat down with a piece of paper to memorize them. Because I have written code for so long, I know the ones I use often. They are in my head.

Q. And now when you wrote out those method signatures exactly as they appear in the specifications, did you copy them from the specifications?

A. No. I just used these method signatures. I used them here so that I could write my code. I'm using these signatures. I need to use them to call the libraries. As a programmer you rely on those libraries. I use them to call the library code.

Q. And so would or would not this program actually run if you put it on a computer?

A. This program would run. I left out an exception, but it would run.

[2171] Q. And what would it show on the screen?

A. It would show the first character from cnn.com on my screen.

Q. In order to actually have it show pictures and other content from cnn.com, tell the jury briefly what else you would need to add to this program, if anything?

A. If I wanted to show pictures – right now all this does is read a stream of information. If I wanted to show pictures, I would have to call a library that displayed pictures. This just prints words, characters.

To make a picture appear, I would need a different library in a different package to cause pictures to be displayed on a screen.

Q. To do that, would you need to use additional API methods that have been implemented in API?

A. I would use a library that allowed that, other classes in other packages and other methods to cause those pictures to be displayed. This would not do that.

Q. And would this program run on the Android platform?

A. This program would run on the Android platform, yes.

Q. Would this program run on a computer that had the Java platform on it?

A. Yes, it runs on a Java platform as well.

Q. Professor, do you have an opinion whether, from a Computer Science perspective, Android and Java would be compatible with [2172] respect to these three methods you just invoked in your program?

A. Yes. Since this runs on both the android platform and the Java platform, that's my definition for what it means to be compatible, that the same code runs on both platforms.

Q. Do you have an opinion, professor, whether, from a Computer Science perspective, Android and Java are compatible with respect to the methods and other constructors and other items in the classes of the 37 accused packages?

A. Yes. For those 37 packages, the code that I write on one platform will run on the other platform.

Q. Now, you've testified you used three or four AP – you invoked three or four methods from the API in writing this program; is that right?

A. Yes, that's correct.

Q. Approximately, can you tell us how many lines of code writing did you save by invoking those four methods rather than just writing a completely new program that didn't use any prewritten code from the libraries?

A. By writing these, I've saved probably, not even probably, absolutely a thousand lines of code. For me to write this out all by myself without using those libraries, it would be a thousand lines of code.

Q. All right. Thank you.

THE COURT: I've got some questions before he leaves.

* * *

[2183] than the implementation code in Java.

Q. Have you formed an opinion, Professor, regarding what, if anything, accounts for the fact that the 37 packages in both platforms have the same structure, organization, and use the same names?

A. Those same names that we have in android and in Java are needed so that the code inter-operates, so that code I write can be reused in another situation. So for the functionality of using those APIs, the method signatures need to be the same so that the code will inter-operate and meet programmer expectations.

Q. Does use of the same structure and organization for the packages in android and the same names, does it or does it not serve any functional purpose in Android?

A. The language specification says I must use package, class and method names. And the functionality that those complete signatures provide is what allows me to use the libraries on both – use the code I write, like that code up there, on both platforms. Because I'm using those method signatures, my code will function the same on both platforms.

Q. All right. I'm going to ask you now about a second comparison, professor. I would like you now to

compare what we see on the third line of the chart, which is, I want to take the APIs in both platforms.

So in the Java platform they are all Java packages;

* * *

[2202] THE WITNESS: the functionality provided by those packages is necessary.

THE COURT: All right. Thank you.

Go ahead.

BY MR. BABER

Q. Professor, do you have an opinion regarding whether or not having these 37 packages in android is or is not something that's required to meet the expectations of programmers who are writing in the Java language?

A. I think it's required to meet expectations of Java programmers.

Q. And do you have an opinion regarding whether having these 37 packages in android is or is not required by expectations of industry, of people who use the programming language?

MR. JACOBS: Lacks foundation, your honor. He has no idea what industry requires.

THE COURT: Are you qualified to answer that Question?

THE WITNESS: I believe I am qualified to answer that Question.

THE COURT: Why would you be qualified?

THE WITNESS: Because my students want jobs in industry, and industry comes and tells me what the

characteristics my students need to go get jobs in industry are.

* * *

[2291] Q. How about Java, Dr. Mitchell. Let's talk about writing in the Java language. When a programmer is writing a new program in the Java language, he or she expects to have available APIs that will perform all the functions that are in these 37 packages; isn't that right?

A. I think if you said write something in Java, that might be the default assumption, but if you explain more about the context, someone would happily –

Q. And, in fact –

MR. JACOBS: can we let the witness finish his Answer.

THE COURT: Yes. Please let the witness finish.

Had you finished your answer?

THE WITNESS: I have now, yes.

MR. BABER: I apologize, Dr. Mitchell. Just trying to watch the clock.

BY MR. BABER

Q. If, instead of using the specifications for the packages as they are in the Java platform, and as programmers know them, if instead you had written completely new APIs, would programmers be able to access these functionalities using the names that they have memorized and have used for years?

A. The new APIs use different names, then the old names would not work.

Q. So it's true, is it not, Dr. Mitchell, that if you wrote [2292] completely new APIs, experienced programmers who wanted to access well-known functions that are contained in these 37 packages would have to learn completely new names and wouldn't be able to use what they have been using for years; isn't that right?

A. They would have to adapt to the new API. And whatever the new API gave them, that would be the programming context.

Q. Okay. Now, you talked about Dr. Astrachan's program down there that he wrote. And do you recall his testimony about the entry point for the platform; that there might be a slight difference between the Java platform and the android platform, as to how you first communicate with a program that someone has written?

A. Yes, I do.

Q. And you agree with what he said about that?

A. I believe so, yes.

Q. So is there anything you would do to Dr. Astrachan's program? If somebody said, "we need this to be executable, compilable on android," would you do anything to it, other than change that word "main," to use the appropriate protocol entry point for Android?

A. I think the source code is fine, module of that change. There are other steps you would follow with that code that are different.

Q. So as to the – as to the three lines of code that

* * *

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA

No. C 10-03561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

v.

GOOGLE INC.,

Defendant.

FINAL CHARGE TO THE JURY (PHASE ONE) AND SPECIAL VERDICT FORM

1.

Members of the jury, it is now time for me to give you the final instructions, including instructions on the law that governs this case. A copy of these instructions will be available in the jury room for you to consult as necessary.

It is your duty to find the facts from all the evidence and to decide whether the side with the burden of proof has carried that burden, applying the elements of proof required by the law, elements I will provide you in a moment. In following my instructions, you must follow all of them and not single out some and ignore others. You must not read into these instructions or into anything the Court may have said or done as suggesting what verdict you should return — that is a matter entirely up to you.

2.

The evidence from which you are to decide what the facts are consists of:

1. The sworn testimony of witnesses, whether presented in person or by depositions;
2. The exhibits received into evidence; and
3. Any stipulated facts or facts I told you were deemed to be evidence.

3.

Certain things, however, are not evidence, and you may not consider them in deciding what the facts are. I will list them for you:

1. Arguments, statements and objections by lawyers are not evidence. The lawyers are not witnesses. What they have said in their opening statements, closing arguments and at other times is intended to help you interpret the evidence, but it is not evidence itself. If the facts as you remember them differ from the way the lawyers have stated them, your memory of them controls.
2. A suggestion in a question by counsel or the Court is not evidence unless it is adopted by the answer. A question by itself is not evidence. Consider it only to the extent it is adopted by the answer.
3. Testimony or exhibits that have been excluded or stricken, or that you have been instructed to disregard, are not evidence and must not be considered. In addition, some testimony and exhibits have been received only for a limited

purpose; where I have given a limiting instruction, you must follow it.

4. Anything you may have seen or heard when the Court was not in session is not evidence.

4.

Evidence may be direct or circumstantial. Direct evidence is direct proof of a fact, such as testimony by a witness about what that witness personally saw or heard, or did. Circumstantial evidence is proof of one or more facts from which you could find another fact. By way of example, if you wake up in the morning and see that the sidewalk is wet, you may find from that fact that it rained during the night. However, other evidence, such as a turned-on garden hose, may explain the presence of water on the sidewalk. Therefore, before you decide that a fact has been proved by circumstantial evidence, you must consider all the evidence in the light of reason, experience and common sense. You should consider both kinds of evidence. The law makes no distinction between the weight to be given to either direct or circumstantial evidence. It is for you to decide how much weight to give to any evidence.

5.

In deciding the facts in this case, you may have to decide which testimony to believe and which testimony not to believe. You may believe everything a witness says, or part of it or none of it. In considering the testimony of any witness, you may take into account:

1. The opportunity and ability of the witness to see or hear or know the things testified to;

2. The witness' memory;

3. The witness' manner while testifying;

4. The witness' interest in the outcome of the case and any bias or prejudice;
5. Whether other evidence contradicted the witness' testimony;
6. The reasonableness of the witness' testimony in light of all the evidence; and
7. Any other factors that bear on believability.

6.

You are not required to decide any issue according to the testimony of a number of witnesses, which does not convince you, as against the testimony of a smaller number or other evidence, which is more convincing to you. The testimony of one witness worthy of belief is sufficient to prove any fact. This does not mean that you are free to disregard the testimony of any witness merely from caprice or prejudice, or from a desire to favor either side. It does mean that you must not decide anything by simply counting the number of witnesses who have testified on the opposing sides. The test is not the number of witnesses but the convincing force of the evidence. You should base your decision on all of the evidence regardless of which party presented it.

7.

A witness may be discredited or impeached by contradictory evidence or by evidence that, at some other time, the witness has said or done something or has failed to say or do something that is inconsistent with the witness' present testimony. If you believe any witness has been impeached and thus discredited, you may give the testimony of that witness such credibility, if any, you think it deserves.

8.

Discrepancies in a witness' testimony or between a witness' testimony and that of other witnesses do not necessarily mean that such witness should be discredited. Inability to recall and innocent misrecollection are common. Two persons witnessing an incident or a transaction sometimes will see or hear it differently. Whether a discrepancy pertains to an important matter or only to something trivial should be considered by you.

However, a witness willfully false in one part of his or her testimony is to be distrusted in others. You may reject the entire testimony of a witness who willfully has testified falsely on a material point, unless, from all the evidence, you believe that the probability of truth favors his or her testimony in other particulars.

9.

In determining what inferences to draw from evidence you may consider, among other things, a party's failure to explain or deny such evidence.

10.

Certain charts and summaries have been received into evidence. Charts and summaries are only as good as the underlying supporting testimony or material. You should, therefore, give them only such weight as you think the underlying material deserves.

11.

Now I will address the burden of proof. In this case, the preponderance of the evidence standard applies on all sides, so whoever has the burden of proof on an issue must carry that issue by a preponderance of the evidence. When a party has the burden of proof on any claim by a preponderance of the evidence, it means you must be persuaded by the evidence that the claim is

more probably true than not true. To put it differently, if you were to put the evidence favoring a plaintiff and the evidence favoring a defendant on opposite sides of a scale, the party with the burden of proof on the issue would have to make the scale tip somewhat toward its side. If the party fails to meet this burden, then the party with the burden of proof loses on the issue. Preponderance of the evidence basically means "more likely than not."

12.

On any claim, if you find that plaintiff carried its burden of proof as to each element of a particular claim, your verdict should be for plaintiff on that claim. If you find that plaintiff did not carry its burden of proof as to each element, you must find against plaintiff on that claim. This same principle also applies to defendants on claims or defenses for which it has the burden of proof.

13.

I will now turn to the law that applies to this case. Oracle seeks relief against Google for alleged copyright infringement. Google denies infringing any such copyrighted material and asserts that any use by it of copyrighted material was protected, among other things, by a defense called "fair use," which will be explained below. If you find liability in this phase, we will consider the extent of damages in the third phase of the trial. Now, I will give you an overview of copyright law in general. Then I will give you a summary of the claims and defenses at issue in this case. After that I will give you a further statement of copyright law to help you in resolving the claims and defenses.

14.

By federal statute, copyright includes exclusive rights to copy a work, rights that lasts for 95 years from the date of publication. The rights include the exclusive rights to:

1. Make additional copies or otherwise reproduce the copyrighted work or to license others to do so;
2. Recast, transform, or adapt the work, that is, prepare derivative works based upon the copyrighted work;
3. Distribute copies of the copyrighted work to the public by sale; and
4. Display publicly a copyrighted work.

It is the owner of a copyright who may exercise these exclusive rights to copy. Even though someone may acquire a copy of the copyrighted work, such as a book from a bookstore, for example, the copyright owner retains rights to control the making of copies of the work.

15.

Copyright automatically exists in a work the moment it is fixed in any tangible medium of expression, such as putting pen to paper. The owner of the copyright may then register the copyright by delivering to the Copyright Office of the Library of Congress a copy of the copyrighted work and applying via a registration form, after which the Copyright Office will either allow or disallow the application. By way of examples, copyrighted works can include

1. Literary works like books, periodicals and, of particular interest here, operating manuals;
2. Musical works;

3. Photographs and drawings;
4. Motion pictures;
5. Computer programs, also of particular interest here.

Only that part of the work comprised of original works of authorship fixed in any tangible medium of expression from which it can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device, can be protected by copyright. To take examples, words can be fixed on paper, and a computer program can be fixed in the memory of a mobile phone.

16.

As stated, the owner of a copyright has the exclusive right to make copies of all or more than a de minimis part of the copyrighted work, subject only to the right of anyone to make fair use of all or a part of any copyrighted material, all as will be explained below.

17.

The copyright confers ownership over the particular expression of ideas in a work but it never confers ownership over ideas themselves. For example, if a book describes a strategy for playing a card game, the copyright prevents anyone (but the owner) from duplicating the book itself but everyone is still free to read the book and to use the strategy, for the idea set forth in the book, that is the strategy, is not protected by copyright. And, everyone is entitled to write their own book about the same game and the same strategy so long as they do not plagiarize the earlier book. Again, the main point is that the copyright protects the particular expression composed by the author.

Another statutory limitation on the scope of a copyright is that copyright never protects any procedure, process, system, method of operation, concept, principle, or discovery. Possibly such things can be claimed under the patent system or by trade secret laws but they may not be claimed by copyright. For purposes of your deliberations, however, I instruct you that the copyrights in question do cover the structure, sequence and organization of the compilable code.

18.

I will now turn to the claims in this case. Oracle claims Google has infringed its copyrights in two registered works, namely, "Java 2 Standard Edition, Version 1.4" (TX 464) and "Java 2 Standard Edition, Version 5.0" (TX 475), and the applications leading to those registrations appear at TX 3529 and 3530. Among other things, the registered copyrights generally include the compilable code and documentation for the Java API packages. The main issues you must decide concern these two general types of material contained therein, namely "compilable code" and "documentation." As used in these instructions and the Special Verdict Form, the term API "compilable code" refers to method names and class names, declarations, definitions, parameters, organization, and implementation (whether in the form of source code or object code) implementing the various API functions. The "compilable code" does not include the English-language comments you have heard about. Even though such comments are embedded in the software program, these English-language comments do not get compiled and are not used by the computer to perform API functions. Instead, the English-language comments are part of what I will call the API "documentation," sometimes referred to as the "specification," a term that encompasses all of

the English-language comments. The term "API documentation" includes all content — including English-language comments as well as method names and class names, declarations, definitions, parameters, and organization — in the reference document for programmers. Again, please remember that although these English-language comments appear in the software program listing, they can be extracted for handy reference in the guides made available to programmers. So, I will be referring to the "API compilable code" and to the "API documentation."

19.

The copyrighted Java platform has more than 37 API packages and so does the accused Android platform. As for the 37 API packages that overlap, Google agrees that it uses the same names and declarations but contends that its line-by-line implementations are different (with the exception of the rangeCheck lines), a contention not disputed by Oracle. Instead, Oracle contends that Google copied the structure, sequence and organization of the compilable code for the 37 API packages as a group. Google agrees that the structure, sequence and organization of the 37 accused API packages in Android is substantially the same as the structure, sequence and organization of the corresponding 37 API packages in Java. Google states, however, that the elements it has used are not infringing and, in any event, its use was protected by a statutory rule permitting anyone to make "fair use" of copyrighted works.

20.

Now, let me tell you the law about names. The copyrights do not cover the names, such as those given to files, packages, classes, and methods, because under

the law, names cannot be copyrighted. This applies to the name lava" as well. Although "Java" has been registered as a trademark, there is no trademark claim in this lawsuit. The name java cannot be copyrighted, nor can any other name, whether one or two words or longer in length. While individual names are not protectable on a standalone basis, names must necessarily be used as part of the structure, sequence, and organization and are to that extent protectable by copyright.

21.

With respect to the API documentation, Oracle contends Google copied the English-language comments in the registered copyrighted work and moved them over to the documentation for the 37 API packages in Android. Google agrees that there are similarities in the wording but, pointing to differences as well, denies that its documentation is a copy. Google further asserts that the similarities are largely the result of the fact that each API carries out the same functions in both systems. Google again asserts the statutory defense of fair use.

22.

The issues just discussed center on the API packages. Apart from the API issues, I will now describe a list of specific items that Oracle contends were copied verbatim by Google. Specifically, Oracle contends that Google copied verbatim certain lines of compilable code, namely the rangeCheck method in two files, other source code as compiled into object code in seven "Impl.Java" files and one other file and, finally, certain English-language comments in two other files. Google responds that any verbatim copying by it was excusable under the law as "de minimis." For purposes of

this group of infringement contentions, the structure, sequence and organization is irrelevant and the comparison must be made to the work as a whole as defined in a moment.

23.

Now, I will turn to the more detailed law. In order to prove infringement, Oracle must first prove that Oracle's work is original and that it is the owner of the part of the work allegedly copied. For your purposes, the parties agree that there are no issues of ownership or originality for you to decide.

24.

Oracle must also prove that Google copied all or a protected part of a copyrighted work owned by Oracle and that the amount of copying was more than de minimis. So, there are two elements Oracle must prove to carry its burden on infringement, namely copying of a protected part and that the part copied was more than de minimis when compared to the work as a whole. These are issues for you to decide.

There are two ways to prove copying. One is by proof of direct copying, as where the copyrighted work itself is used to duplicate or restate the same words and symbols on a fresh page.

The second way is via circumstantial evidence by showing the accused had access to the copyrighted passages in question and that there are substantial similarities or, in certain instances, virtual identity between the copyrighted work and the accused work. The virtual identity test is used when the subject under consideration is a narrow one and we would expect certain terms and phrases to be used. This is in contrast to, for example, a fictional work in which

there will be a broad range of creativity, in which case it is necessary only to prove substantial similarity. In this trial, you should use the substantial similarity test for all such comparisons except for those involving the API documentation, in which case you should use the virtual identity test. This is because the documentation for the API packages describe narrow technical functions and it is to be expected that some of the same words and phrases would likely be used.

25.

To determine whether the copyrighted work and the accused work are substantially similar, or where appropriate, virtually identical, you must compare the works as a whole. I will define the works as a whole in a moment.

However, in comparing the works as a whole, you cannot consider similarities to the unprotectable elements of Oracle's works. I have instructed you about the protectable and unprotectable elements of Oracle's work.

26.

Now, I will explain the law governing Google's defense based on the statutory right of anyone to make "fair use" of copyrighted works. Anyone may use any copyrighted work in a reasonable way under the circumstances without the consent of the copyright owner if it would advance the public interest. Such use of a copyrighted work is called a "fair use." The owner of a copyright cannot prevent others from making a fair use of the owner's copyrighted work. For example, fair use may include use for criticism, comment, news reporting, teaching (including multiple copies for classroom use), scholarship, or research.

Google has the burden of proving this defense by a preponderance of the evidence.

In determining whether the use made of the work was fair, you should consider the following factors:

1. The purpose and character of the use, including whether such use is of a commercial nature, for nonprofit educational purposes, and whether such work is transformative (meaning whether Google's use added something new, with a further purpose or different character, altering the copied work with new expression, meaning, or message). Commercial use cuts against fair use while transformative use supports fair use;

2. The nature of the copyrighted work, including whether the work is creative (which cuts against fair use), functional (which supports fair use), or factual (which also supports fair use);

3. The amount and substantiality of the portion used in relation to the copyrighted work as a whole. The greater the quantity and quality of the work taken, the less that fair use applies; and

4. The effect of the use upon the potential market for or value of the copyrighted work. Impairment of the copyrighted work cuts against fair use.

All the factors should be weighed together to decide whether Google's use was fair use or not. It is up to you to decide how much weight to give each factor but you must consider all factors. If you find that Google proved by a preponderance of the evidence that Google made a fair use of Oracle's work, your verdict should

be for Google on that question in the Special Verdict Form.

27.

With respect to the infringement issues concerning the rangeCheck and other similar files, Google agrees that the accused lines of code and comments came from the copyrighted material but contends that the amounts involved were so negligible as to be de minimis and thus should be excused. To be clear with respect to a different issue. The parties are in agreement that the structure, sequence, and organization of the API packages is more than de minimis.

28.

Copying that is considered "de minimis" is not infringing. Copying is "de minimis" only if it is so meager and fragmentary that compared to the work as a whole the average audience would not recognize the appropriation. You must consider the qualitative and quantitative significance of the copied portions in relation to the work as a whole. The burden is on Oracle to prove that the copied material was more than de minimis.

The relevant comparison is the copied portion contrasted to the work as a whole, as drawn from the copyrighted work, not contrasted to the accused infringer's work as a whole. For example, if an infringing excerpt is copied from a book, it is not excused from infringement merely because the infringer includes the excerpt in a much larger work of its own.

29.

In your deliberations, you will need to make certain comparisons to the "work as a whole." It is my job to isolate and identify for you the "work as a whole." You

must take my identification as controlling if and when this comes up in your deliberations. This issue arises when (1) comparing Oracle's work and Google's work for similarity under both substantial similarity and virtual identity standards, (2) deciding whether Google copied only a de minimis amount of Oracle's work, and (3) evaluating the third factor of fair use: the amount and substantiality of the portion used in relation to the copyrighted work as a whole.

Although you have seen that the copyright registrations cover a large volume of work, the entire registered work is not the work as a whole for these purposes. This may seem odd to you, so let me give an example. An entire magazine issue may be copyrighted but a specific article or advertisement or photograph may be the relevant work as a whole, depending on what was allegedly copied.

For purposes of this case, I have determined that the "work as a whole" means the following: For purposes of Question No. 1 in the Special Verdict Form, the "work as a whole" constitutes all of the compilable code associated with all of the 166 API packages (not just the 37) in the registered work. This excludes the virtual machine. Similarly, for the purposes of Question No. 2 in the Special Verdict Form, the "work as a whole" means the contents (including names, declarations and English-language comments) of the documentation for all of the 166 API packages (not just the 37) in the registered work. For purposes of Question No. 3, the "work as a whole" is the compilable code for the individual file except for the last two files listed in Question No. 3, in which case the "work as a whole" is the compilable code and all the English-language comments in the same file.

30.

Unless you find fair use, de minimis, or non-infringement in Google's favor, Google had no right to copy any elements of the Java platform protected by copyright unless it had a written license to do so from Sun or Oracle or had a written sub-license to do so from a third party who had a license from Sun or Oracle conferring the right to grant such sub-licenses. The burden would be on Google to prove it had any such express license or sublicense rights. But in this trial it makes no such contention. Put differently, if Google claims a license from a third party, Google has the burden to prove that the third party itself had the proper right and authority from Sun or Oracle as to any of the copyrights owned by Sun or Oracle and used by Google, for Google could acquire from the third party no greater right than the third party itself had in the first place. Similarly, if Google contends that Oracle or Sun had dedicated elements protected by copyright to the public domain for free and open use, the burden would be on Google to prove such a public dedication but the parties agree that that issue is for me to decide, not for you as the jury to decide. This statement of the law regarding licenses is simply to put some of the evidence you heard in context.

31.

When you begin your deliberations, you should elect one member of the jury as your foreperson. That person will preside over the deliberations and speak for you here in court. I recommend that you select a foreperson who will be good at leading a fair and balanced discussion of the evidence and the issues.

You will then discuss the case with your fellow jurors to reach agreement if you can do so. Your verdict

as to each claim and each defense, if any, must be unanimous. Each of you must decide the case for yourself, but you should do so only after you have considered all of the evidence, discussed it fully with the other jurors, and listened to the views of your fellow jurors.

Do not be afraid to change your opinion if the discussion persuades you that you should. Do not come to a decision simply because other jurors think it is right. It is important that you attempt to reach a unanimous verdict but, of course, only if each of you can do so after having made your own conscientious decision. Do not change an honest belief about the weight and effect of the evidence simply to reach a verdict.

I will give you a special verdict form to guide your deliberations.

32.

Some of you have taken notes during the trial. Whether or not you took notes, you should rely on your own memory of what was said. Notes are only to assist your memory. You should not be overly influenced by the notes. When you go into the jury room, the Clerk will bring in to you the trial exhibits received into evidence to be available for your deliberations. The Clerk will also provide you with an index to them.

33.

As I noted before the trial began, when you retire to the jury room to deliberate, you will have with you the following things:

1. All of the exhibits received into evidence;

2. An index of the exhibits;

3. A work copy of these jury instructions for each of you;
4. A work copy of the verdict form for each of you; and
5. An official verdict form.

When you recess at the end of a day, please place your work materials in the brown envelope provided and cover up any easels with your work notes so that if my staff needs to go into the jury room, they will not even inadvertently see any of your work in progress.

34.

A United States Marshal will be outside the jury-room door during your deliberations. If it becomes necessary during your deliberations to communicate with me, you may send a note through the marshal, signed by your foreperson or by one or more members of the jury. No member of the jury should ever attempt to communicate with me except by a signed writing, and I will respond to the jury concerning the case only in writing or here in open court. If you send out a question, I will consult with the lawyers before answering it, which may take some time. You may continue your deliberations while waiting for the answer to any question. Remember that you are not to tell anyone — including me — how the jury stands, numerically or otherwise, until after you have reached a unanimous verdict or have been discharged. Do not disclose any vote count in any note to the Court.

35.

Now you are going to begin your deliberations. As mentioned earlier, you must stay until 1:00 P.M. today. If you do not reach a verdict by the end of today, then you will resume your deliberations tomorrow and

thereafter. The Court recommends that you deliberate at least from 8:00 A.M to 4:00 P.M tomorrow and thereafter. Your schedule is up to you. You may, of course, take reasonable lunch breaks.

It is very important that you let the Clerk know in advance what hours you will be deliberating so that the lawyers may be present in the courthouse at any time the jury is deliberating.

36.

You may only deliberate when all of you are together. This means, for instance, that in the mornings before everyone has arrived or when someone steps out of the jury room to go to the restroom, you may not discuss the case. As well, the admonition that you are not to speak to anyone outside the jury room about this case still applies during your deliberation.

37.

After you have reached a unanimous agreement on a verdict, your foreperson will fill in, date and sign the verdict form and advise the Court that you have reached a verdict. The foreperson should hold onto the filled-in verdict form and bring it into the courtroom when the jury returns the verdict. Thank you for your careful attention. The case is now in your hands. You may now retire to the jury room and begin your deliberations.

Dated: April 30, 2012.

/s/ William Alsup
WILLIAM ALSUP
UNITED STATES DISTRICT JUDGE

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA

No. C 10-03561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

v.

GOOGLE INC.,

Defendant.

SPECIAL VERDICT FORM

YOUR ANSWERS MUST BE UNANIMOUS.

1. As to the compilable code for the 37 Java API packages in question taken as a group:

A. Has Oracle proven that Google has infringed the overall structure, sequence and organization of copyrighted works?

Yes _____ No _____

(IF YOU ANSWER "NO" TO QUESTION 1A, THEN SKIP TO QUESTION NO. 2.)

B. Has Google proven that its use of the overall structure, sequence and organization constituted "fair use"?

Yes _____ No _____

2. As to the documentation for the 37 Java API packages in question taken as a group:

A. Has Oracle proven that Google has infringed?

Yes _____ No _____

(IF YOU ANSWER "NO" TO QUESTION 2A, THEN SKIP TO QUESTION NO. 3.)

B. Has Google proven that its use of Oracle's Java documentation constituted "fair use"?

Yes _____ No _____

3. Has Oracle proven that Google's conceded use of the following was infringing, the only issue being whether such use was de minimis:

	Yes (Infringing)	No (Not Infringing)
A. The rangeCheck method in TimSort.java and ComparableTimSort.Java	_____	_____
B. Source code in seven "Impl.java" files and the one "ACL" file	_____	_____
C. The English-language comments in CodeSourceTest.java and CollectionCertStoreParameters Test.java	_____	_____

4. Answer the following special interrogatories only if you answer "yes" to Question 1A.

A. Has Google proven that Sun and/or Oracle engaged in conduct Sun and/or Oracle knew or should have known would reasonably lead Google to believe that it would not need a license to use the structure, sequence, and organization of the copyrighted compilable code?

Yes _____ No _____

B. If so, has Google proven that it in fact reasonably relied on such conduct by Sun and/or Oracle in deciding to use the structure, sequence, and organization of the copyrighted compilable code without obtaining a license?

Yes _____ No _____

Your answers to Questions 4A and 4B will be used by the judge with issues he must decide. Questions 4A and 4B do not bear on the issues you must decide on Questions 1 to 3.

Dated:

_____

FOREPERSON

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA

No. C 10-03561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

v.

GOOGLE INC.,

Defendant.

SPECIAL VERDICT FORM

YOUR ANSWERS MUST BE UNANIMOUS.

1. As to the compilable code for the 37 Java API packages in question taken as a group:

A. Has Oracle proven that Google has infringed the overall structure, sequence and organization of copyrighted works?

Yes ___✓___ No _____

(IF YOU ANSWER "NO" TO QUESTION 1A, THEN SKIP TO QUESTION NO. 2.)

B. Has Google proven that its use of the overall structure, sequence and organization constituted "fair use"?

Yes _____ No _____

2. As to the documentation for the 37 Java API packages in question taken as a group:

A. Has Oracle proven that Google has infringed?

Yes _____ No ___✓___

(IF YOU ANSWER "NO" TO QUESTION 2A, THEN SKIP TO QUESTION NO. 3.)

B. Has Google proven that its use of Oracle's Java documentation constituted "fair use"?

Yes _____ No _____

3. Has Oracle proven that Google's conceded use of the following was infringing, the only issue being whether such use was de minimis:

	Yes (infringing)	No (Not Infringing)
A. The rangeCheck method in TimSort.java and ComparableTimSort.Java	✓	
B. Source code in seven "Impl.java" files and the one "ACL" file		✓
C. The English-language comments in CodeSourceTest.java and CollectionCertStoreParameters Test.java		✓

4. Answer the following special interrogatories only if you answer "yes" to Question 1A.

A. Has Google proven that Sun and/or Oracle engaged in conduct Sun and/or Oracle knew or should have known would reasonably lead Google to believe that it would not need a license to use the structure, sequence, and organization of the copyrighted compilable code?

Yes ___✓___ No _____

B. If so, has Google proven that it in fact reasonably relied on such conduct by Sun and/or Oracle in deciding to use the structure, sequence, and organization of the copyrighted compilable code without obtaining a license?

Yes _____ No ___✓___

Your answers to Questions 4A and 4B will be used by the judge with issues he must decide. Questions 4A and 4B do not bear on the issues you must decide on Questions 1 to 3.

Dated: May 7, 2012

/s/ [Illegible]
FOREPERSON

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

Case No. 3:10-cv-03561 WHA

ORACLE AMERICA, INC.,

Plaintiffs,

v.

GOOGLE INC.,

Defendant.

Trial Date: May 9, 2016
Dept: Courtroom 8, 19th Fl.
Judge: Hon. William Alsup

GOOGLE INC.'S DEPOSITION CLIPS OF HENRIK STAHL PLAYED BY VIDEO DURING TRIAL

Trial Exhibit 7803

* * *

01/14/2016

* * *

Q. Has Oracle ever licensed Java SE for use in a mobile phone?

A. I don't know. Not that I'm aware of.

Q. Are you aware of any efforts by Oracle to do so?

A. Not that I can recall.

* * *

Q. So consumers began to demand smartphones with more features as opposed to older feature phones, correct?

A. Consumers began to demand a more advanced functionality from their phones. That's correct.

Q. And Oracle did not provide a solution that could be used in those more modern feature-rich phones, correct?

A. We did not provide – actually, we never provided a complete software stack for any phone. We licensed Java for phones.

Now, when lower-end phones were more prevalent, Java ME was a great solution and a lot of vendors licensed Java from us.

When the hardware became more capable, the OEMs moved to Android, because it provided a Java-like environment. Why would they come and buy something from Oracle.

You could have taken Oracle's Java technology and used it in an Android phone, right? There are no technical issues with that. It just wasn't done.

Q. And as people moved towards more advanced phones with more capabilities, they moved away from feature phones, correct?

A. They moved away from low-end phones where Java ME was the best solution to run Java applications, which is, I guess, what we mean when we say feature phones here.

Q. The second bullet point in the document on page 30 says, "Old technology stack."

A. Uh-huh.

Q. Was it your view, at the time of this document in April 2012, that Java ME was an old technology stack?

A. Yes, I believe that if you wanted to continue to use and license Java ME and, in particular, to be able to compete with something like Android, would have to make significant investments in it.

Now, that might not be the right decision, which is kind of what the third bullet here is referring to.

Q. And Oracle did not, in fact, make the significant investment required to keep up, correct?

A. We considered it.

A. We considered it. We decided against it.

Q. So there eventually came a point in time where Oracle gave up marketing Java ME for phones of any kind, correct?

A. There came a time when we decided that we would not invest in new versions of Java ME for phones. It's quite possible that

Q. Do you recognize Exhibit 1426, Mr. Stahl?

A. (Perusing.) I don't remember it, but I'm sure I sent this.

Q. In the e-mail in Exhibit 1426, you say, "TCK's concern, on the other hand, was more geared towards dated upper stack APIs, which make Java ME look old and feature poor compared to Android."

A. Uh-huh.

Q. What did you mean by that?

A. So with upper stack APIs, we meant APIs that were specifically built to interact with modern, you know, hardware capabilities provide by in the current generation phones at the time, things like cameras and, you know, the GPS receiver, so on and so forth.

There were Java ME APIs that provided access to such features, but those APIs were fairly dated, and they didn't work on Java SE.

So what we did here was try to analyze, like, which of those APIs could we carry forward, and if we carried them forward, like, where would we start.

Q. What do you mean by carry forward?

A. Take the APIs, modernize them, and make them – make them work, not only on ME, but also on SE, so that you could more easily move between the two platforms.

Q. And why didn't Oracle go very far down that path of modernizing Java ME and Java SE?

A. So that's not a correct statement. We actually did modernize both Java ME and Java SE. The core platforms, as a matter of fact, are significantly modernized, and actually very, very good today, both Java ME 8 and SE 8.

What we didn't do is we didn't produce new versions of all of these APIs that we identified as necessary to have, you know, a complete, modernized phone tablet stack.

And the reason for that was simply once, you know, push came to shove, and we did the resource cost analysis and the revenue analysis, it didn't make sense anymore. We didn't think there would be enough market. So we went and focused on embedded, which didn't need these APIs.

Q. So Oracle chose not to reproduce or modernize the APIs in Java ME or Java SE, correct?

A. For supporting phone-specific hardware, yes, we never produced any product or, like, officially supported updated library with this functionality.

We did spend a fair amount of time on it, and there was a number of R&D projects proving the concept that this would be doable.

Q. But Oracle chose not to actually complete that process?

A. We choose never to productize it, correct.

Q. Did you believe, at the time you wrote this, that Java ME looked old and feature poor compared to Android?

A. I don't remember. I certainly believed that there were, you know, development we had to make to enable the full range of, you know, modern phone capabilities to Java developers, things like an updated camera API, a GPS stack and Bluetooth, and so on and so forth.

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA

No. C 10-03561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

v.

GOOGLE INC.,

Defendant.

ORDER RE 62 CLASSES AND INTERFACES

The Court has reviewed the parties' responses to the order to show cause regarding specific class, interface, and method declarations from 62 classes and interfaces and the structure, sequence, and organization thereof that all agree were technically necessary to copy (Dkt. No. 1765). It is hereby established that copying the declaring code identified by Oracle (TX 5332, Dkt. No. 1794-2) was technically necessary for Google to use the Java language and thus constituted fair use. This is without prejudice to either side's arguments as to fair use of the other elements of the 37 API packages at issue herein, including Google's contention that the declaring code and SSO of additional classes and interfaces in the three core packages identified by the Federal Circuit were also technically necessary to copy. Further, this order does not preclude the parties from offering evidence relating to the declaring code and SSO addressed herein, provided such evidence is consistent with this ruling.

The parties' experts may update their reports to ensure any proffered testimony complies with this ruling, but only for that purpose. Any updated report shall be served by THURSDAY, MAY 11 AT NOON.

IT IS SO ORDERED.

Dated: May 6, 2016.

/s/ William Alsup
WILLIAM ALSUP
UNITED STATES DISTRICT JUDGE

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

Case No. CV 10-03561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

v.

GOOGLE INC.,

Defendant.

Dept. Courtroom 8, 19th Floor
Judge: Hon. William Alsup

JOINT FILING REGARDING AGREED STATEMENT REGARDING COPYRIGHTABILITY (ECF NO. 1788)

---

* * *

In its May 3 Order entitled "Further Mediation With Judge Kim," ECF No. 1788, the Court ordered the parties to meet and confer with Magistrate Judge Kim regarding an agreed statement for the jury regarding what is protected by copyright and what is not, and regarding the additional issues addressed in the Order.

The parties met and conferred with Judge Kim starting at 10:00 a.m. on Saturday, May 7, 2016, and agreed to the following statement:

AGREED STATEMENT

The Java platform is a software application platform that is used to write and to run programs in the Java programming language. The Java programming language is free and available to use. The Java platform includes, among other things, the Java Virtual Machine and the Java API packages. "API" stands for "Application Programming Interface."

What is at issue in this case are the Java API packages, which are sets of prewritten computer programs used to perform common computer functions without a programmer needing to write code from scratch. These prewritten computer programs assist developers in writing applications. These prewritten programs are organized into packages, classes, and methods. An API package is a collection of classes. Each class contains methods and other elements.

The packages, classes and methods are defined by declaring code. The declaring code is the line or lines of source code that introduce, name, and specify the package, class, or method. The declaring code allows programmers to understand and make use of the prewritten programs in the API packages to write their own programs.

The declaring code for the packages, classes and methods reflects the structure, sequence, and organization (or "SSO") for the Java API packages. The SSO specifies the relationships between and among the elements of the Java API packages, and also organizes the classes, methods and other elements in the package.

Each individual method performs a specific function. The declaring code for a method is sometimes referred to as the "method declaration," "header" or "signature." The declaring code for a method tells the programmer

the information the method needs (the inputs) to perform the desired function.

Each method also contains implementing code. The implementing code provides step by-step instructions that tell the computer how to perform the function specified by the declaring code.

The declaring code and the SSO of the 37 Java API packages at issue are protected by copyrights owned by Oracle. The copyright protection does not extend to the idea of organizing functions into packages, classes, and methods, but the copyright protection does cover the SSO as expressed in the 37 Java API packages.

Dated: May 7, 2016

ORRICK, HERRINGTON & SUTCLIFFE LLP

By: /*s*/Vickie Feeman
ANNETTE L. HURST
GABRIEL M. RAMSEY
PETER A. BICKS
LISA T. SIMPSON

Attorneys for Plaintiff
ORACLE AMERICA, INC.

Dated: May 7, 2016

KEKER & VAN NEST LLP

By: /*s*/Bruce W. Baber
ROBERT A. VAN NEST
CHRISTA M. ANDERSON
DANIEL PURCELL

Attorneys for Defendant
GOOGLE INC.

ATTESTATION OF CONCURRENCE

I, Bruce W. Baber, the ECF user whose ID and password are being used to file this Joint Statement Regarding Agreed Revisions To Modified Proposed Jury Instructions On Fair Use (ECF No. 1688), hereby attest that Vickie Feeman, one of the attorneys for plaintiff Oracle America, Inc., concurs in this filing.

Dated: May 7, 2016

By: /s/ Bruce W. Baber
BRUCE W. BABER

[216] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOGGLE, INC.,

Defendant.

San Francisco, California
Tuesday, May 10, 2016

Before the Honorable William H. Alsup

TRANSCRIPT OF PROCEEDINGS

---

* * *

[337] Q. What about the language itself? Did Sun make efforts to promote the language and its use?

A. Well, of course. We wanted everyone to use the language because the more people who would use the language, the more – the bigger the ecosystem that would be built would occur.

Q. How did you go about promoting the language? Was it taught?

A. Yes. But we had big programs – we decided to try to get computer science schools to teach it, so there were textbooks that were developed in universities,

and the theory was that young programmers would emerge with the ability to use this language and do amazing things.

Q. What is JavaOne?

A. JavaOne was a conference that we held also here at Moscone which I was in charge of when I was at Sun, and the idea was to get all of the people in the ecosystem in one place. And we had our – a huge JavaOne conference. All the programmers came. We celebrated the brilliance of our ideas and our accomplishments.

Q. Did JavaOne become an annual event?

A. Yes.

Q. And continues on through the '90s and beyond as well?

A. Yes. It was of immense scale.

Q. Are you familiar with the term API, Application Programming Interface?

* * *

[347] A. No. The work that had been done was simply preparatory. It was exploratory. It was research and development, and the first Android phone did not come out until years after the acquisition.

Q. And were there various options for Google in building Android?

A. Well, we purchased Android.

Q. Right.

A. So we had the choice of not purchasing Android and doing something else. We had tried strategies before involving partnerships with telecommunica-

tions companies, but none of them worked very well. They were just bad products, if I could be blunt.

Q. Okay. And once Android came in, once Android had been acquired by Google, did you still have options as to how to build it, whether to buy it or build it internally?

A. We had many choices once we acquired because we needed to fill out the offering. And at the time, if you wanted to build a smartphone, you had to pay a lot of different fees to a lot of different players.

So typical example is that there is the thing that does audio and video. You had to pay a royalty or fee to use this particular piece of software. So our idea was that we would have an offering, a piece of software, that would pay off all of those royalties. In other words, it would be free software [348] to the licensee who wanted to use it. This is called an open source. And it was revolutionary at the time.

Q. Why was it so revolutionary?

A. Because most people were trying to pay for their implementations by licensing, but we thought we will just allow it to be freely licensed, and we could always make money from our applications.

Q. Your applications being things like Search?

A. I think Search would be a primary example there.

Q. Early on, did you consider partnering with Sun to develop Android?

A. We did.

Q. At that time, did Sun have a Java product for use in mobile phones?

A. At the time – and, again, remember I'm now at Google so these are my friends at Sun, but I'm not at Sun anymore. Google had Android, and we thought it would be a good idea to take what is called the Java Mobile and put it inside of this Android product. We thought that that would be good for everybody. I'm obviously very pro Java, etc.

Q. And did Google expect Android to be different from what existed at that time? By that I mean feature phones?

A. Our view of Android was there was never anything like it and it was completely different from any other approach. There was nothing like – we were building something new from our [349] experience.

Q. What did you see as the benefit to Google as a partnership with Sun?

A. Well, I had an emotional reason why I thought it would be good to work with Sun from my own history, but as a technical matter, I liked the team, I liked the implementation that they had done because I had overseen it, and it would be good to have that quality inside of our phone, in my view.

Q. And what is the technology you were hoping Sun would contribute to the partnership?

A. So if we go back to this business about implementations, there was an implementation that the Java people had done on the Sun side that we – that if we just put it in our phone, it would allow people to write applications and invent the future in interesting ways. We did not understand at the time how it would be used, but we thought it would be useful.

Q. And were you considering using the Java logo and brand, too?

A. Yes. That would be ideal because, again, we liked the logo, we liked the brand.

Q. Would there have been a time advantage to Google if you had been able to work out a partnership with Sun?

A. Well, probably.

MR. BICKS: Objection, Your Honor. Hypothetical.

THE COURT: You may rephrase that. It's improperly

* * *

[361] that you didn't have a partnership with Sun?

MR. BICKS: Objection on relevance, Your Honor.

MR. VAN NEST: Good faith, Your Honor.

THE COURT: No. That's a proper question. It is leading, but please don't lead the witness, but the objection made is overruled.

Please answer.

THE WITNESS: We believed it was permissible to implement the language without a license from Sun.

BY MR. VAN NEST:

Q. And did you have a belief as to whether or not you could use the APIs?

A. Yes.

Q. What was that believe based on?

A. That it was permissible to do so.

Q. What was your belief in that based on?

A. Forty years of experience.

Q. Tell the jury what you mean.

MR. BICKS: Your Honor, we're going on 40 years of experience. I'm going to object depending on where this is going in terms of time frame.

MR. VAN NEST: Let me ask another question, Your Honor.

Q. At the time that Google built Android using the Java APIs, was it your understanding that that was permissible to do as [362] long as you wrote your own implementing code?

A. That is correct.

Q. What was that understanding based on at the time?

A. Well, business advice, legal advice, many years of experience in the industry.

Q. Let's leave the legal advice out.

What was the many years of experience contributing to this belief?

THE COURT: The jury will disregard that comment about legal advice. Because –

MR. VAN NEST: Let's stick – excuse me.

THE COURT: – because I could get into the reasons, but the parties have elected not to get into and tell you what the legal advice was, so we can't make these vague allusions to legal advice. You must disregard the idea that there was legal advice behind what the witness is going to say. I'm not saying the legal advice was bad or good, but you just have to disregard it.

However, the other parts of the question are permissible.

So please answer as to the other parts of the question, the basis.

THE WITNESS: Would you like a historical example?

BY MR. VAN NEST:

Q. Sure.

A. Okay. When I was at Sun in the early '90s, I built a [363] product called WABI which was roughly analogous to what we were doing at Google, and this used the public – public interfaces for Windows to build an implementation of Windows without the license fee to Windows.

Q. That was something you did at Sun?

A. That is correct.

Q. Now, by then, by 2006/2007 at Google, did you have other experience with APIs that informed your decision?

MR. BICKS: Again, Your Honor, objection because this witness is not an expert on custom relating to APIs.

THE COURT: Well, as long as it's his actual – he's explained – as long as this is meant to explain what he actually thought at the time as to his good faith or bad faith, it's permissible. So I will overrule that objection, but it has to be cast in terms of what he actually had in mind at the time and not veer off into expert opinion.

So to that extent, the objection is overruled.

Please answer.

THE WITNESS: So in the industry, I had seen a number of examples over my service of this kind of thing. So that's why I believed what I did.

BY MR. VAN NEST:

Q. Was it a secret that Google was developing Android with Java?

A. Certainly not.

[364] Q. Was Mr. Schwartz told that you would be continuing to use the Java language in the APIs?

MR. BICKS: Again, Your Honor, leading.

THE COURT: It is leading. Sustained.

Please don't lead the witness on something that is important.

BY MR. VAN NEST:

Q. Had you and Mr. Schwartz discussed the nature of Google's effort to build the phone?

A. Yes. Many times.

Q. Okay. And in those discussions, did you have any discussion about either the language or the APIs?

A. I told Mr. Schwartz the details of what we were doing as I knew them at the time, which would have included that information.

Q. Now, do you have any idea how long it took Google to build Android, Mr. Schmidt?

A. If we refer to the timelines, since I got the date wrong last time, we purchased Android in –

Q. 2005?

A. – 2005. And it had been in development for a couple of years before. And we released the first version of Android in 2009 – 2008.

Q. Actually, November 2007 Google releases Android?

A. Yeah. But that one didn't work. We really released the [365] working version in 2008.

Q. Why did it take so long to develop? That's three years. July '05 to '08?

A. There is just a tremendous number of pieces to make the magic happen on these smartphones, and we were under a great deal of pressure because the iPhone had come out earlier.

Q. Okay. We have on our timeline an announcement, Google releases Android. What announcement was made in November of 2007?

A. So we announced something called the Open Handset Alliance, and again, our idea was to have as many partners in the ecosystem that would use this, so our goal was to get as many people on this platform as possible, which, of course, we were freely licensed – licensing. Excuse me.

Q. Was the announcement made on behalf of the entire Alliance?

A. Yes. And I did it.

Q. Can you provide a few examples of members of the Alliance to the jurors?

A. Well, pretty much all of the telecommunications companies and pretty much all of the enterprise software companies.

Q. At the time you announced Android, did you also announce the nature of licensing for Android to the public?

A. Yes. We indicated that Android would be freely licensed.

Q. What do you mean by that?

[366] A. You did not need to pay a fee to use Android.

Q. Now, at this time, November of 2007, was Mr. Schwartz still the CEO of Sun?

A. He was.

Q. And did he have a popular blog?

A. He did.

Q. Was this something you read periodically?

A. I did.

Q. Did you understand whether or not it was an official statement of Sun?

A. I – I assumed it was his view and also the view of the company.

Q. Did Mr. –

MR. BICKS: Objection, Your Honor. It's add-on testimony. It's not responsive.

THE COURT: Well, it – even if he doesn't know for sure, it may go to his state of mind, which is an issue in the case, on propriety of the use under Factor 1, so the objection is overruled.

BY MR. VAN NEST:

Q. Did Mr. Schwartz publish a blog post concerning Android?

A. He did.

Q. Did you read it at the time?

A. I did.

MR. VAN NEST: May I approach the witness, Your Honor.

* * *

[378] Q. Did he express disapproval, in any way, of Android's use of the Java APIs?

A. He did not.

Q. Did he ever tell you, in any of your meetings, discussions or emails, that Google needed a license from Sun to use the Java APIs?

A. He did not.

Q. Based on your discussions with Mr. Schwartz, and his comments to you, did you feel you had a good understanding of what was permissible from Sun's perspective?

A. I did.

Q. And what was that?

A. That our approach was appropriate and permitted.

Q. And what understanding, if any, did you have at the time as to whether or not your use of the APIs in Android was consistent with Sun's business practices?

A. Well, because of my own history, I – I had a long history with this, and I was quite sure this was all permissible.

Q. Why?

A. Because I set up the original deal 20 years earlier, and I participated with them for such a long time.

Q. Okay. And when you say you "set up the original deal," would you tell our jurors what you mean.

A. This was when we did the original Java announcements. These were the terms. So I did them.

[379] THE COURT: It's unclear whether you're talking about Google or Sun.

THE WITNESS: Oh, I apologize.

THE COURT: Please clarify.

THE WITNESS: In 1995, when we announced Java– which I did – these were the terms that I set. So I knew them quite well.

BY MR. VAN NEST

Q. And by these are the terms we set, can you tell the jury what you mean?

A. To summarize, that if you have an implementation, it has to be licensed and you have to pay for it; but that you can use the published interfaces and build your own implementation.

Q. Did you rely on that knowledge in going forward in building Android?

A. I did. Or we did.

MR. VAN NEST: Pass the witness, Your Honor.

THE COURT: All right. Thank you.

CROSS-EXAMINATION

Before you get started, Mr. Bicks. . .

My plan is to push all the way through to 1:00 o'clock. If anyone in the jury box needs a break we, of course, will take one. So if we get to that point, raise your hand and just let me know. Are we okay for now, or do we need to break now?

Great. We will push on. If you need one, raise your hand

* * *

[439] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOGGLE, INC.,

Defendant.

San Francisco, California
Wednesday, May 11, 2016

Before the Honorable William H. Alsup

TRANSCRIPT OF PROCEEDINGS

---

* * *

[495] A. I believe so.

Q. What responsibilities did you have as – in Java product marketing?

A. To pair with the engineering leader at the time to try to help craft the products and the strategy we would use to get those products distributed across the world.

Q. And as you rose up through the company through Chief Operating Officer and Chief Executive, did you continue to have oversight responsibility for Java?

A. Absolutely, yes.

Q. Now, during the time you were employed at Sun, was the Java programming language free and available for anyone to use?

A. Absolutely, yes.

Q. And how long has that been the case?

A. Since its inception. Since long before I arrived at Sun.

Q. Okay. And during your time there, was Sun promoting widespread use of the Java programming language?

A. Absolutely, yes.

Q. How did you go about doing that?

A. Any way we could.

Q. Can you tell the jurors how you went about promoting the language?

A. Yes. We distributed free educational materials. We made sure the technology was broadly available to anyone who wanted to use it.

[496] We visited high schools and colleges universities around the world. We gave money to those universities and students to try to promote their becoming aware of and educated about Java, because it was in our interests to do so.

As we promoted that language and as we promoted that technology, that created – that opened that market that historically we couldn't have gone after. But if you were using Java, then everything else that Sun sold we could sell to you.

If you were using Microsoft Windows, which was at the time the dominating operating system, we had nothing to sell you. So by promoting Java we were creating an alternative to Windows and creating a marketing opportunity for the company.

Q. I'm going to ask you just to slow down a little bit, Mr. Schwartz. We're trying to transcribe every word.

A. My apologies.

Q. You mentioned that one goal was to help you sell the other Java products that Sun had, apart from the language. What products are those or were those?

A. So when you write a program, at some point the program has to run somewhere. So it's going to run on a computer. And we made those computers.

Q. And what is JavaOne?

A. JavaOne is a conference.

THE COURT: No. Wait. Wait. Let's put it in the

* * *

[500] creating those APIs.

Q. Okay. Fair enough.

What was the purpose for having APIs as part of Sun's offerings?

A. They make the underlying technology accessible. They help organize the technology. And they give developers the standardized way of writing applications using those preexisting sets of functionality.

Q. Did Sun promote the Java APIs along with the language?

A. Yes.

Q. How did they go about doing that?

A. They were promoted across the world. We wanted people to understand how to write programs. But then when we wanted them to use preexisting functionality, we delivered our own preexisting functionality. We licensed those APIs. We made them broadly available to anybody else who wanted to create technology.

So it was one in the same. We wanted to promote the availability of the programming language, and then we wanted to make the APIs available as well.

Q. And were the APIs marketed by Sun along with the language; in other words, as free and open?

A. Absolutely, yes.

Q. And can you tell the jurors how that was done?

A. When you are marketing products to technologists across [501] the world, you are not simply saying, here is the book, good luck.

You are saying, here's the download site. Go download not only technologies that might help you show movies and display pictures and manipulate text or do calculations, you are promoting the language and giving a whole set of APIs and preexisting functionality to those developers.

Q. And were the APIs made free and open like the language at that time?

A. Absolutely, yes.

Q. Okay. Were the APIs – during your tenure at Sun, were the Java APIs ever sold or licensed separately from the language?

A. No.

Q. Did you use the term "open APIs" at Sun in the aughts when you were employed there?

(Reporter interrupts.)

Q. The aughts. In the period that you were employed at Sun.

A. Absolutely, yes.

Q. Tell the jury what open APIs meant at that time.

A. So the strategy, which had been the strategy long before I joined Sun, was we agree on APIs, on these open APIs; we share them; and then we compete on implementations.

So we have a way to think about, for example, rendering a movie. And we're going to tell developers how they could go [502] about doing that. But the code you would use to show the movie, you are going to have to write on your own. We have to agree on the APIs so that the application I write to show a movie runs on your device. Then it would still run, but it would run your implementation of how to show a movie.

So you agreed at a high level on what you wanted the application to do. You used a standard set of interfaces. But then when you went to run the application, you would compete on those implementations. My movie rendering application would compete against your movie rendering application.

Q. You're distinguishing or you distinguished between the API and the implementation. Can you explain to the jurors what you meant by implementation.

We've been hearing about implementing code and that sort of thing. What do you mean by implementation?

A. Can I go back to my restaurant analogy? Because I think that might be helpful.

Q. Well, that didn't work with the judge, but go ahead.

(Laughter)

THE COURT: If you think it works, go ahead.

THE WITNESS: I do think it works.

When you walk into a restaurant, there's a menu. And you understand what the different items on the menu are. And one restaurant may offer hamburgers. The hamburgers are the implementation of the item you saw on the menu.

[503] And so when you agree on APIs, you're agreeing on the menus. And then you all have your own implementation. The implementations are the products that you create that are accessed through the APIs.

So if I write a complicated application server for my enterprise, the way it's used is accessed through a set of APIs that are common to all application servers like that. And then we compete by delivering our own application server.

The reason why you agree on APIs and compete on applications is then you can turn around to another developer who wants to use your application server, and you can say, hey, write to my server; it will run here. But because we and all these other companies had agreed on APIs, the application you write can work on their servers as well.

So it's a way of pooling together resources to make sure we can all agree to a common set of instructions. And then the applications can run wherever the implementations are available. But you're not locked into one company's implementation.

I hope that was helpful.

BY MR. VAN NEST

Q. Well, was it important – that last phrase you mentioned, you're not tied to one company or locked into one company, why was that particularly important, if it was, at that time?

MR. BILKS: Your Honor, again, the examples and things [504] like this are really expert testimony.

THE COURT: Well, you have to rephrase it and ask, first, "To what extent, if at all," and stop leading the witness. You're doing a lot of leading. To what extent if at all did you think it was important? That's the proper question.

MR. VAN NEST: There's a good one.

THE COURT: All right.

THE WITNESS: What was important?

BY MR. VAN NEST

Q. Having open APIs, so no one company could control things.

A. Having an open API was very important to us.

As an example, we agreed on how applications could be written for servers running in big businesses. The companies we got together to agree on how those applications could be written had names like Oracle and IBM and SAP. You know, very large companies

who if they felt that what Sun was delivering preferenced Sun, they would never agree to work with us.

So open APIs allowed us all to say, let's agree on this common set of standards. And then we'll go be competitive in the marketplace, but we won't have to change what any one company is doing to give anyone a bias or preference.

So it was a way to try to make things fair. We would make the APIs accessible to anyone who was willing to use them, and then we would compete on the implementations of the products [505] that would be built using those APIs.

Q. During your tenure at Sun, did you use the phrase "reimplementing APIs"?

A. I don't believe I did.

Q. Do you know – well, was the phrase a common one during your tenure at Sun, "reimplementing APIs"?

A. I don't recall.

Q. Let me ask this question: During the time you were at Sun, did Sun ever build its own implementations for APIs produced originally by other companies?

A. Yes.

Q. Can you give us an example?

MR. BICKS: Your Honor, on this testimony now we're really getting beyond the disclosures for this witness, because there's no disclosure about API practices and things of this nature.

MR. VAN NEST: He was broadly disclosed, Your Honor, on practices regarding Java and APIs at Sun.

MR. BICKS: I have it right here.

THE COURT: Can I see what you're talking about?

(Pause)

MR. VAN NEST: Your Honor, I stand by my question.

THE COURT: Show me the part that you think picks up this question.

MR. VAN NEST: Sun's positions and – Sun's positions [506] and communications with regard to independent implementations. Sun's actions regarding copyrights. The history of Java program and the APIs. His representations to the public. Sun's actions or inactions. Industry use of and support –

THE COURT: We're going to pass this until the jury is not here.

You have to go to something else for now.

MR. VAN NEST: That's fine, Your Honor.

BY MR. VAN NEST

Q. Mr. Schwartz, during your tenure at Sun, all the way up to the very end, was there ever a time where the Java APIs were considered proprietary to Sun?

A. No, never.

Q. Did Sun have a trademark for Java?

A. Absolutely, yes.

Q. And what was the Java trademark?

A. The Java trademark was the name as well as the logo.

Q. Okay. And was that licensed to folks for a fee?

A. Absolutely, yes.

Q. Were your implementations, the implementations that Sun wrote, were those also licensed for a fee?

A. It was important that when people went to companies to sell their products, that they could put a logo on the top to say this is Java compatible or this is written to be Java, so that they would understand this is consistent with the Java

* * *

[538] world. Rather than going through people who would interpret what we were doing, we wanted to speak directly.

Q. And did you consider, at the time, the blog to be an official statement of Sun itself?

A. It was an official statement.

Q. What do you mean by that?

A. It was how we announced our quarters. It was how we told the SEC to view our statements we were making, the regulatory agency that oversees companies. So it was very much a formal mechanism for us to communicate with the world about where we were headed.

Q. Now, when Android was announced, did you publish a statement on your official company blog?

A. Yes, I did.

MR. VAN NEST: May I approach the witness, Your Honor?

THE COURT: Yes, you may.

BY MR. VAN NEST

Q. Mr. Schwartz, if you would please identify Trial Exhibit 2352.

MR. VAN NEST: This is in evidence, Your Honor.

BY MR. VAN NEST

Q. What is it?

A. It is a blog from November of 2007.

Q. Okay.

MR. VAN NEST: And could we publish it to the jury, [539] please, first paragraph.

(Document displayed.)

BY MR. VAN NEST

Q. Mr. Schwartz, let's come back to our timeline. Discussions with Google, we're showing, ended in around May of 2006.

Is that consistent, roughly, with your recollection.

A. Yes.

Q. And then in November of 2007, that's the date of your blog post?

A. Yes.

Q. So we're a year and a half later.

The very first paragraph says, "I just wanted to add my voice to the chorus of others from Sun in offering my heartfelt congratulations to Google on the announcement of their new Java/Linux phone platform, Android. Congratulations."

When you said "Java/Linux phone platform," what were you referring to there?

A. The fact they were going to use the Java programming language and build a phone using the Linux operating system.

Q. And there's a reference in the next paragraph.

MR. VAN NEST: Let's highlight that. Can we make it bigger? There we go. Okay.

BY MR. VAN NEST

Q. The last sentence there says – let me read the first one.

[540] "I'd also like Sun to be the first platform software company to commit to a complete developer environment around the platform as we throw Sun's NetBeans developer platform for mobile devices behind the effort. We've obviously done a ton of work to support developers on all Java-based platforms. We're pleased to add Google's Android to the list."

What is NetBeans?

A. NetBeans is a developer environment. It's a software product you would use as a developer to write an application.

Q. And does NetBeans have to be adjusted depending on the platform, or does it work on all platforms?

A. It basically runs on computers and can be used primarily to write Java applications.

Q. And then a little further down, two paragraphs down, you say, "And, needless to say, Google and the Open Handset Alliance. . ." What was the Open Handset Alliance?

A. The group of companies that came together with Google to try to promote Android.

Q. You say that Google and that group "just strapped another set of rockets to the community's momentum."

What did you mean by that? "The community's momentum," what does that mean?

A. So we referred, at the time, to the Java community not to the Java customers. Because there were so many developers who were just a part of the movement we were creating to get people [541] aware of Java, using Java, promoting Java.

So we did our best to invest in the community by making free products available, by making educational materials available. And our view at the time was this was going to give more for the community to take advantage of, create more opportunities for that community.

Q. So at this time, at the time of your blog, can you tell us to what extent you thought Android might help Sun?

A. It was certainly helpful that it wasn't a Microsoft phone. And given the choice between Google embracing Microsoft or Google embracing Java, obviously Google embracing Java was better.

It would have been better yet if they had agreed to take a license from Sun to do so.

MR. VAN NEST: May I approach the witness, Your Honor? THE COURT: Yes.

BY MR. VAN NEST

Q. Mr. Schwartz, take a look at TX 3441.

MR. VAN NEST: This is in evidence, Your Honor.

BY MR. VAN NEST

Q. Do you recognize that email?

A. Yes.

Q. Is it an email exchange between you and Mr. Schmidt on November 9th?

A. Yes.

* * *

[560] evidence TX 7275_1, which is an excerpt from his announcement of open sourcing at JavaOne in 2006.

THE COURT: 7275?

MR. VAN NEST: Underscore 1, yes. It's a video. THE COURT: Any objection?

MR. BICKS: Objection on relevance grounds, Your Honor.

THE COURT: Overruled. Go ahead.

(Government Exhibit 7275_1 received in evidence)

MR. VAN NEST: Could we play 7275_1 for the jury, please.

(Whereupon, the video was played for the jury)

BY MR. VAN NEST:

Q. Is that a somewhat younger version of Jonathan Schwartz in the video?

A. I'm not sure I'd say *somewhat younger*.

Q. Did Sun ever consider, during your tenure there, building a full-stack smartphone platform based on Java?

A. Absolutely, yes.

Q. Do you recall approximately when you first considered doing so?

A. I – from the earliest times surveying other handset manufacturers. We sold technology to Nokia and Ericcson and Sony and other companies.

Q. Was Sun ever able to successfully build a Java-based [561] smartphone platform?

A. We had the foundation technologies to make it work. Had Java FX Mobile, which was the core platform. But we weren't able to get it to market by the time we were sold.

Q. Why not?

A. It's complicated. It's very difficult, as Google can no doubt attest. But, you know, we also had R&D choices we had to make given R&D – Research and Development choices and staffing. Given the economic environment we were operating in, we couldn't fund every project with every dollar we had.

Q. Was Sun's failure to build its own Java smartphone platform attributable in any way to Android?

MR. BICKS: Objection, Your Honor. It's beyond the scope, the disclosure.

THE COURT: All right. Let me see the disclosure. I think I handed it back.

MR. VAN NEST: I have one in a notebook right here, Your Honor.

THE COURT: Can you highlight the language you think covers it? Highlight the language you say covers it so that I can – Mr. Van Nest, can you highlight it or circle it in some way so I can just focus on what you think is the key language.

MR. VAN NEST: Thank you, Your Honor. Thank you, Dawn.

THE COURT: All right. Have you shown counsel?

* * *

[581] Q. Right.

Because at that time, you had commercial relationships with many of the major handset carriers; right?

A. Not for Java SE. That was freely available on computers.

Q. Sir, how many contracts and licenses did you have with people in the handset company, handset world?

A. For a tiny version of Java, we had contracts with all the major handset manufacturers.

Q. And which handset manufacturers?

A. Nokia, Ericcson, Sony, many. I don't recall the number.

Q. And how many phones, mobile phones, at this time was Java in, ballpark?

A. Well, none of them were running SE. None of them were running desktop Java.

Q. Right.

And did you – are you familiar actually with the terms of your licensing to tell us here under oath that you didn't have licenses out to the handset manufacturers for SE?

A. To the best of my knowledge, we didn't. We had licenses for Java ME which was the micro edition, the tiny version of Java.

Q. Are you familiar with the license agreement with Nokia?

A. I don't recall the terms of it, no.

Q. Are you familiar with the license agreement with Danger?

A. No.

* * *

[621] A. Left Danger in 2004.

Q. And why did you leave Danger?

A. I wanted to pursue my next – my next startup company.

You know, I thought Danger – we did a good job of, kind of, creating that category of smartphones – really nailed being a very, very good Internet device.

When we founded the company, we loved the Internet. We loved that you could always have information at your fingertips. But we didn't like being rolled up to our desktop computer to do it.

So we spent a lot of time thinking about the technology necessary to cut that cord and bring the Internet with you. So I hope we did a great job.

But in order for it to be the hundred-million category, you know, the mass-marketed product, I felt it also needed to be a very good phone. Needed to be pocketable. It needed to be small.

So I left to start another company that really focused on bringing that to the mass market.

Q. And what was the other company that you decided to found?

A. That company was Android.

Q. What year did you found Android?

A. Android was founded in 2004, right after I left Danger.

Q. And what was the goal of Android?

A. To create a combination of the best Internet experience [622] and the best phone.

Q. All right. Did you have a business plan, when you founded Android, as to how you would make Android devices available?

A. Again, I mean, this is a matter of scale. How do you – how are you a small startup company, and how can you best leverage your expertise to bring it to as many people as possible. That was the goal of the new company.

So we kind of innovated this model of open source, which is, I didn't need to have a skyscraper full of salespeople selling my product into the wireless industry, which is a global industry. So I didn't have to have people in Korea, China, in the United States.

With open source, I could just basically create the perfect operating system and the perfect smartphone, and let the open source adoption spread it across the globe.

Q. And who would be using this open source software that you are describing?

A. Primarily, it would be the engineers at – at mobile phone manufacturers like Samsung, as well as wireless operators. But it was basically – it was software, right. So it was mostly meant for engineers.

Q. And when you say "open source," what do you mean by that?

A. I mean that everything we created we gave away for free. So there was no charge to get it. And we uploaded it to a file server on the Internet. And anybody could download it in [623] source code form.

Q. When you say you were planning to provide the software for free in an open source fashion, what were you hoping that would accomplish, if anything, in regard to innovation?

A. Well, you know, I was kind of unhappy with the innovation in the cell phone space in that era. It was really, really hard to build a phone. And it was mostly hard because there was a whole ecosystem of software developers, these little companies that somebody like a Motorola would have to aggregate all these different pieces, like pieces of a puzzle. And they would have to build a video player from one company, and an operating system from another company, and the user interface from a third company.

And that would all get kind of put into these feature phones. And the result, it just wasn't a good user experience. It wasn't good for the consumer.

And I was frustrated because I was a consumer. I used these things. And I just wanted it to be much, much better. So our goal was to make that whole vertical stack in one space, where we could control the user experience and make it the best for consumers.

Q. Did you have any business plan, in regard to Android, that Android would provide any way to make any money, given that you were providing the software, the open source software, on a free basis?

* * *

[633] A. My opinion was it wasn't necessary. It was just another one of those accelerants.

Q. Why did you decide to turn to the Java Language as the language that would be available for developers who wanted to write apps for Android?

A. I mean, one of the mechanisms I used to reduce the set of all of the languages that everybody was arguing for is I spent a lot of time thinking about, how do developers learn these languages? You know, what does it take to bring a developer up to speed?

I mean, we could have created our own programming language for this thing. But the work that a developer would have to go through to learn something completely new, I thought, was just out of question. I didn't want to burden a new developer with having to learn something new, because I had a new platform that I was about to release to the market and they had other choices. So I was trying to get the mind share of the developer so that it was frictionless for them to adopt my platform.

And the way I did that is I looked at all the programming language that a developer – that an engineer might be taught in university, right. And that reduced dramatically the number of languages we could have chosen, including building our own because that couldn't have been taught in university. And we ended up picking one that was, you know, taught in university [634] to engineers.

Q. During the course of your time with the Android team at Google, did you have reason to pay attention to what potential competition existed in the market to Android phones?

A. Of course. We spent a lot of time thinking about the landscape and where the market would move in the future.

Q. Okay. Did there come a point in time when you became aware of the iPhone during the period you were with Android?

A. Yeah. I mean, the iPhone was – we were going to be in development a long time when we were working with Android. And then, you know, even before we shipped our first phone, the iPhone launched.

Q. Okay. And did you have an understanding as to what language developers were using to develop apps for iPhone as part of the work you did in studying the market?

A. Sure. I mean, you know, the market – the existing mobile market consisted of a lot of different programming languages to make phones work.

You know, there was Microsoft Windows software that was out there, that used a specific programming language called C and C++. And when the iPhone was announced and the developer kit was made available, they used an derivative of C called Objective C.

Q. So when you're talking about Objective C, is that the same language as Java?

* * *

[639] BY MS. ANDERSON

Q. During your time with the Android team, Mr. Rubin, to what extent did you have an opinion, if at all, as to whether your team was free to use API declarations and organization for the Java APIs as part of the development of the platform?

A. I mean, we didn't think there was any problem with us using the API declarations for the development of Android.

Q. All right. Let's turn, now, to another subject but during the same period of time. Your discussions that you may have had with Sun, all right.

After you joined Google in 2005, did you begin having discussions with Sun as to whether or not they might have a relationship with Google with respect to Android?

A. Yes. I was – we had a lot of discussions with Sun over the years. The initial discussions around them contributing some of the pieces of Android in the make-versus-buy decisions, and ultimately inviting them to become a member of the Open Handset Alliance.

Q. Beginning in 2005, did you have any personal interaction with Sun over these subjects?

A. Yeah, I would frame it as I led those discussions.

Q. All right. And at a high level, could you please explain to the jury what was the general nature of what you were discussing with Sun on behalf of Google in or around 2005, on the subject of Android?

* * *

[658] THE WITNESS: That's when you, kind of, know what the target is. You know what you want to build. You're not going to get help from anybody else. And you just do it in first principles.

You sit down at your blank screen on the computer, and the engineers start writing the code to basically target your destination. Just created from scratch.

THE COURT: All right.

BY MS. ANDERSON

Q. During the time that you were with the Android team at Google, did you have an understanding of what the phrase "implementing code" meant?

A. Yes.

Q. And what did it mean to you?

A. It means allowing a computer scientist to practice their craft, which is write software.

Q. Can you be more specific in distinguishing it from any other part of code?

A. Sorry, can you ask that question again. I want to be sure I understand.

Q. Sure. I would be happy to.

Have you ever heard of the phrase "declarations"?

A. Yes.

Q. And have you heard of the phrase "declarations" used in the context of writing source code?

[659] A. Yes.

Q. All right. And is that concept distinct or the same as the concept of implementing code?

A. I think it's different.

Q. Okay. Could you please explain what your understanding was during the time you were with Android.

A. Sure.

I mean, implementing code is what an engineer does when he wants to make something happen. I think he or she could write code in a thousand different ways. And they have to be very precise in the way they craft this code. It's a little bit of a creative process.

And the majority of the work in making these things work, whether – you know, we talked about a desktop computer and we talked about mobile phones.

I think the magic that happens when an engineer practices their craft is worrying about things like power management, does it run on a battery. You know, unlike your desktop computer that you have to plug into the wall.

So all those, kind of, thousands of decisions they have to make along the way are the things that actually make the function serve its purpose. It does its function by executing/ implementing those decisions.

Q. Thank you.

Let's turn to the time you were at Google when you were

* * *

[672] A. It was a variety, but it was mostly C and C++.

Q. And I see in that green area there, there's a box called "WebKit," which is the far-right column of the three columns there.

Do you see that?

A. Yes.

Q. All right. Did you have an understanding, based on your work with the Android team, as to whether or not that had been provided to Android under an open source license?

A. Yes. WebKit is a – it's the engine that runs browsers. And it was a – it was licensed as open source.

Q. Do you know which open source license it had been offered to Google under?

A. That one was under the LGPL license.

Q. All right. You mentioned, in response to the Court's questions, that in describing the Android runtime section, and specifically the core libraries. Would you tell the jury whether or not that is the part of the platform that contains declarations of Java APIs.

A. Yeah. A small part of that is the declarations themselves.

Q. Okay. And so if you were to be looking at this diagram for where the 37 Java API declarations would be found, would you look to that box or somewhere else?

A. That box. A small part of that box, yes.

[673] Q. All right. And you mentioned there were other things in that box. Could you describe what's also in that core library box?

A. Sure. You know, when – when – when you adopt something, we were trying to fit Android between, kind of, a desktop computer and the mobile phones of that era, which were these flip phones, these feature phones.

So we added a lot of libraries of our own that made phones do more desktop-like things. That functionality wasn't available in the – in the Java implementation.

Q. Okay. And based on your experience working with the Android team and supervising its development, did you have views as to the relative importance of various levels of the Android platform?

A. Yeah. I mean, I had – I had some view into it.

I think, you know, we were taking industry approaches in some places, and then our own approach in other places. And I think I had a pretty good grasp of that.

Q. Did you have a particular view about the application framework layer?

A. Yeah. I mean, I think – again, if you're – if the whole reason you're doing this – the world doesn't need

another operating system. The world needs an open one.

Your customer – you have to think both of the consumer, the person like you and me who would use the phone, and also

* * *

[694] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOGGLE, INC.,

Defendant.

San Francisco, California
Thursday, May 12, 2016

Before the Honorable William H. Alsup

TRANSCRIPT OF PROCEEDINGS

---

* * *

[729] much time and resources in it.

Q. Would you explain to the jury, please, why you thought it was fine to do that?

A. It was just my understanding as a computer scientist about how open source works and how to build systems that were interoperable.

Q. What, if anything, did the concept of independent implementation have to do with your beliefs regarding the use of declarations for Java APIs?

A. Well, the independent implementation or the clean room implementation – I believe software engineering is a creative process. The implementation is where a lot of the creativity happens. We had a lot of computer scientists on staff whose job it was to do this every day. So setting them on that course and asking them to, you know, create these independent implementations, it's what engineers do.

I think that, you know, the clean room implementation, again, I was really transparent with how I managed the team and I asked everybody to beware of external influences that might, you know, change the creativity and the code that you were writing. So I asked people not to, you know – you know, seek out the aid of outsiders and just do it in a clean room, in a closed chamber.

Q. And –

MS. HURST: Your Honor, this is going behind the

* * *

[743] You testified yesterday that Android was announced in November of 2007. Did you observe how the industry responded to that announcement?

A. Yes, of course. I read the press that it generated.

Q. Okay. Please take a look at – excuse me.

Could you characterize the reaction that you observed at the time to the announcement?

A. I would say enthusiastic. You know, some of the– some of the competitors were skeptical, but the majority of the press was positive and enthusiastic.

Q. All right. Did you ever have occasion to see any press released by Sun in response to that release?

A. Yes. I remember.

Q. All right. Would you take a look at Exhibit 2352, which is in evidence.

Do you recognize this exhibit?

MS. HURST: Your Honor, I object to the characterization of this document as a press release by counsel.

THE COURT: All right. It's up to the jury to decide what it is so the jury will disregard the characterization by counsel, but whatever the witness says is okay.

Go ahead.

MS. ANDERSON: Thank you, Your Honor.

Q. What is Exhibit 2352?

[744] A. It looks like a post from the then CEO of Sun's blog.

Q. When did you first read this post?

A. Pretty much the day it came out.

Q. All right. And where did you read it?

A. Online after somebody had forwarded me to the link letting me know that it existed.

Q. All right. And how did you feel about this particular announcement in Exhibit 2352 by Jonathan Schwartz?

A. Similar to their announcement to open source Java. This is more support for what we're doing and I

think a further indication that Sun was kind of thinking along the same lines that we were.

Q. And who did you understand had authored Exhibit 2352 when you read it back at the time?

A. I mean, it was posted to the CEO's blog. It's his personal blog.

Q. And the CEO of what company?

A. Sun.

Q. What was your reaction to reading this blog at the time?

A. I was excited and delighted.

Q. And why is that?

A. Because it was basically putting Sun's support behind our open source mobile operating system. In, you know, no uncertain terms, it was thrilling.

Q. All right. Did you receive any visit from any Sun

* * *

[924] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOGGLE, INC.,

Defendant.

San Francisco, California
Friday, May 13, 2016

Before the Honorable William H. Alsup

TRANSCRIPT OF PROCEEDINGS

---

* * *

[964] what we're talking about.

A. Yeah. So when a programmer sits down to write a Java program, there are certain things that they all have to do, certain just basic tasks. So programs have a lot of, let's say, sorting of lists, searching in lists. You have to write things out to the screen of the computer, write things out to the disk to be stored, and it would be at best tedious and at worst impossible for every programmer to write these low level building blocks themselves so the libraries provide this functionality in nice little packages of code that the programmer can call on, and those are the libraries.

Q. How are the libraries that you worked on at Sun organized?

A. A, it's a three-level hierarchy, so you have packages which consist of multiple classes, and each class consists of multiple methods or functions. You can call them one or the other, but they're the same things, and those are the things you actually call on to do stuff like searching and sorting.

Q. Why did you organize the libraries in that way, in that hierarchy?

A. I actually had no choice in the matter. That's mandated by the language. The language spells that out for you. A computer language is completely inflexible. You – it has a certain set of rules, and you have to obey those rules; and in Java, all libraries are organized in that way. They are packages containing classes containing methods.

* * *

[972] and development?

A. Pretty much all of it. The first API that I actually remember writing was, like, back in the summer of 1983. I worked for IBM Yorktown Heights Research Center as a summer intern, and I wrote an API for a parallel processing IBM 370 that they had there.

Q. For which edition of Java were you developing APIs?

A. It's the one that eventually became known as Java SE for standard edition. Before that it was Java 2 SE, and before that it was just Java, the JDK.

Q. What kind of environment was Java SE being used in during the time you were at Sun?

A. It was being used for desktop computers, servers, you know, powerful laptops, that sort of thing.

THE COURT: Remind us when you were at Sun again.

THE WITNESS: I joined Sun in 1996, and I left Sun in 2004.

THE COURT: Okay. Thank you.

Go ahead.

BY MR. KAMBER:

Q. Dr. Bloch, how, if at all, were the APIs that you developed made public?

A. They are translated into HTML, which you probably know is the basic language of the Web. So when you look at a Web page, you're looking at HTML, and that – excuse me – that HTML was [973] published on the Web, and you can look at it now. Probably not now; but, anyway, it was also – we – we – I wrote books about it and, as I said, I gave lectures about it. So that's how they were published.

Q. Can you explain to the jury why it was that the APIs that you were writing were being made public?

A. Yeah. They were – they were being made public so people could use them. If you build a tool but you don't tell people about it, you know, might as well not have built it. So you have to both build the tool and you have to tell people, "We've got this new tool for you that you can use."

Q. How were you involved in documenting the APIs that you developed?

A. I documented all of my own APIs. I took great care to do my best to document them well.

Q. Why were – excuse me.

Why did you want to provide documentation for the APIs you developed?

A. Again, so they could be used. I mean, if you give people a tool but you don't tell them how to use it, you might as well not give it to them.

Q. Were there any other reasons besides having them know how to use the API?

A. Yeah. Sure. Once an API has been documented, then other people can provide their own independent implementations of [974] that API, and so that's another reason to write good documents. If your documentation isn't good, then people won't be able to re-implement the API.

Q. When you talk about re-implement, again, that was the implementation code that we saw before; correct?

A. Yeah. That – that was like the code that actually told the computer how to reverse the list. So it would – like, you could write a new way of doing that.

Q. Okay. To what extent, if at all, did you expect that other people, other programmers, might create independent implementations of the APIs that you developed at Sun?

A. I certainly hoped they would.

Q. Why did you hope so?

A. Because it's pretty much the mark of a successful API. Once an API starts getting reimplemented, you know it has succeeded.

Q. What, if anything, did you do to promote the APIs that you developed at Sun?

A. Well, as I said, I – I gave lectures. I wrote books. I talked to engineers about them whenever I could. You know, pretty much did everything in my power.

Q. You just mentioned a book, Dr. Bloch. Let me hand you what's been marked as Trial Exhibit 7640. Do you recognize this document?

A. I do.

* * *

[991] would improve the specification.

As I mentioned, it takes a very, very good specification to admit an independent re-implementation. So by having these guys doing a re-implementation so early, they improved the quality of the specification, and that improvement was usable by every programmer who used this spec to do their programs.

And the other benefit is that when you have multiple implementations of an API, the skill sets transfer so someone who's learned it from GNU Classpath can then transfer those skills to Sun's JDK and vice versa. So it makes the skill set of learning these APIs more valuable if there are more implementations of it out there.

Q. Okay. To what extent, if at all, were you aware of anyone at Sun suggesting that it wasn't acceptable for GNU to do an independent implementation of those declarations that we saw of APIs that you wrote?

A. I was unaware of anyone saying anything of the sort.

Q. When you were at Sun, to what extent were you ever involved in re-implementing an API sort of from the outside world?

A. That same release, Java 5, under a project called nio, we added what was called regular expression processing to the language at the time. And that's a complicated word, but it's just a kind of text processing. When you're dealing with text, it makes things much easier to do.

[992] And it was a junior engineer by the name of Michael McCloskey who actually did the work. And instead of designing our own API from scratch, we decided we would use the regular expression API from this language called Perl 5. It was a – it was a large, complex, and well-known API to do regular expression handling.

Q. How did you go about doing the re-implementation of that specification?

A. We downloaded the specification from the Web, from their website, and then the engineer, Michael McCloskey, studied it until he understood it well enough to write a new implementation from the ground up without using any existing code.

Q. Why did you choose to re-implement the regular expression API from Perl 5 instead of creating your own?

A. Because it was really quite widely known. It was called a *de facto* standard. Every programmer – not every programmer, but most programmers who wanted to use regular expressions wanted to use Perl regular expressions, so we transferred their skill set from the Perl language to the Java language by implementing the same API.

Q. To what extent, if at all, did you seek permission from the folks, Perl 5, before doing your re-implementation?

A. To the best of my knowledge, we didn't seek permission at all.

[993] MS. HURST: Objection. Lacks foundation. Calls for a legal conclusion.

THE COURT: When you say "to the best of your knowledge," it begs the question how good your knowledge is. So tell us what you base your statement on that no permission was sought.

THE WITNESS: I didn't seek it. I was – you know, David Bowen, our manager, didn't seek it, but I don't know that anyone did. It seems unlikely that anyone did. Just based on the way we did things at the time, it seems very unlikely.

THE COURT: If someone had tried to obtain permission, to what extent would you have known about that effort?

THE WITNESS: I almost certainly would have known because we made extensive use of mailing lists, and it would have been on the mailing list.

THE COURT: All right. The objection is overruled. There is sufficient foundation for the testimony.

Next question.

BY MR. KAMBER:

Q. To what extent, if at all, were you aware of a license that would have allowed you to do an independent implementation of the regular expression APIs from Perl 5?

MS. HURST: Objection. Lacks foundation.

THE COURT: Just a moment.

Well, I think the same foundation applies. Objection [994] overruled.

Please answer.

THE WITNESS: I'm quite certain that we never sought a license.

BY MR. KAMBER:

Q. Why did you think it was okay to re-implement the Perl 5 regular expression API?

A. Because we've always done things this way. I've been in the profession for a long time, at Sun Microsystems from 1996 to 2004, and before that we have always felt free to re-implement each other's APIs.

Q. At the time that you were at Sun, were you – what, if any, industry or what, if any, sort of practice were you aware of with respect to re-implementing APIs?

A. You know, as I say, it was all over the place. FORTRAN APIs, which were designed by IBM–

THE COURT: Wait one second. This is – I'm going to say that is outside the scope of his work at Sun, and at this point, Ms. Hurst would have a good point. So I'm going to sustain the objection on the grounds that this is not his work at Sun. This is a more general statement about what others were doing. That would have to be expert testimony. So I'm sustaining her objection on – which she made earlier to this particular question.

[995] BY MR. KAMBER:

Q. Dr. Bloch, what did you do after you left Sun?

A. I moved on to Google.

Q. When did you move on to Google?

A. In 2004, just after Java 5 was released, the release that was documented in this exhibit.

Q. And you're pointing to TX984; is that correct?

A. I'm pointing to the third edition, yes.

Q. Now, what did you do at Google when you went there?

A. All things Java.

Q. What do you mean?

A. I had a very sort of wide-ranging role over my time there. I wrote Java APIs at Google for our own internal infrastructure. Google had a file system and it had something called a MapReduce bulk data processing system, and I led a team that implemented Java APIs for these things.

I contributed – I continued contributing back to this platform, so I actually wrote Java language features as well as Java libraries while I was employed at Google. I actually wrote documentation for the stuff from Java 5 and contributed that back while I was at Google.

You know, I gave talks. I – once again, I helped junior engineers with their designs, their APIs. You know, too many things to list.

Q. At any time when you were at Google, did you work on [996] Android?

A. Yes. I worked on Android for approximately one year starting at the very end of 2008 or the very beginning of 2009. I forget which.

Q. Okay. What did do you as a member of the Android team?

A. I worked on these same core libraries – java.util, java.lang – and I worked on implementations, independent implementations, of these libraries trying to

make them run their best on these mobile devices for which the Android platform was targeted.

Q. Okay. How, if at all, was the work that you did on Android related or specific to the fact that Android was this mobile platform?

A. It was actually quite specific to it because mobile devices have really different constraints from those servers and desktops for which Java 2 SE was written.

So, for example, a server or desktop is plugged in, you have infinite power. You don't have to worry about power consumption. But your cell phone has a little battery and if you use too much power, the battery runs out and that's bad.

So we had to always be conscious of how much power we were consuming. Phones have less memory. They have chips which are called ARM chips, which are far less powerful than Intel chips that run our servers and our desktops, and also it is just different instructions run at different speeds on these two [997] things. So one has to engineer them specifically to run their best in this constrained environment, and it actually can be quite a challenge.

Q. And remembering back to the code that we had up on the screen, Dr. Bloch, which part of the code was that optimization work being done in? Was it being done in the method declaration or the package or class declaration, or was it being done in the implementing code?

A. Of course it was being done in implementing code, as I hope I showed you. The declarations don't change. The declarations can't change. They are the nexus. They are what allows the caller of a function to

call it, but then you write a new implementation that is tuned for the new environment, and that's what we did.

MR. KAMBER: Thank you, Dr. Bloch.

Pass the witness.

THE COURT: Maybe we should take our 15 minute break at this time, give counsel a chance to set up.

Please remember the admonition. No talking about the case. Thank you.

(Proceedings were heard out of presence of the jury)

THE COURT: Be seated, please. The witness can step outside for a moment.

Can counsel give me the percentage breakdown on the Ellison and Duimovich read-ins?

* * *

[1013] name is Reed Mullen. I'm one of the lawyers representing Google in this case.

The next witness we're going to here from is by video. It's Mr. Donald Smith. His sworn testimony was taken on November 20, 2015, and at the time of his deposition he was an employee of Oracle.

THE COURT: All right. Don't play it yet until we get all this stuff out of the way; and then once we have the jury's undivided attention, hit the button.

(Pause in proceedings.)

THE COURT: Roll the tape.

(Whereupon, the video deposition of Donald Smith was played for the jury not reported.)

MR. VAN NEST: We need to start that again. Excuse us, Your Honor.

THE COURT: What's the issue?

MR. VAN NEST: Just the system needs to be switched, and madame clerk is going to do that right now.

(Whereupon, the video deposition of Donald Smith was played for the jury not reported.)

MR. VAN NEST: That concludes the presentation,

Your Honor.

THE COURT: All right.

MR. VAN NEST: We're ready to call our next witness.

THE COURT: Please do.

* * *

[1085] declaration and –

A. (Witness complying.)

Q. Perfect.

And now could you take a different color and identify for us where the implementing code or implementation would have been in this example from your time at Android?

A. Yep. So this right here (indicating) is the implementation. Too many letters. Okay.

Q. And if you can, maybe step either a little more that way or to this side, whatever side is easier just so the whole jury can see.

There you go, so everybody can see.

With respect to what you've identified as the declaration for the particular method max, that's the one that starts public static; is that what you said?

A. Yeah. That's right.

Q. When you were with Android, were there rules that governed how that had to be written?

A. Yeah. Absolutely.

Q. And could you give us an explanation of what kind of rules you were operating under at the time?

A. Oh, sure. Okay. Math.max is something that is available for application programmers to use, which means it has to be declared public. So this word "public" is required.

"Static" means that – it's something that we haven't [1086] really talked about here, but there is two different sort of categories of methods in Java. It so happens max has to be a static one given the way it's defined for programmers. So this word has to be there.

The name is – the name is the name, so there's no choice in what the name is.

Parens also have to surround these things called parameters.

The types of the parameters are well-defined. So as I said, int is short for integer, a kind of number. And then this names the parameters.

Q. Thank you.

And in terms of flexibility, how would you have characterized the rules governing how one writes declarations back at the time at Android?

MS. HURST: I'm going to object to that question, Your Honor. I don't know what "rules" means in this context.

THE COURT: It's also present-day opinion. Sustained. You can rephrase it.

BY MS. ANDERSON:

Q. Back at the time you were with Android when you were working on the Java language, what did you understand to be the amount of flexibility you had in how you would write a declaration for a method like max?

MS. HURST: Objection. Leading.

[1087] THE COURT: No. It's not leading. That's overruled. Please answer.

THE WITNESS: I would say that there's very little flexibility.

BY MS. ANDERSON:

Q. Why is that?

A. So the names of the – the places where these names occur, there's – there's fixed positions – there's some flexibility, like these two could have been reversed. There's a standard way of doing it. So it could have been, but probably wasn't, the name just – the name had to go here. There's no other choice about that. The parens had to go there. No other choice about that.

These names of these types, the int, there was no choice about that. And there was, say, some flexibility in the names of these parameters. That would – I think that covers the flexibility.

Q. And, again, back at the time you were with Android, what did you understand to be the relative amount of flexibility available to you in writing the actual implementation for this method?

A. For the implementation?

Q. Yes?

A. Much more flexibility. That's where more of, say, a programmer's experience and taste in sort of design limitation [1088] would come into play.

Q. We've heard the phrase "independent implementation." How, if at all, does that relate to the testimony you just gave, back at the time you were with Android?

A. Oh, at the time. So an independent implementation, it's talking about what would go here and, of course, there's lots of packages, lots of classes. So it's sort of all of the peers for the many, many methods in– in the – that are defined by the API. So an independent implementation is just somebody taking the knowledge that they have about how Java works and writing a whole bunch of new code to implement it.

Q. When you said "here," you were pointing to which box?

A. The orange implementation box.

Q. All right. Thank you very much.

So I think you can take the stand again. Thank you, Mr. Bornstein.

Putting you back again to the time you were on the Android team, at the time did you have a view as to whether or not you were free to use the Java programming language?

A. I did.

Q. What was your view?

A. I believed I was free to use the Java programming language.

Q. Why did you believe that at the time?

A. Because programming languages are made to be used. [1089] They're made to be used by programmers. There would be no point in its existence for it to be somehow published in, you know, like, a book like this, to have documentation made available if the intent wasn't to make it used.

And there's, you know, long history of, like, my entire career of, you know, new programming languages coming along and things get published about them and people use them in various ways.

Q. Again back at the time you were working on the Android team at Google, did you have a view as to whether or not you were free to use declarations for Java APIs?

A. Yeah.

Q. What was your view?

A. I thought that the – that Java declarations were A-okay to be used.

Q. Why was that?

A. Because I had seen at the time many – many examples of a programming language coming along published by or built by one set of people, and a different set of people come along and do a new implementation of that language and inevitably you use, you know, the same declarations.

Q. Okay. Do you have a particular example in mind?

A. I'm thinking of the C programming language or C++. Very early in my career I actually had reason to use – to build a system that was mostly in the C programming language, and we [1090] used three different C compiler and runtime products from three differ-

ent companies, but we used the same code base. So we used our – like, our application code was the same. We used three different C implementations to provide the executables, the applications for a few different platforms.

Q. You just described, when you were showing us the sketch you made on the easel, information that you knew back at the time in Android about how much flexibility you did or did not have in how you wrote a declaration.

Back when you were with Google, did that have any effect on your view as to whether or not declarations for the Java APIs were available for use?

A. Okay. That was a bit of a mouthful. Could you–

Q. No problem. I'm happy to –

MS. HURST: Your Honor, I'm going to object. This seems to be about designing APIs. There has been no foundation laid that this witness was in any way involved in that. That was Dr. Bloch's job. We already heard from him.

MS. ANDERSON: Your Honor, we are simply explaining the witness' testimony, and this witness, we will be discussing momentarily, was very much involved in–

THE COURT: Why don't you go at it a slightly different way and tell the story up until the point that he gets to where he's using or redesigning some API, and then you can ask to what extent did you feel you were able to do that;

* * *

[1097] So at one point, I issued some commands on a command line and – which caused the source code

from the Apache Harmony project, hosted at the Apache organization or hosted on their servers – that came – so that was – via network was pulled onto my local computer. From my local computer I issued another series of commands that caused that code to then get integrated into the Android code base.

THE COURT: And so that example came from Apache Harmony?

THE WITNESS: That's right.

THE COURT: And that was, at least as you're telling us – it was the Java APIs, but with an independent implementation? Is that correct or not correct?

THE WITNESS: Yeah. It was an independent implementation of Java APIs.

THE COURT: Okay. All right. Thank you.

Go ahead.

MS. ANDERSON: Thank you, Your Honor.

Q. You mentioned a minute ago, Mr. Bornstein, that sometimes Google had to work on modifying the open source Apache Harmony code that it had downloaded to work on. Could you give us some examples of why you would have to do that?

A. Sure. Apache Harmony – I guess – the thing to remember is any given software project has sort of a context in which it's sort of envisioned to operate. Apache Harmony happened to [1098] be one that was targeted towards servers and desktop computers and not for mobile devices. And there's a lot of differences between a big computer that sits on a desk or on a data center and a little computer that fits on your hand and runs on a battery.

So a lot of what we were doing when we were modifying the Apache code was finding those places where it– you know, where the code was kind of – assuming it was in a – in a different context and changing it– changing the code to no longer make those assumptions and to in fact make different assumptions.

Q. At the time when you were doing this work, did you have an understanding as to whether the Apache Harmony code was an independent implementation of a particular Java platform?

A. When you – I'm not sure what you're asking. I'm sorry.

Q. Sure. We've talked a little bit about independent implementations and open source. Did you have an understanding as to which, if any, platform Apache Harmony was an independent implementation of?

A. I would – speaking very informally, I would have said it was an independent implementation of Java, but there's a lot to unpack in that.

Q. Okay. And you mentioned that Apache Harmony's implementation was more targeted at desktops and servers. How did you know that?

* * *

[1104] A. The Android engineering team as a whole.

Q. Were those the same or different than the Java API packages?

A. They were different.

Q. Why did Google develop Android API packages?

A. Because the short version is there wasn't anything like what we wanted to achieve out there in the

world, so that was something that we had to write ourselves.

Q. Could you give us a few examples of goals you wanted to achieve with the Android API packages?

A. Sure. So, for example, we intended to have a smartphone that did multiple – multiple – it was able to run multiple applications simultaneously, and there's some – that has implications on how an application itself is designed, so we had to define Android packages and classes to enable application developers to play in that – in that world of multiple applications, running on a smartphone-type device.

Q. Thank you.

Did you have any role in determining which Java API packages would be implemented as part of Android's core libraries?

A. I did.

Q. What was your role?

A. As I said before, I was the technical lead for the core libraries, and among the duties that fell to me was figuring [1105] out what made sense to have in our Java APIs.

Q. When you say "figuring out what made sense," what do you mean by that?

A. What I mean is that Android was – was this different context for running code in general. We were looking at a – you know, we were looking at a previous, you know – previously-exiting set of APIs that had been defined. Not all of them made sense in the context of a smartphone, and it was in part up to me to help figure out which pieces of that made sense to sort of – to include in our core libraries.

Q. And did any of those considerations relate to specific aspects of a mobile device?

A. Yeah.

Q. Can you give us a few more examples of those considerations.

A. Oh, so – let's see. So there were – I don't know. I'm sorry. I can't think of, like, a very good example for you.

Q. That's fine. No problem.

Did your team create implementations of all the Java APIs in Java SE?

A. No.

Q. Why is that?

A. It didn't make sense.

Q. Why didn't it make sense?

A. Like I said, I wish I could give you a concrete example. [1106] It's been a while. But in looking through, you know, like what was being defined as part of Java SE, there was – there was some stuff that just – you know, just didn't really make sense to have– actually, here's a good one.

So Java SE included its own idea of what an application looked like. And that implied kinds of interaction models of like what, you know – what it means to run two applications at the same time, what it means to switch between them, how does an application start up and shut down, and we didn't want to use any of that for Android, so it did not make sense to include that in– among the APIs that we were using on Android.

Q. All right. Thank you.

Approximately how long did it take, based on your experience while you were at Android, to do the development work required to launch the first full source code stack for Android?

A. So I – I joined in October 2005. The project was already underway. We released the first phone using it – "we" in a larger sense – in, what was it, October of 2008. So – and I guess before I joined, the project was ongoing in one form or another for something like a year, year and a half. So add that all together.

Q. All right. In or around 2006, did you hear of any announcements by Sun relating to open source?

* * *

[1129] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOGGLE, INC.,

Defendant.

San Francisco, California
Monday, May 16, 2016

Before the Honorable William H. Alsup

TRANSCRIPT OF PROCEEDINGS

---

* * *

[1194] system. Those kind of things which weren't part of Java.

Q. So in contrast to Java, Android, the Google platform that eventually came out, Android was a full stack; correct?

A. Uhm, in how – how it was referred to in that time, my understanding is yes.

Q. And Apple's iPhone operating system, that was also a full stack; correct?

A. Oh, it's a completely closed platform. So I don't actually – I don't know, actually, what's in there. But

the highest level of trying to get – characterize those things, yes.

Q. At the time Sun bought Savaje, Sun didn't have a full stack on the market; correct?

A. That's correct.

Q. But you do know that Sun, after acquiring Savaje, attempted to turn the Savaje technology into a full stack platform; correct?

A. I know there were plans to build a stack of some sort. I don't know if it was a completely full stack or a mostly full stack. That I don't know.

Q. The project that Sun pursued to build a stack from the Savaje technology, that was internally at Sun called Project Arcadia; correct?

A. So there were multiple projects for that basic idea of the vertical offering. It was called Arcadia at one point in time, [1195] but it changed names a few times based on where it was in the organization, who it was reporting to. And I'm not sure if the feature sets changed or not with it. It was referred to with multiple names over a period of time.

Q. If Sun had gotten the Project Arcadia technology to market, it would have had a full stack on the market to compete with Android; correct?

A. I don't remember the exact details of what Arcadia was versus the initial Savaje acquisition. So I don't recall if the Arcadia project was still a complete full stack or mostly a full stack. I just don't recall the details.

Q. Mr. Gering, as you sit here today, you're not aware of any product that Sun brought to marked based on the technology Sun bought from Savage?

A. No.

Q. And, certainly, Sun never brought a full stake mobile operating platform to market based on the Savaje technology; correct?

A. No, not to my knowledge.

Q. In fact, during your time at Sun, Sun never brought a full stack mobile operating platform to market at all, did it?

A. No.

Q. All right. So you were at Sun when Google released Android; correct?

A. Yes.

[1196] Q. And you were aware that Google had released Android?

A. I was aware of Android being in the marketplace. I don't know the exact date.

Q. After Google released Android, Sun made an effort to develop technologies that would work with Android; correct?

A. There was a point in time when we did technical explorations of various technologies that we had in-house, with Android, for different reasons.

Q. I've just handed the witness Trial Exhibit 2052.

Mr. Gering, there is a presentation on – a Sun-formatted presentation titled "Java and Wireless Business Review." Do you see that?

(Document displayed.)

A. Yes, I do.

Q. And your name is there on the front page; correct?

A. Yes, it is.

Q. It's dated March 16, 2009?

A. Yes.

Q. So just going by the timeline, that's after Google released the Android platform in October 2008.

A. Okay.

Q. Could we turn to page 20 of the document.

(Document displayed.)

Q. This page discusses something called Project Daneel. Do you see that?

[1197] A. Yes, I do.

Q. And Project Daneel was also known inside Sun as Project Sundroid; isn't that right?

A. There was a Project Sundroid. There was a Project Daneel. They had a lot of the same similar characteristics. I don't remember if they were exactly the same or not.

Q. All right. The idea of both Project Daneel and Project Sundroid was to try to insert a Sun Java virtual machine into the Android platform in place of Google's Dalvik virtual machine; right?

A. Yes. Daneel project had two – it had multiple phases. The first phase was to put Sun's VM and stack next to the Google stack that was 0, a Google VM. So it had two VMs on that stack. And that was called a Google stack approach.

And then the second, the Phase One, which was the second phase, was to actually replace the VM with Sun's VM.

Q. And that's reflected here on Trial Exhibit 2052. There's a reference to Phase 0 and Phase 1?

A. Yes.

Q. And there's also Phase 2, which is a full Linux platform. Do you see that?

A. I do.

Q. So Project Daneel ultimately would have evolved into a full stack. Is that how you understand that?

A. So my memory of Daneel is Phase 0 and Phase 1 were fairly [1198] well defined. And Phase 2 was not as well defined, at least as I recall.

Q. So as far as you recall, Sun never really developed a concrete definition of Phase 2 of Project Daneel?

A. More accurately, I think there were multiple definitions at that Sundroid-Daneel time. But I just don't recall what were the contents of that bucket, because we were focused – the engineering team was focused on Phase 0 and Phase 1.

Q. With respect to Project Daneel, Sun got as far as developing a Phase 1 prototype of a Sun virtual machine running on the Android platform in place of the Dalvik virtual machine; is that right?

A. That's correct.

Q. But that was as far as it went; correct?

A. As far as I know.

Q. The product that was developed in Project Daneel never got to market; correct?

A. Correct.

Q. Mr. Gering, I just handed you Trial Exhibit 2061.

(Document displayed.)

MR. PURCELL: If we could blow up the top half of that.

Thank you, Ben.

BY MR. PURCELL

Q. This is an email that you sent to Vineet Gupta in [1199] January 2009; correct?

A. Yes.

Q. First off, Mr. Gupta, in January 2009, his job at Sun was negotiating Java licenses with manufacturers of mobile phones; correct?

A. He was the CTO of the – he was in charge of the SEs and also the CTO for the embedded sales force. So as part of that responsibility, he was involved in those discussions.

Q. Mr. Gupta is referring there in the second paragraph, "I've been getting several requests regarding partnering with us to provide a Dalvik/Java ME combined platform. Samsung is really pushing for partnership discussions ASAP."

Do you see that?

A. Yes.

Q. And in the next paragraph he refers to, "Samsung, HTC, Sprint, T-Mobile, LGE are the top candidates approaching us."

Do you see that?

A. I do.

Q. Those are some of the most prominent phone manufacturers in the world, aren't they?

A. Yes, they're a subset of them, yes.

Q. Despite Mr. Gupta's optimism that there were these opportunities out there for Sundroid, with some of the most prominent mobile phone manufacturers in the world, Sun still never managed to get a Sundroid product to market; correct?

[1200] A. Sun did not bring a Sundroid product to market.

Q. Mr. Gering, this document is Trial Exhibit 3508. And, Mr. Gering, this is an email you received in October 2009.

Do you see that?

(Document displayed.)

A. Yes, I do.

Q. And it attaches a couple of presentations?

A. Yes, I see it.

Q. If we can just look at the first presentation right after the cover email. It's called "OneJava Market Landscape Discussion." Do you see that?

A. I do.

Q. And if we could just go to the second page.

(Document displayed.)

Q. Looking at the second bullet point there, that's "Sun's leadership around Java is perceived as stagnant, and Java is considered legacy."

Do you see that?

A. I do.

Q. First bullet under that says, "Stagnant innovation." Do you see that?

A. Yes.

Q. The third bullet says, "Fragmented between Java SE and Java ME, and between Java ME mobile and TV and within mobile and TV."

* * *

[1208] teaching computer science.

Q. So you're familiar with the Java programming language?

A. I am. I have written many programs in Java. I teach many courses that use Java programming language.

Q. How did you first learn Java?

A. I first learned Java right in about 1995, when it came out, so I could begin teaching with it. And I read books and used some online sources to help me understand how Java worked and how the API libraries with it worked as well.

Q. When you first learned Java, how did it compare to computer languages with which you were already familiar?

A. Java is an object-oriented language, which means it uses classes. And that's similar to C++, a language with which I was very familiar because I had been using it for several years in both my research and my teaching.

So it was relatively straightforward to pick up Java because, conceptually, it was related to C++. And, also, the API libraries were similar to C and C++ as well.

Q. What courses, if any, have you taught on Java?

A. The first course I taught in Java was in 1996. That was an advanced course in software design. And we continue to use Java in that course today.

We also began using Java in our first courses for majors in the early 2000s. And I still teach a course with Java. That's the second course that our majors take, and we continue [1209] to use it there.

Q. Have you won any awards?

A. I've won several teaching awards at Duke for my teaching. I've won an award, when I was on leave, at the University of British Columbia, in Canada, for teaching a Java course when I was there for one year.

And I've won some awards from the National Science Foundation.

(Reporter interrupts.)

A. Sorry.

I've won an award for teaching a Java course at University of British Columbia. I've also won some awards from NSF, for the work I do on my research and teaching.

MR. PAIGE: Your Honor, may I approach the witness?

THE COURT: Yes.

BY MR. PAIGE

Q. Professor Astrachan, I've handed you Trial Exhibit 7642.1. Could you take a look at it and identify that document?

A. This is a copy of my curriculum vitae, my CV or resume.

MR. PAIGE: Your Honor, me move the admission of TX 7642.1.

MS. HURST: That's fine, Your Honor.

THE COURT: All right. It's received in evidence.

(Trial Exhibit 7642.1 received in evidence.)

[1210] BY MR. PAIGE

Q. Professor Astrachan, have you been retained by Google in this case?

A. I have.

Q. And are they compensating you for your time in this case?

A. They are.

Q. Okay. What assignment were you given by Google?

A. I was asked to look at the 37 API package labels that are at issue in this case and to develop opinions about how those API package labels are used on the Android platform as part of creating Android.

Q. What did you do to form your opinions on that subject?

A. I used my understanding of programming languages, and of Java in particular. I used my knowledge of Android and programming. And I wrote software that I used to analyze the code base for both Android and Java SE, as part of developing my opinions.

Q. And could you just tell the jury, at a high level, what the opinions you developed were.

A. At a high level, Google is using the 37 API packages, the labels, the method declarations from these 37 API packages in creating a new context, the Android Operating System.

So in my technical analysis using the code bases for these, I've seen that the 37 API labels are combined with new implementing code as part of creating the

Android Operating [1211] System and full stack platform.

And they've used C++ and Java libraries also optimized and designed for a mobile platform in creating Android.

I also see that these API levels, the method declarations and class declarations, are by nature functional because that's what API labels are. And, in particular, these API labels are short, descriptive and functional in terms of what they do. And the API label declarations are very small part of the Java SE platform.

In creating – in develop – in using these labels to make Android, we see that Java is still the number one programming language in the world, and these API package declarations are part of the OpenJDK release of Java SE.

Q. So before we go into the details of your opinion, I would like to have you give the jury a little background.

Can you explain to the jury what a computer programming language is.

A. Sure. And I have a slide that shows some of this information.

At the lowest level, programs run on computers. That's why they're called computer programs. And those are 0s and 1s.

But when we talk about programming languages, we're talking about languages like Java and C++. That's the source code that you see in this diagram on the right.

And the process of taking a program written in these [1212] high-level languages, in Java and C++, if you

look at the language, they look a little more like English, a language that you would, kind of, be able to read and write, not like the 0s and 1s that are actually executed on the computer.

And the high-level source code is translated – in the diagram you see it's compiled – into the binary code or 0s and 1s that are executed on the computer.

Q. And you have two different types of setups there.

Can you explain to the jury what the top and bottom ones are.

A. Sure. In some languages, like C++, when that source code is compiled or translated into the 0s or 1s, that binary code runs directly on the hardware.

And in a language like Java, there's a virtual machine. And the source code in Java is compiled into bytecode. And that bytecode is run on the virtual machine, which turns it into the 0s and 1s that are executed on the computer.

But their process is the same in both languages. Starting with source code and ultimately getting to the 0s and 1s that are the computer program that runs.

Q. What are the differences, if any, between languages that use virtual machines and those that do not?

A. The virtual machine has a small overhead. So often programs that are run on the virtual machine might run slightly more slowly than they do for programs that don't use the [1213] virtual machine.

Q. Was Java the first language to use a virtual machine?

A. No. Virtual machines had been used for years before Java. The P-code virtual machine ran both Pascal and PL1. So there were virtual machines in existence before Java.

Q. Why do computer programmers use high-level computer programming languages?

A. It would be really hard to write a program just with 0s and 1s. So the high-level programming languages allow programmers to be productive and effective in making the programs and applications that they do as part of their job and their hobbies.

Q. Can a computer programmer who writes in one language generally write in any computer language?

A. Well, you have to pick up the new language. But once you've learned one language and the libraries that are associated with it, it's usually reasonably straightforward to pick up a new language and the new API libraries, especially if those languages are similar.

But computer languages are much closer to each other than, say, if I already know Spanish, it would be really hard for me to learn Chinese. The alphabets are different. The words are different.

But when you know Java and the libraries associated with it, it's relatively straightforward to be able to pick up a new [1214] language because the source languages and the libraries are often very similar.

Q. Now, could you explain to the jury what an application programming interface is.

A. Sure. We've heard "application programming interface" or "API."

And that's a piece of software that allows you to connect my – the program I'm writing as the developer with code written that's stored in the library. So the API is one way that I can write my code and use code that someone else has developed.

Q. What is an API used for by programmers?

A. Well, as I explained, when I write my source code, my software, I could write everything myself. But some programs would be unbelievably long and complicated.

For example, the process of opening a Web page in a program would be really long. Or Internet protocols and Web protocols I would have to understand, I would like to just be able to say "open a Web page," and then have thousands of lines of code that were needed to actually open that Web page and get it. It would be wonderful if those were already written and debugged and robust and I could just use that code.

So what the API does in this case, the label "open Web page" would be enough for me to use in my program and then access the thousands of lines of implementing code that had [1215] already been written and tested.

So in that case, the API is, I write the code that says "open Web page," and then I access the step-by-step functionality that's part of the implementing code that lets me actually accomplish that task.

Q. Now, can you provide the jury with an example of something in everyday life that's comparable to an API?

A. Sure.

I have an example that I often use in my classes. And I have an exhibit that we've created to kind of explain that.

Probably everybody has been in a car. Most of us have been a driver in a car.

And when you get into a car, even if you've never had that model car before, whether it's a convertible or a pickup truck or a smart car, we know that the steering wheel, when you turn it to the left, the car goes to the left. When the steering wheel turns to the right, the car goes to the right. And that works whether the steering is a rack and pinion or power steering.

So the steering wheel is this, kind of, API that allows me to operate the car without knowing how the underlying steering mechanism works.

The accelerator works the same way. I know, in any kind of car I get into, when I press down on the accelerator, the car is going to go. And when I ease up, the car slows down.

[1216] And that works whether it's an electric car, a V6, a V8, fuel injection.

(Reporter interrupts.)

A. Sorry. Sure.

If the car has an electric engine, or a fuel-injected engine, a V6, a V8, when you press down, the car goes. And I don't have to know how the drive shaft works our how the pistons work. So all that functionality of how the engine works is accessed by the accelerator that works the same way.

Brakes work in a similar way. When I press the brake, I know that some kind of brake, disk brake, caliper, is going to stop the car.

So these aspects of driving, steering wheel, accelerator, brake, they serve as a kind of analogy because I can use them to accomplish the task of driving my car without knowing how the underlying system works, without having to understand what kind of engine it is.

And the API software also allows me to access that functionality of the implementing code without needing to understand exactly how it works, just being able to rely on the step-by-step instructions that I've accessed by using the label declaration of the API.

Q. How does use of APIs help computer programmers that are trying to program in other contexts or other platforms?

A. Well, I talk about how, first, it saves me the time from [1217] having to write the thousands of steps that might be needed.

And if an API in one language is the same between platforms, whether I might be writing for a desktop or a mobile device, if I can rely on that API being used to access that same functionality, that will help me develop software more easily.

Q. And can you –

THE COURT: Can I ask a question on this?

MR. PAIGE: Of course.

THE COURT: It sounds like, you know, you've drawn a distinction between the label or the declaring code and then the implementing code; right?

THE WITNESS: Yes.

THE COURT: Okay. As you use the term "API," it sounds like you're referring to the collection of all of

the labels and not including the implementing code; is that correct?

THE WITNESS: I'm – I'm trying to be careful because I understand that what we're talking about here is just the declaring code. And so I'm trying to say declaring code in API.

But API as a term is reasonably broad. And in teaching with it and understanding how software engineers use it, an API by itself could refer to API services, or the implementing code, or, kind of, a general understanding of how to use the [1218] API.

So I'm trying to be precise here in saying the label declarations, that's the declaring code that we're talking about. But I might slip a few times and say "API," and end up encompassing the implementing code.

So what I know here is that we're talking about just the declaring code for what Google used at the beginning of creating the Android platform.

So I'll try to not use "API" in this all-encompassing way. Although, I think that's part of the general confusion that we've seen that can mean this conceptual piece or the implementing code.

THE COURT: All right. Thank you.

Go ahead.

BY MR. PAIGE

Q. It seems like it's a good time, Professor Astrachan, to perhaps explain to the jury the parts of the API. Could you do that?

A. Sure.

The API – and I've got a diagram that shows this; and this goes to what Judge Alsup just asked– includes, for the purposes of what we're talking about here, the method declaration. That's the label that we've been referring to when we talk about the labels of the declaring code.

And that's in yellow, that you see on this slide. And [1219] this particular method, we see the name of the method. That's shown as "compareto." And in Java, that would be in a class, in a package, that's also part of this API not shown there.

And the method declaration "compareto" – API methods also have an input and an output. We've heard that before.

You can think of things like in a calculator when square root might be the name of the API. And it has input, you put in the number 25. And then you have an output. You get back 5.

In this case, the method declaration has a name, "compareto"; an input, that's labeled as "String anotherString," that's what goes into this method; and then an output. That's the return type. That's shown as "int."

So, in general, all these method declarations have a name – and in Java that includes the class and package name – and an input or parameter, and output, the return type. Name, parameter, return type.

Q. And can you point to where the return type perimeters are found there on the slide?

A. When you read "public int," the "int" is the return type; "compareto" is the name of the method; and then "anotherString" is the parameter.

So it starts with the return type, then the name of the method, and then the parameter. Those all together create the method declaration.

[1220] Q. Now, can you explain to the jury what the implementation of an API is?

A. Sure.

I've talked before about how this one label allows me to access all the functionality as a programmer, so I don't have to write it myself over and over again.

Here, the implementing code is shown in gray. And that's this step-by-step sequence of instructions that would actually get ultimately executed as 0s and 1s when I call the API label.

So I use the label to access the functionality. And that gray step-by-step sequence of instructions then returns my result. So there's the declaration and the implementing code.

Q. And what's at issue in this cases, Dr. Astrachan?

A. In this case, what's at issue is just the declaring code, just what you see in yellow on that slide, the return type, the name that includes the package and class, and the parameters.

Q. Can you explain to the jury what might happen if you had APIs change?

A. Sure.

Here's another example that might make sense for what an API is.

If you use software and you print a Web page or print a wordprocessing document, sometimes you see "print" in the file menu. And that might be control P or command P. That just makes things print.

[1221] What would happen if, all the sudden, control P or command P meant paste, because P starts with paste. Then printing wouldn't work anymore. So that if the API changed so that command P meant paste, then users of that file menu and their software wouldn't be able to accomplish their tasks.

The same thing would happen for a software developer. If the API labels change, then either the software wouldn't continue to work anymore or the developer would have to use a whole – would have to learn a whole new language to be able to use these API labels.

Q. Can you explain to the jury what libraries for programming languages are.

A. Sure. A library – well, in this file menu that I talked about, if you've used software before, you know in the file menu you see "print" and "new" and "open" and "save." That's a collection of operations that are in one place. The file menu.

And a library in software is kind of the same idea. Classes or ideas that are grouped together are in the same library. And the methods that are in the class that are in a library all would be related functionality. So a library is a collection of related software.

Q. Are there other names for libraries in computer languages?

A. There are.

In Java, we use the word "package." That's, kind of, a required name in Java. And "package" is a library.

[1222] A package contains classes in Java that are related. Conceptually related. So a package in Java is a library.

Q. And what's the relationship between libraries and computer programming languages?

A. In order to make effective use of a programming language, you need libraries. We can't write all the code ourselves. And sometimes it wouldn't even be possible, without libraries, to do things like print or make your program run.

So for developers and programmers to be effective, you have to have libraries that are essentially associated with the language, to be a productive program.

Q. I would like to turn, now, to talk a little bit about Java.

When were the Java APIs initially created?

A. The Java APIs, along with the Java programming language, were first introduced to the public in about 1995. So we had the language and the APIs that were released at the same time.

Q. And you've said in Java that libraries are called "packages." Can you explain to the jury how Java organizes the material within those packages?

A. Sure.

I mentioned that Java is an object-oriented language, which just means that we use the word "class" to encompass a bunch of code, a bunch of concepts that are realized in code.

So in Java, a package is a collection of classes. That's [1223] required by the language. And each class is a collection of methods and a few other things.

So the organization in Java, that's required by the language, is a package. It's a collection of classes. That's software. And each class is a collection of methods.

So these labels are methods within the class that are part of a package in Java.

Q. And what's the order of naming in these labels?

A. We typically, and the way Java kind of requires things to write is you say, package name, class name, method name.

Q. And how do the names of the labels relate to the structure, sequence and organization of Java SE?

A. Well, the Java Language requires that we use package name, class name, method name in describing these labels.

So if we saw something like the max method, that we saw Mr. Bornstein write, that would be java.lang, that's the package, .map, that's the class, .max, that's the method.

So that sequence of package name, class name, method name, that's required by the language, and that's how Java works.

Q. So is the name of the method interchangeable with the SSO of the method?

A. I treat the names – because they start package name, class name, method names. That is the structure, sequence and organization that Java requires us to use. So I, kind of, treat those declarations in the SSO as the same.

[1224] Q. What's your understanding of Java's place in the world of programming languages today?

A. I know that Java is really widely used in teaching in the academic setting I work in. And I know that the students I teach go out and get jobs writing Java. And I also know that on Oracle's website it says

that Java is the number one programming language. So it's really widely used.

Q. And do you have an understanding as to how Java became so popular?

A. Well, I know in part how it became popular.

When Java was first released in 1995, Sun made a great effort to make Java and the API libraries available to both me in my teaching responsibilities, but also to companies that would be able to use Java. And they developed programming environments. Some specifically for professional programmers.

We've heard, maybe, about NetBeans here. But also for beginning programmers. So there's a programming environment called Blue Jay, that was supported by Sun, designed specifically for teaching.

So Sun took great, kind of, care and steps to make sure that Java and the APIs were both well-known and easy to use for both teaching purposes and for developers writing programs.

Q. Now I would like to talk about the Java platform.

Could you tell the jury about what versions there are of the Java platform?

[1225] A. We've heard about three platforms here, and I, kind of, have a graphic that describes them all.

The Java Standard Edition, that we see in the middle, that's Java SE, that's used by developers to create programs that run on desktop and laptop computers and maybe small servers. So Java SE is the platform that we're talking about here. And it has about 166 packages in it.

Q. What's the Java Enterprise Edition?

A. The Java Enterprise Edition, that you see on the left, is used for enterprise applications. That would be, kind of, big server applications or things that you would deploy in a business with thousands of computers.

And that has a hundred more packages, roughly, than we see in Java SE because those programs have, kind of, a different functionality or purpose than they do on the ones – than the ones that run on your laptop or desktop computer.

Q. And can you tell the jury a little bit about the Java Micro Edition?

A. We see there on the right the Micro Edition that we've heard it's a programming platform used on feature phones. And that has far fewer of the packages that we're talking about here. It has about 10 packages.

Q. Which of these versions of Java is at issue in this case, Professor Astrachan?

A. The API declarations, the label declarations that we're [1226] talking about here, come from Java SE, the platform that's designed to create programs that run on desktop and laptop computers.

Q. And what specific packages are at issue in the case?

A. There are 37 packages that we're talking about here. And we can see them listed up there.

We don't have to go through all these packages, but these are all the package names. And you can see they start with either Java or JavaX. And then the names essentially describe what you'd expect to find in those classes – in those packages.

Q. Now let's talk a little bit about Android. Could you explain to the jury what the Android platform is?

A. Sure.

We have a picture of the Android platform that I could use –

MR. PAIGE: Your Honor, may I show the board?

THE COURT: Of course.

MR. PAIGE: Exhibit 43.1 in evidence.

THE WITNESS: It would be helpful if I could go over there and point –

THE COURT: Be my guest. Just keep your voice up.

THE WITNESS: Okay.

THE COURT: Can the jury see that okay? We'll move it if you can't see.

[1227] Why don't you move it closer to the jury box.

THE WITNESS: Okay.

MR. PAIGE: Certainly.

THE WITNESS: Here we go.

THE COURT: How many lawyers does it take to move an easel?

(Laughter)

THE COURT: Can you all see now?

All right. Mr. Paige, you're going to have to move back. I think you're blocking the view of some of the jurors.

THE WITNESS: Stand to the side.

THE COURT: So let's go ahead.

BY MR. PAIGE

Q. So could you explain what exists at the lowest level of the Android platform?

A. At this lowest level, we see the Linux kernel. That's the low-level operating system. And this is an open source kernel that Google used and made, kind of, special for this mobile platform. And that's what's at this lowest level.

Q. And what exists above that lowest level in the Android platform?

THE COURT: Mr. Paige, would you scoot back one more step.

MR. PAIGE: Of course.

THE COURT: There we go.

[1228] THE WITNESS: We can see here what's labeled the "Android runtime." And that consists of the core libraries, which are the new implementations of the packages from which these 37 API package labels come from. So those are included here, as are other libraries that are designed for this mobile platform.

So these are the Java libraries, including new implementations of the 37 packages. And then new libraries that are part of creating this mobile platform. Those are part of this Android runtime.

BY MR. PAIGE

Q. And what are those new libraries for, Professor Astrachan?

A. Well, these libraries would be for things like making Web browsers. Or in a smartphone, it has location awareness for GPS. That's not something that

you would expect on a laptop or desktop computer, where Java SE comes from.

It has accelerometers. When your phone shakes, something happens. It has a camera. Those are also features that you wouldn't expect on desktop or laptop computers.

So many of those libraries are designed specifically for the mobile platform, which is a different platform from where the 37 API packages came from.

Q. And beneath the core libraries what is that, Professor Astrachan?

A. Well, that's labeled here the "Dalvik virtual machine." [1229] And we either have a Dalvik virtual machine or the Android runtime.

If you remember back from the first slide I had, where translating a programming language into the 0s and 1s used a virtual machine, this is a virtual machine designed specifically for a mobile platform.

It has smaller bytecodes than you'd find in the virtual machine on the Java SE platform. It's designed to run more efficiently on a mobile platform, which has battery capabilities that are different than you would find on a desktop computer. And it runs on a mobile smartphone platform that would have less memory, for example.

So this virtual machine is designed specifically for this Android smartphone platform.

Q. And in the green area, can you explain to the jury what that is, Professor Astrachan?

A. We can see here that those are labeled "libraries," which we know are collections of code. And in this

case these are open source or public domain libraries that were written in C++ or maybe Java.

And they are specific, again, for a mobile platform, a smartphone platform. Web.Kit, for example, is software produced by Apple that allows us to create a browser that would run on a smartphone.

We have Open GL embedded system. That's a graphics [1230] library that makes graphics run quickly and smoothly.

SQlite is a database that allows the mobile phone to access the database.

In general, these are libraries that are part of the mobile platform. And these are open source libraries that are integrated with the core libraries as part of creating the Android platform.

Q. If a developer wanted to create an application, how would a developer do that on the Android platform.

A. Top level applications, that is where a developer would create an application, like a contact list or a phone screen that you see over here over here. Those applications, which are largely written in Java or they can be written in C++ or C – those applications take advantage of an application framework, which is a set of services kind of provided by the Runtime in these other libraries.

So the key here is that the applications are written at this high level and it makes use of all the libraries here and then we heard how those also depend on the Linux kernel down below.

Q. Thank you, Professor Astrachan.

Now, I believe you had mentioned that the – the applications could be written in Java or C++. How would one write an application in C++ for Android?

A. Well, an application written in C++ on Android would use [1231] what is called the NDK. That's a library instead of code that Android provides to use C++ in creating software that would also run in the Android platform.

Q. How is the Android platform distributed?

A. The Android platform is distributed as an open source platform. So that means the source code is free for anyone to use.

Q. Now is the Android platform compatible or intraoperative with the Java SE platform?

A. No, it's not. We talked about how the Java SE is designed for laptop and desktop computers, and an application written for those would use likely the 37 API packages and their implementations, but maybe the hundred-plus more that were there for a desktop or laptop computer.

So if it used all those API packages, the declaring code and implementing code to run on a desktop or laptop, we wouldn't expect it to run on a mobile device because it wouldn't use all those API packages.

And similarly, if we wrote an application that ran an Android, it would – it might use some of those 37 independent implementations in the packages, but it might use the accelerometer and the location services, and if it used those, those new libraries that were designed specifically for the Android platform, it wouldn't work on your desktop or laptop computer. So in general, those platforms aren't compatible.

[1232] Q. Was it necessary for Google – was Google required to replicate the API labels in SSO of the 37 Java SE API packages in order to use the Java programming language?

A. I understand that Oracle has said that roughly 60 classes in 3 packages are constrained –

MS. HURST: Your Honor, I'm going to object. This is beyond the scope of the witness' opening report.

THE COURT: Is that true?

MR. PAIGE: It is, Your Honor.

THE COURT: Sustained.

MR. PAIGE: It's required by your motion in limine.

THE COURT: What?

MR. PAIGE: This is – this testimony was required by your motion in limine. I'm happy not to put it in –

MS. HURST: But not in the opening, Your Honor.

THE COURT: Well, I apologize for not having – I just have to count on counsel to do it the way I said before. Can you skip this for now?

MR. PAIGE: I can, Your Honor.

THE COURT: Are you saying I ordered you to take this up on direct?

MR. PAIGE: You ordered that he save it, so I'm happy to move on if you don't –

THE COURT: Let's come back to it.

MR. PAIGE: Very good, Your Honor.

[1233] Q. Professor Astrachan, what might happen if you use different method declarations for classes in a given package?

A. Well, I talked before about how if the method declarations changed, then software that had already been written would no longer work, but if the method declarations changed, that wouldn't meet developer expectations, and so developers wouldn't be able to be effective in using these packages if the label declarations were all different than what they expected and had knew from what was – their development with Java.

Q. What would the programmer need to do if the method declarations were to change?

A. Well, if the method declarations changed, the programmer would have to learn the new labels by consulting documentation and reading books, and software that had already been written would have to be rewritten to use these new API declarations.

Q. Okay. Professor Astrachan, you had given an overview of your opinions earlier. I would now like to discuss some details in your opinions.

Based on the work you've done, what opinions do you have with respect to the way Google has used the API labels of these 37 packages in Android?

A. Can I use the diagram again?

Q. Of course.

Your Honor, may he approach the easel?

THE COURT: Yes, of course.

[1234] THE WITNESS: I'm going to just kind of describe at a high level what happened here.

We talked about these core libraries. The first thing Google did was they selected just the 37 packages and then you used the labels from these 37 packages in creating the Android platform.

BY MR. PAIGE:

Q. How many packages did they select those 37 from?

A. Those 37 packages were from selected from 166 packages that are part of Java SE, so the ones that were selected were the ones that would be useful to use on this mobile platform. And then after selecting those 37 packages, they then implemented them with new source code that was optimized for the mobile platform.

So because they only took the label declarations, they then had to create new implementations, again optimized for a mobile platform. That was part of how they used these declarations in creating Android.

And then once they implemented these 37, they had to add new libraries – I talked about this before – so the new libraries that would be part of the mobile Android platform. So after selecting the 37 and then implementing them, they developed new libraries that were integrated with these as part of creating the platform.

Q. And what did implementing, putting the new implementing [1235] code in do for the platform?

A. This created a new context for these other labels to be used because now they're part of a mobile smartphone platform that's different from the desktop and laptop platform that these 37 labels had been used before.

So in selecting them and then re-implementing them for mobile and then developing the new libraries, we see that these are used in the new context, a different context than they had been used before.

Q. When you say they develop new implementations, how was that tuned for mobile, if at all?

A. The new implications of these 37 packages are about 80 percent the size of the libraries that were implemented for Java SE, so we've seen that those new implementations are a smaller amount of source code than they were on the Java SE platform.

Q. The new Java library, what was the point of putting those in?

A. These new Java libraries, as I mentioned, are needed to access the functionality and purpose of this new context in which the labels are used: Cameras, accelerometers, location services. These new libraries allow these labels to be used in this new context.

Q. So in addition to the new libraries, what did Android add?

A. Well, I talked before and we can see down here that there [1236] is this Dalvik Virtual machine. Adding that to these helped make the whole platform work because this virtual machine was optimized again for a smartphone platform.

So by selecting the 37, making new implementations, developing new libraries, and then creating this new Dalvik Virtual machine, the Google engineers were able to use those declarations in this new context on the smartphone platform.

Q. How was Dalvik optimized for mobile?

A. Dalvik is optimized for mobile by first using smaller bytecodes than are used in the Java Virtual machine, and it's also designed to take advantage, as I mentioned earlier, of power constraints and memory constraints that are different on a smartphone platform than they are on a desktop or laptop.

Q. What else was added to the 37 labels?

A. Well, finally at the bottom level – and I'm not going to get down to the floor to point. I'll just reference that it's down there in red – we see the Linux kernel, and that Linux kernel is the low-level operating system that Google used because it's an open source operating system and then specialized for use on the smartphone handset.

So that's the lowest level of what Google did in creating this new context in which those 37 labels were used. So we can see that all these features went into creating the new context that shows how those 37 labels – they're 37 packages, labels from them, were used in this new context.

[1237] Q. And does the green layer enter into the context as well, Professor Astrachan?

A. Well, that green layer is kind of part of these new libraries, so some of these libraries were written in Java and some of the libraries were open source libraries that were incorporated into that section.

Q. And how, if at all, did those serve to create a new context for the 37 Java APIs?

A. Again, since we have the re-implementation, the new libraries that allowed the smartphone to function, things like a media framework and the secure sockets layer and the web kit, those hadn't been incorporated before with these 37 packages to create this smartphone platform.

Q. Thank you, Professor Astrachan.

A. You're welcome.

Q. So could you explain to the jury how, if at all, the ways in which this whole Android platform has

been used affect your opinion regarding the nature of Google's use of these APIs?

A. Yes. As I mentioned, in selecting all of these and then creating a smartphone platform and releasing it as open source, Google has created new opportunities. I can show on the screen here one of my independent– one of my exhibits that says the Kindle Fire, which is both hardware and an operating system that Amazon has developed, that's based on Android, but not the same as Android, so because Android is released as open source, [1238] Amazon was able to use it and develop Kindle Fire, which is both a device and an operating system that Amazon releases that doesn't work the same way as Android does but is built on Android.

On the right, we see a handset created by Wileyfox, which is a handset manufacturer in the UK that runs Cyanogen, which is – the CyanogenMod is an operating system, again built on Android because it's open source, but different with different functionality. Because Android is open source, the Cyanogen Company can take that and do what they want. So here are two examples that show how the open source nature of Android has created opportunities for companies to use that.

Q. How, if at all, do past attempts by Sun to create a smartphone platform figure into your opinion about Google's use of the 37 APIs?

A. We know that Sun had the 37 packages and their labels and their implementation, along with 100 other packages. Those were part of Sun's Java software product, and Sun was not able to use those to create a smartphone.

Q. To what extent, if at all, do Oracle's statements regarding Android enter into your opinions regarding Google's use of the Java APIs?

A. Well, we just saw a video right before here of Terrence Barr saying that Android was transformative, and in my opinion, that's true. The Android platform is a transformative use of [1239] these package labels from the 37 APIs.

MS. HURST: Your Honor, that was covered by the MIL, and that's a specific legal definition.

THE COURT: Well, you are talking about Mr. Barr? Is that the motion in limine you're referring to?

MS. HURST: Actually, the motion in limine for Dr. Astrachan also had limitations on his ability to characterize things as transformative, Your Honor.

MR. PAIGE: I don't think that is correct, Your Honor.

THE COURT: I honestly don't remember. We will have to take this one up at the break. So let's strike that answer for the time being. We will come back to it after the next break. Okay. Go ahead.

BY MR. PAIGE:

Q. Professor Astrachan, based on the work that you've done, have you formed any opinions about the nature of the material that Google has used from the 37 APIs?

A. Yes. I talked earlier in my kind of summary that the label declarations are functional because they connect the developer software with the software in the library, the implementing code.

So these API labels are very functional in nature. And the labels themselves are also descriptive and

functional of their purpose because that allows developers to be able to use them more effectively.

[1240] Q. And so do you have any opinions about what the names themselves of the API are in terms of their nature?

A. Sure. I can – I can show you some of the names. Certainly not all of the label declarations, but I've got a slide, it's in my slide 9, that shows some of these.

We can look at this slide first. This is a slide that shows the package names, and we can see that the packages are very descriptive of their purpose. So, for example – and all packages start with Java or Java X. Net is a collection of network classes. IO is a collection of input/output, which are called IO by programmers.

SEQUEL or SQL is a structured query language, and in that package, we'd find the classes to access database programs using SEQUEL.

Security is very important in programming today. When your program accesses your bank, for example, you want to be sure that it's a secure transaction. And in the security package, we'd find the classes that are related to security.

Java.util is a collection of utility classes that programmers use to connect different pieces of software. So here we see that the package names are highly descriptive of what their purpose is.

And then on the next slide, we see some method and class names.

Q. Being descriptive of their purpose, what does that mean [1241] about the nature of these names?

A. The names, as I mentioned, are both descriptive and functional in describing what they do.

Q. So you would say you're going to look at some method and class names. Could you explain to the jury what these tell you about the functionality of what Google has used?

A. As it turns out, in Java, methods are supposed to start with a lower case letter. So on that graphic, you see two methods that are in lower case letters, that's the top. That's get date and time. And it's probably reasonable to think that that method gets the date and time.

And then the method set date and time would allow you, the programmer, to set the date and time that are used when you write programs. So those method names, although they're not short, are highly functionally descriptive of what their purpose is.

Q. Have you looked into how many such get and set methods there are in Java?

A. Yes. There are thousands of get and set methods in Java. Get and set is a reasonably common convention used in programming to get and set different properties in a class, so there are thousands of them in these 37 API packages.

Q. How about the rest of those classes on your slide?

A. Class names in Java start with capital letters, and we can see here that the class names are also descriptive of their [1242] purpose and function. So events are things that occur in programming that you connect one part of your program to another. And there we see preference change event and connect event.

For example, if a user changes their preferences in a phone or piece of software, there might be a prefer-

ence change event. And then a connect event would be connecting one event with something else.

We see three input streams. A stream is a way for information to flow into your program. Well, an input stream has information flowing in, and that could come from a file or from a zip file, that's a zip – a compressed collection, or from an object. So we see here that these class names are also related and similar to each other in describing their function and purpose for a programmer.

Q. Again, how are these names used in Java or Android?

A. Well, the method declarations require that we have a package name and a class name and a method name. So programmers would use these class names in writing the programs that they need to run on an Android platform, for example.

Q. When they wrote that class name, what would then happen?

THE COURT: You're saying wrote it in the particular program that they themselves are writing, or do you mean when it's written as part of the library?

[1243] BY MR. PAIGE:

Q. Sure. When someone writes that name into a program they themselves are writing and compile it, what does that name then do?

A. You write the class name and the method name in the program that I'm writing over here, and that would then complied into this bytecode that runs as 0s and 1s, and it would access the implementing code in the library. So as the developer of the application, I use the package class and method name to connect my

software with the implementing code that's in the package.

Q. So what function does that name serve?

A. That name serves the function of connecting my software with the implementing code in the library.

Q. Based on your knowledge of programming languages, what similarities, if any, exist between the names used in Java and the names used in other programming languages?

A. Well, we've heard here last week, and I know from my experience teaching, that Java uses many names that are similar in, say, C++ and C. So that's part of what makes learning a new programming language more straightforward because we expect to see the same names in one language in the libraries that are associated with that language used in another language.

Q. Based on the work that you've done is it important that names for APIs be creative?

[1244] A. No. We wouldn't want names to be creative because as software developers, we'd expect to have the names in our programming libraries be descriptive and functional of their purpose. So we would want square root, for example, to mean find the square root, not some complicated, long name that wouldn't be indicative of its function and purpose.

So creative names wouldn't be helpful for a developer in finding and accessing the functionality that we'd expect to find in libraries.

Q. Professor Astrachan, as part of your work in this case, did you analyze amount of material that Google used from Java SE?

A. I did. I wrote software to analyze both the Java SE platform and the Android platform.

Q. And how many lines did you understand have been used by Google from Java SE?

A. I talked about the method and class declarations that were selected in creating Android, and there are about 11,500 declaring lines of code that were selected in creating the Android platform.

Q. Did you quantify how that Java SE material compared to the amount of source code in Java SE?

A. If we look at Java SE and the 166 Java packages that are part of Java SE, we see that in those 166 packages, there are about 2.86 million lines of code. So the 11,500 that were [1245] selected, that's about .4 percent of the implementing code for all 166 packages, and if you look at all of Java SE, that's about five million lines of code.

Q. How many lines are code are there of the Android platform generally?

A. The software I wrote indicates that there are about 15 million lines of code in the Android platform.

Q. Professor Astrachan, based on the work you've done, have you formed any opinions about the effect that the release of Android has had on Java?

A. Yes. As I mentioned earlier, Java is the number one programming language in use. That's what I know from my own work and what we see on Oracle's website. So I think that's a good indication of part of the development environment we see today.

Q. And what is part of that development environment that has Java still remaining number one?

A. We have the Android platform so that programmers can develop in Java for the Android platform, and we have the Java SE platform so programmers can continue to develop with Java for that desktop and laptop platform as well.

Q. What is OpenJDK?

A. OpenJDK is an open source implementation of Java SE that Sun Oracle has released.

Q. And how did it release the OpenJDK?

* * *

[1262] THE COURT: Please continue.

BY MR. PAIGE:

Q. Professor Astrachan, do you have an opinion about what developers would expect in terms of the Java API packages availability?

A. I do. I think developers, just like my students, would expect that if you're going to be using the Java programming language, that you have access to a rich suite of APIs, both the declarations and the libraries, to be able to write the programs that you would be writing for whatever platform that would be.

Q. What does that mean for the ability to make effective use of the language?

A. In general, programs do complicated things. They might open a web page or print something or connect with a user in a touch screen. All those things require libraries because developers couldn't do them from scratch.

So the effective use would be depending on the purpose of your program to write it effectively I need libraries to be able to use the language.

Q. How do that relate to the selection of the 37 out of the 166?

A. I spoke earlier about selecting the 37 packages and using just those label declarations, and in incorporating that into the Android mobile smartphone platform, we saw that developers [1263] would expect to see both the implementations of those 37 packages and the other libraries that I spoke of to be able to make applications for that platform.

Q. Thank you, Professor Astrachan.

May we mark as his demonstrative as Exhibit 7793, Your Honor.

THE COURT: Sure.

(Trial Exhibit 7793 marked for identification)

THE COURT: Give me the number again. Seven what?

MR. PAIGE: 7793.

THE COURT: All right. That will be the demonstrative. All right. Okay.

Ms. Hurst, are you ready?

MS. HURST: I'm ready, Your Honor.

THE COURT: Please proceed.

CROSS-EXAMINATION

BY MS. HURST:

Q. Good morning, Dr. Astrachan.

A. Good morning.

Q. Let's talk about some terminology first. When you say "specification," you mean the – both the API

declaration and the text that describes how to use it; is that correct?

A. I think that's correct. The text meaning what's typically the comment or what you'd find along with the declaration to understand how to use it, that's right.

* * *

[1322] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOGGLE, INC.,

Defendant.

San Francisco, California
Tuesday, May 17, 2016

Before the Honorable William H. Alsup

TRANSCRIPT OF PROCEEDINGS

---

* * *

[1351] BY MS. HURST:

Q. Ms. Catz, we looked yesterday at the email that Mr. Schwartz sent that mentioned battles with Google

Android. Can you just remind the jury what Mr. Schwartz told you about that?

MS. ANDERSON: Objection. Hearsay again, Your Honor.

THE COURT: Was this already testified to?

MS. HURST: Yes. It was offered for the limited purpose that the Court gave for the instruction yesterday to respond to Mr. Schwartz.

THE COURT: Is the answer going to be the same?

MS. HURST: Yes.

THE COURT: All right. Go ahead and repeat the answer.

Objection overruled. It will be received for this limited purpose.

THE WITNESS: Mr. Schwartz told me it was an unauthorized unlicensed fork of Java SE.

BY MS. HURST:

Q. Ms. Catz, since acquiring Sun, what has Oracle done with respect to the Java platform?

A. We've invested a lot in the Java platform. We've hired engineers. We've had hundreds of engineers working on it to continue to enhance it. We've actually had two new versions, Java 7 and Java 8. Both versions have come out since we took [1352] over.

We have – we have really expanded the entire educational network of Java. We teach Java to high school teachers. We share Java with universities, not only in the United States, but around the world, so we work very extensively through the Oracle Academy teaching Java.

We market Java. We use Java and we support the big show JavaOne which is now even larger than it's ever been, which is a big show for developers, for Java developers, to meet and work, and of course we run the JCP, the Java Community Process.

Q. And can you approximate for the jury how much Oracle has spent in all of those efforts?

A. Hundreds of millions of dollars.

Q. Ms. Catz, how would you characterize the significance, if any, of intellectual property protection to Oracle?

MS. ANDERSON: Objection, Your Honor, to the extent it calls for a legal opinion.

MS. HURST: It's for the witness' understanding and her business, Your Honor.

THE COURT: Well, maybe it's a problem, but it's a vague question. I'll let her answer, and we'll see how problematic the answer is.

Stay away from legal things, but otherwise, you may answer, Ms. Catz.

THE WITNESS: All right. Thank you.

* * *

[1572] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOGGLE, INC.,

Defendant.

San Francisco, California
Wednesday, May 18, 2016

Before the Honorable William H. Alsup

TRANSCRIPT OF PROCEEDINGS

---

* * *

[1664] Q. And then did you have business dealings with the original equipment manufacturer, the OEMs?

A. Once the manufacturers understood that to sell their products they had to have this capability, they would come to Sun to get a license for the technology.

Q. Let's talk a little bit about your second reason, the developer community. Can you tell me what you meant by that?

A. So Java began as a desktop and a enterprise technology. And by 2000, there were several million

Java – Java developers. And they quickly were able to learn to develop for the Java ME platform as well.

Q. And the third reason you mentioned, I believe, was security reasons. Can you explain that again. What was the security issue in the 2006 timeframe?

A. Yeah. In this time period, cell phones were built with proprietary operating systems from the device manufacturers. And these operating systems were relatively weak in security.

So if you wanted to add downloadable applications to devices at that point – which the carriers very much did – those operating systems would have been exposed to, you know, being hacked in the same way that many desktop systems have been over the years by things like viruses, rogue applications that would be built to attack the device.

And carriers were concerned that that would subject their networks to risk or that the applications the phones were [1665] carrying would be tampered with or the data lost.

So Java would give them a way to support downloadable apps but prevent those things from happening by using a technique called managed code. Using that technique, the applications would be prevented from using sensitive operating system functions like access to the network. And they would be prevented from accessing the data held by other apps. So it was a very effective security solution for the situation at the time.

Q. So as of the 2006 timeframe, when you were– just before you left Sun, what was your view, if any, as to whether Sun's efforts to market Java for mobile devices had been successful?

A. Well, it had been extraordinarily successful. We were supported by hundreds of carriers. And we were adopted in, you know, as I said, about 80 percent of the devices that were shipping at the time.

So the approach worked. And the technology was a very good fit for the value proposition downloadable apps that we were trying to market.

Q. By 2006, which carriers were requiring Java in their phones?

A. Well, some examples would be Sprint, AT&T, T-Mobile, in the U.S. Vodafone Orange. Telefónica Europe. NTT DoCoMo. It's the largest carrier in Japan. And in Korea, KT was another. So they were all over the world. All of the major [1666] carriers or most of the major carriers.

Q. And which manufacturers sold Java-powered phones in 2006?

A. Again, the bulk of the industry. So, for example, Nokia; BlackBerry; Samsung LG. Danger was another. Panasonic. Yeah. Long list.

Q. Are any of those manufacturers still selling Java-based phones in 2016?

A. I think most are not, at this point.

Q. I want to talk specifically about some of the phones that were on the market just before you left Sun, again in the end of 2006 time frame.

Did you use the term "smartphone" and "feature phone" when you were at Sun in that time frame?

A. I did.

Q. And what was your understanding of the difference, if any, between smartphones and feature phones?

A. Well, in that time frame, the differences would be different than they are today because the smartphone category, at that point, was just emerging. And, in fact, sort of going through a change.

In that time frame, it was more or less a continuum. Smartphones, at that point, often had larger screens, higher resolution screens, color capability. They often had keyboards, QWERTY keyboards. Whereas, a feature phone would have a 9-key keypad, like you might see on a conventional [1667] telephone, and less screen capability, less memory.

Both smartphones and feature phones in this period, though, would have had network capability and downloadable applications.

Q. So with that continuum in mind, if we look specifically at 2006, what percentage of feature phones were powered by Java at that time?

A. I would say about 80 percent of the phones, feature phones in that time, were Java-enabled.

Q. And looking, again, at that specific time frame in 2006, what percentage of smartphones were Java-powered at that time?

A. Nearly a hundred percent at that point.

Q. And which manufacturers were making smartphones at that time?

A. Uhm, well, Nokia would be one. The Series 60 devices and Series 80 devices. All of the BlackBerry products were Java-powered. But others had as well. Samsung LG, Panasonic. Again, quite a long list. Sony Ericsson was another. So several.

Q. In the 2006 time frame, what was your view, if any, regarding Sun's ability to capitalize on its success

and continue in its dominant market share with respect to the mobile industry going forward?

A. I thought we were really well-positioned in light of our presence in the industry, the fact that the industry was fully [1668] licensed. Carriers had adopted the value proposition of downloadable apps. And we were, you know– we had a very large developer community. So we had a number of assets that we thought gave us a chance, a good chance to be a strong player in the smartphone space.

Q. What is Java ME, Mr. Brenner?

A. Java ME is an application development platform for mobile devices, cell phones, and in that time frame pagers. Connected – connected mobile devices. You use it to build an application that runs on the phone and executes securely.

Q. Were you involved in the initial creation of ME?

A. Yeah. I led the initial creation of ME.

Q. Can you tell me how ME was created?

A. I had a mandate to develop a Java platform for mobile devices. And in that time frame I discovered or my team discovered a research project in Sun labs. Two researchers had developed a lightweight Java implementation for the Palm Pilot, which my team took over and commercialized in September of 2000.

Q. And how did you go about developing the ME software?

A. We selected classes from Java SE. Java ME was essentially derived from Java SE and complemented or added to that mobile-specific classes that would fill out the API of the device for mobile applications.

Q. And during your time at Sun what, if any, changes did Sun

* * *

[1691] (Videotaped testimony played as follows:

"Q. You suspected that they would independently implement the Java Class library APIs, or some subset of those; correct?

"A. Yes.")

BY MR. PAIGE

Q. And that belief was quite generally held by many people at Sun at the time; right?

A. By the people who are involved – by several people who are involved in the negotiations with Google. I would say that's true.

Q. Okay. Mr. Brenner, you're not a lawyer; correct?

A. I'm not.

Q. And when you talk about something being derived from Java SE, you weren't offering an opinion about whether it is a derivative work, were you?

A. I was speaking to the origins or the sources of the technology and the content of the products.

Q. You weren't offering any legal opinions on a derivative work, were you, sir?

A. I can't offer a legal opinion, no.

Q. Okay. Now, you said that there was code from Java SE put into Java ME later on. Is that your testimony?

A. My testimony is that the API – APIs from Java SE were applied to Java ME. In some cases they – we

leveraged code. [1692] In other cases we implemented a consistent API spec.

Q. How much code was put into Java ME from Java SE, sir?

A. I can't recall enough to quantify that.

Q. Can you give any idea as to how much was moved in?

A. Well, my guess – my recollection would be there were at least several dozen APIs that spanned the two editions at the time. But I couldn't quantify beyond that.

Q. How many APIs did Java ME have, sir?

A. I don't recall the number, but –

Q. How many APIs did Java SE have?

A. I don't recall the number.

Q. Is it possible there were several dozen APIs added to Java ME?

A. Well, the word "API," of course, would – you know, there are classes and methods. There were certainly several dozen methods. Probably quite a bit more than that in Java ME.

Q. Now, your job responsibilities at Sun, they involved licensing Java; correct?

A. Uhm, I was responsible for the business and the P&L. And the sales team would draft and negotiate the licenses. But I would often support them for product purposes.

Q. You don't recall ever offering to license just the Java API specifically to any company; correct?

A. I don't.

Q. And during the time you were with Sun, you don't recall

* * *

[1698] Q. Move the admission of 7238, Your Honor.

THE COURT: Any objection?

MS. SIMPSON: No objection.

THE COURT: Received.

(Trial Exhibit 7238 received in evidence.)

(Document displayed.)

BY MR. PAIGE

Q. Now, you wrote presentations, before you left Sun at the end of 2006, saying that Java ME was fragmented; correct?

A. I did.

Q. And, again, 2006, that was a year before Android was even announced; right?

A. Yes, it was.

Q. Look at page 13 of the document.

You wrote there that "Fragmentation undermines Java value proposition in the market"; right?

A. Right. That's what the title of the slide says.

Q. And that phrase refers to when a program written in Java ME might not be working as well on one OEM's implementation as another; right?

A. It does. Working differently.

Q. And, again, these problems all existed before Android was announced; correct?

A. That's correct.

MR. PAIGE: I pass the witness, Your Honor.

[1699] THE COURT: Wait. Wait. Before you leave that, the jury has heard the word "forked." Forked. F-o-r-k-e-d, forked. Is that the same thing as fragmentation? Or are they different things?

THE WITNESS: Not as it's used in this context.

What this slide is talking about is that each manufacturer had slightly different implementations. They– each of the phones had, typically, a proprietary operating system. And the port – ports of the technology were done by different entities. And so they were not bug-for-bug compatible, which impacted developer costs according to the actual devices, and changed behavior for the user. So we were concerned about that variation in the industry.

THE COURT: What you're talking about there was fragmentation. And that's what you were concerned about. But forked is something else?

THE WITNESS: Forked is something else.

THE COURT: Okay. Thank you.

All right. Let's go back to Ms. Simpson.

REDIRECT EXAMINATION

BY MS. SIMPSON

Q. Mr. Brenner, in connection with Trial Exhibit 7237, the revenue slide –

A. Right.

Q. – you had mentioned that you were making that forecast,

* * *

[1704] context – are you asking about the negotiations with Google or subsequent?

Q. I am not. I'm asking about what you said you would have done to avoid the revenue drops that you showed in your presentation.

One of the things you were trying to do in the 2005/2006 time frame, you led a project to try to create a mobile platform that used more than 75 percent of the API packages in Java SE?

A. That's true, yes.

Q. And that project was never completed, as far as you know; right? It was still unfunded when you left Sun; correct?

A. Work had been done, but, no, I don't believe that project received the funding we were looking for.

Q. And, in fact, rather than continue with that project, Sun went and bought SavaJe instead; correct?

A. That's correct, yes.

MR. PAIGE: Nothing further.

THE COURT: May the witness be excused?

MS. SIMPSON: Yes, Your Honor.

THE COURT: Thank you, sir. You may step down. I'm going to discharge him from the subpoena, unless I hear an objection.

MR. PAIGE: No, Your Honor.

THE COURT: You are discharged from the subpoena.

* * *

[1750] A. Right. So what they do do is they charge the device makers. So if you want to put Java on your phone or you want to embed it in the operating system

for a phone, Sun and then Oracle will offer you a license to do that, and they charge you what's called a royalty, which is just the fees that you pay on the license. They charge you a fee for doing that. And that is the primary mechanism by which Java makes money, which is – monetization is just a fancy word for making money.

Q. In this platform market, what might be the role of open source software?

A. So, again, I'm not talking from a legal perspective, but from an economic perspective, because of the platform nature of this market, it frequently makes sense for the owner of the platform to say, you know what? I'm going to make this available in an open way with a license, an open source license so that people can work with it, they can improve it, they can experiment with it, but I'm going to do that with a license that I think doesn't make it suitable for the device makers to use in a commercial product. So that what I do is I set up that open source license in a way that it's not designed to substitute for the royalty-bearing license that the device makers take to make money from me.

Q. You said that having a large community of developers was important. How popular was Java to develop?

A. So I think we've heard that in testimony in this trial. [1751] In the mid 2000s, there was something like six million developers writing applications for Java.

Q. What kind of devices were Java app developers building programs for?

A. So Java was on a whole bunch of devices: Automobiles, desktop computers, then phones, Blue-ray devices, set-top boxes and so forth. And as you can see here, as devices evolved and new devices emerged, Java evolved and found its way onto these new devices as they came about.

Q. And what was Java's position in the phone market in the mid 2000s?

A. So we've heard some testimony about this as well. But in the mid 2000s, Java was doing very well in the phone market. I won't go through all of this, but basically what it says, this is a Sun document from the mid 2000s. And what it says is there were a billion phones that had Java in them; 600 different models; almost 200 carrier deployments; 600 million phones sold just in 2005. And you notice the green arrows here, all of these things were – were growing at that time.

Q. And let's talk a little bit about Google. Why is it important to understand how Google makes money?

A. So one of the issues that we're going to talk about, because I understand it to be related to fair use, is just how commercial was this use, and if we're going to try to understand how commercial was the use, we need to understand

* * *

[1781] A. Yes.

Q. That's part of the platform?

A. It's part of the Android platform, yes.

Q. Right. And there are more than a hundred libraries, Android libraries, that were written by Google engineers as part of the Android platform?

A. That is my understanding.

Q. And all these elements have value, don't they, Dr. Jaffe?

A. They do.

Q. And – but you haven't bothered to sort out value as between the 11,000 lines of method declarations and the 15 million lines of code in Android; right?

A. It wasn't necessary for my opinion to sort that out, so I did not do it.

Q. And, therefore, you didn't do it?

A. I did not do it, no.

Q. And you understand that the method declarations that are at issue in this trial represent less than one-tenth of 1 percent of the lines of code in Android; right?

A. That arithmetic sounds right.

Q. Now, in the mid 2000s, consumers were in the process of transforming their digital behavior; right?

A. I think that's generally fair.

Q. In fact, in your report you said precisely that, that in the mid 2000s, consumers were in the process of transforming [1782] their digital behavior?

A. Uh-huh.

Q. Right?

A. Yes.

Q. And the mobile application platform space is what you call a dynamic ecosystem; right?

A. Yes.

Q. It's a market that could be characterized as highly volatile; right?

A. Yes.

Q. One that moves up and down quickly. Isn't that what "volatile" means?

A. Well, I don't think it means the market moves up and down quickly.

Q. All right.

A. When I say it was volatile, what I meant was there was a lot of uncertainty about the place in the market of particular phones and particular participants.

Q. Fair enough.

But, as a result of that, the mobile platform market we're talking about is extremely challenging to model and predict; right?

A. Yes.

Q. And that's particularly true – that's particularly true when market outcomes are highly uncertain, as they were at the [1783] time of Android's launch; right?

A. Yes.

Q. I think you said on direct, platforms often fail; right?

A. Yes.

Q. And, in your view, predicting success in platform markets with certainty is nearly impossible?

A. Yes.

Q. "Nearly impossible." That's what you said; right?

A. Predicting with certainty is nearly impossible, yes.

Q. And, yet, it's your conclusion that without Android, Java was poised for great success; right?

A. Poised, yes.

Q. Poised for great success. Isn't that what you told the jurors?

A. That is what I said, yes.

Q. You also told our jurors that there was a market window of opportunity; right?

A. Yes.

Q. Even though you acknowledge you can't give us a starting point or an ending point for that window?

A. I explained why that's irrelevant, yes.

Q. And, in fact, do you have any idea when the window of this market opportunity opened, Dr. Jaffe?

I'm standing over here at the timeline to give you some help.

[1784] Can you tell us with any sort of precision when the window opened?

A. Well, I think if we're talking specifically about the window, which is the way I talked about it, for Android to get into and succeed in the –

Q. Excuse me, Dr. Jaffe. I'm not limiting my question to Android.

You have said you can't determine with any degree of precision when the market opportunity for smartphones opened; right?

A. Well, I talked about window of opportunity only with respect to Android.

Q. Well, it's an opportunity for anyone that wants to get there; right?

A. But the nature of that window is different for different players.

Q. Okay.

A. Because they come to it with different assets and different liabilities.

Q. All right. And, certainly, given your views that this is a volatile market that's almost impossible to predict, you'd agree it's almost impossible to predict when the window closes too; right?

A. Yes.

Q. Now, you've said that it was a feat, a feat for Google to [1785] have established Android as a new, viable mobile application platform; right?

A. Yes.

Q. And that's a feat that many other very sophisticated tech companies failed to achieve; right?

A. Yes.

Q. Microsoft failed to achieve it?

A. Well, they still have a phone. It's not very successful.

Q. Facebook failed to achieve it?

A. Yes.

Q. Amazon failed to achieve it?

A. Yes.

Q. And Sun failed to achieve it too?

A. Yes.

Q. And so far Oracle has failed to achieve it as well; right, Dr. Jaffe?

A. Yes.

Q. Sun invented Java; right?

A. Yes.

Q. And they were the experts in Java starting in the early '90s; right?

A. Yes.

Q. And yet they failed to achieve this feat of building a full stack smartphone platform; right?

A. Well, I don't know what the "and yet" is about there.

* * *

[1799] A. Well, the history that we see has Android in it. And with that –

Q. Excuse me. Excuse me. That's not an answer to the question.

MR. BICKS: Your Honor, he can't answer the question without being interrupted.

THE COURT: He was about to slide off and go in a different direction.

So you've got to answer – he's entitled to an answer to his question. And then you can give an explanation.

So ask the question again. Then say yes or no and give an explanation.

THE WITNESS: Okay. Fair enough.

So in the world that we see, they did not further develop ME. And SE has not been successful in smart-

phones. But all of those decisions were affected by the entry and success of Android.

BY MR. VAN NEST

Q. Well, let's – let's test that, Dr. Jaffe. When did the SavaJe phone fail?

A. In the mid 2000s.

Q. Yeah. It failed before Android was even announced; right?

A. I don't remember specifically.

Q. And do you recall a project at Sun called Acadia?

A. Yes.

[1800] Q. And that failed too; right?

A. It was abandoned, yes.

Q. It was abandoned.

And do you recall a project called Daneel? "Sundroid" they called it.

A. Yes.

Q. That failed too?

A. It was abandoned, yes, is my understanding.

Q. And you saw last week that Mr. Ellison looked at a Java phone project and concluded, at least according to the slides that he prepared, that Oracle had too little expertise to build a smartphone; right?

A. I did see that, yes.

Q. That slide deck that Mr. Ellison prepared and that we examined him on, that didn't say anything about Android?

A. It didn't, no.

MR. VAN NEST: Could I have up the chart that– it's TX 5397. It's one of – it's the – that's the one.

(Document displayed.)

MR. VAN NEST: That's the one.

BY MR. VAN NEST

Q. Now, you showed this as – as evidence that Android somehow substitutes or caused harm to Java; is that the idea of this?

A. Yes.

[1801] Q. Okay. And I think you said this is only smartphones?

A. That's correct.

Q. So there was a time when Java had 80 percent of a smartphone market?

A. That's correct.

Q. Was that with the SavaJe, or what?

A. I believe it was BlackBerry and other – also some Simion-based phones.

This exhibit is not based on my deciding what's a smartphone and what's not. It was taken from an industry source that people frequently use. And it was Gardner himself who characterized the phones as smartphones versus feature phones.

Q. Okay. So the industry people, just like Sun, they also distinguish between the feature phone and the smartphone market; right?

A. I said in my testimony that various parties throughout this period, at points in time, did make that distinction, yes.

Q. Okay. And this was – this was from a well-respected industry source; they were distinguishing between smartphones and feature phones?

A. That's correct.

Q. They considered them different markets?

A. I wouldn't say they considered them different markets. I would say they considered it useful, for market analytical [1802] purposes, to look at the two groups and their relationship.

Q. I see.

But just like the Sun people considered smartphones and feature phones as different; right?

A. I don't know if it was "just like" or not.

Many people, at many points in time, found it useful to have that categorization.

Q. Now, you were aware, Dr. Jaffe, that people inside Sun were predicting exactly that decline for Java, before Android was even announced; right?

A. Their projections did not go through 2015.

Q. Let's take a look at TX 7237. And this is a – this is one of the slide decks that you had access to, Dr. Jaffe; correct?

A. That's correct.

Q. It was written in – this is important. September 29th, 2006.

According to this, that's a year, more than, before Android was announced; right?

A. That's correct.

Q. A full year.

And, by the way, once it was announced, that didn't mean there was a phone on the market, did it?

A. It took about another year.

Q. Another year. And that phone, HTC Dream, it didn't take [1803] off as a barn burner either; right?

A. Not immediately, no.

Q. Not as bad as SavaJe, but it sure didn't skyrocket; right?

A. That's my understanding.

Q. It wasn't until later, 2010, when Motorola came out with Droid, that Android took off; right?

A. Yeah, I think that's what my exhibit showed.

MR. VAN NEST: Now, could we go to page 3 of this.

(Document displayed.)

MR. VAN NEST: Let's go to page 4. I think there's a chart here. Do I have the wrong one?

You're right. Let's go back. Let's go back.

BY MR. VAN NEST

Q. So this is folks at Sun, a year before Android came out, predicting that the market was changing; right?

A. Yes.

Q. "Increase in device capability. . ."

MR. VAN NEST: Could we underline that?

BY MR. VAN NEST

Q. ". . . and networking, shifting the market to advanced platforms. Growth is more than 5x feature phones."

Right?

A. That's what it says.

Q. So before Android even came along, the folks at Sun were aware there was a threat out there that the market was [1804] changing; correct?

A. Yes.

Q. And – and you said that too. You are said consumers were transforming their behavior; right?

A. Yes.

Q. All right.

MR. VAN NEST: Now, could we go to page 5.

(Document displayed.)

BY MR. VAN NEST

Q. This looks a lot like your chart that you showed the jury on direct, doesn't it, in terms of the line for Java?

A. Well, it doesn't really. It goes through 2010. And it doesn't show the revenue going to zero.

Q. Well, let's look at what it does show. By the way, this is Java ME; correct?

A. Yes.

Q. And Java ME is what Sun had in feature phones; right?

A. Yes.

Q. And Sun recognized that the world was changing and that there was a new market emerging for smartphones. Or let's call them "more powerful phones," okay. Right?

A. Okay.

Q. And what they said was – and this is people at Sun in 2006; right –

A. Yes.

[1805] Q. – Dr. Jaffe? Not created for litigation, but created to plan their business?

A. That's my understanding.

Q. Okay. They were telling their management that, if we stay the course, revenue is going to go from 140 million down to less than 60, maybe down to 50 million potentially; right?

A. Under one scenario, yes.

Q. Right. And the other scenario, it drops – it still drops, but not so bad?

A. Right. It remains above a hundred million.

Q. So people at Sun – and this is all before Android was – had even been announced in a phone; right?

A. Yes. We've said that several times.

Q. And a couple of years before a phone came on the market? An Android phone.

A. That is correct.

Q. Now, did you make any investigation to determine whether or not Sun did anything to avoid just staying the course?

A. Uhm, I don't think I did an investigation framed in those terms, no.

Q. Okay. So you can't tell our jury whether Sun did or did not do anything to avoid what they saw as inevitable if they didn't change what they were doing at the time; right?

A. Not specifically with respect to this memo.

THE COURT: All right. We're at 1:00 o'clock.

* * *

[1810] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOGGLE, INC.,

Defendant.

San Francisco, California
Thursday, May 19, 2016

Before the Honorable William H. Alsup

TRANSCRIPT OF PROCEEDINGS

---

* * *

[1842] wasn't intending to build a search engine at all. We were just looking at the web and the way the pages linked to other pages, and so we actually downloaded a large portion of the web, which was fairly small at the time. This was '98 or so, something – a little before '98, and we realized we had a good way of ranking web pages using just the links of the web.

Q. How did you get the money to start the company?

MR. BICKS: Your Honor, I would object. This is beyond the scope of my cross.

THE COURT: True.

MR. VAN NEST: Very brief background, Your Honor.

MR. BICKS: We had two Google witnesses –

THE COURT: I think that – I think this is beyond the scope of the cross.

MR. VAN NEST: Okay, Your Honor.

Q. Mr. Page, why did – let's turn the clock forward to 2005. We have a timeline here. I hope you can see it.

2005, Google acquires Android. Why did Google acquire Android?

A. Well, I think we were – you know, I was super frustrated with the state of phones at the time, you know, many of which were running Java. But they didn't really work very well. You couldn't even like take a picture and share it with someone.

We actually – I remember visiting – we had a closet literally full of like a hundred phones so that we could test [1843] them, and they all worked differently, and basically we couldn't get our software to work on those phones. So it was tremendously frustrating.

Q. What software were you hoping to have work on the phones?

MR. BICKS: Your Honor, again, this is beyond the scope of my cross.

MR. VAN NEST: This has to do with Android, Your Honor, and why they got it.

THE COURT: Stick to Android, and Android's within the scope.

MR. VAN NEST: Thank you.

THE COURT: But some of this background is beyond the scope.

BY MR. VAN NEST:

Q. Mr. Page, what was the software you were trying to get on the phones?

A. You know, we had basic Google Search, which I think is how we make most of our money, you know, which is available to you on any phone, but we also like to improve it so we add things like voice. We like having things that work really well and having a platform that actually works reliably, you know, that you can use to use our services is really important, and so I think that was the main reason why we did Android.

Q. You mentioned during your direct examination that Android is open source. Was it always the idea to make Android [1844] available as an open source project?

A. Yeah. I think from the very early days, that was–that was the intention.

Q. Why was that important?

A. I think that we were generally aligned around having really wide distribution and also because, like I said, we make most of our money from Google Search. We want people to be able to access that, you know, even if they don't have that much money or have basically very wide distribution. That's always been our philosophy.

Q. You mentioned that it took five years to get Android out on the market. Can you tell the jurors why it took so long?

MR. BICKS: Again, Your Honor, this is beyond the scope of my cross.

THE COURT: The thing about five years did come up on your direct. Overruled.

THE WITNESS: I think even longer than five years. I mean, the company we acquired also had been working on Android for a while. People had also been working on phones for a while and built other phones and so on.

So I think as much as I would like things to not take very long, it takes a long time to make something good. I think the iPhone took a similar amount of time. So in order to make a product that's really transformative and that is much better than the ones before, it did take a significant amount of time [1845] and effort.

THE COURT: All right. This has turned into a speech about transformative. Objection sustained. Okay. Come on.

BY MR. VAN NEST:

Q. Did you ever believe there was some window in which you had to get Android out on the market?

A. No. I think there's always moving targets and we always have different options available to us and we have very clever engineers and product people.

Q. Now, you mentioned – you were asked on direct examination about discussions with Sun. Were you kept advised about discussions between Google and Sun? Let's focus on these early days in August 2005 through May of 2006. Were you generally kept aware of the progress of discussions with Sun?

A. Yeah. I think we were briefed from time to time.

Q. What was Google looking for in a relationship with Sun at that time?

A. I think we were, you know, intending to use their technology, the implementation of Java, and their proprietary technology to put into Android.

Q. Would you have needed a license for all that?

A. Yes.

MR. BICKS: Objection, Your Honor. Calling for a legal conclusion.

THE COURT: Well, that's true. But you could ask the [1846] question *were you seeking a license*. That would be all right.

MR. VAN NEST: Sure.

Q. Were you seeking a license as part of those discussions?

A. Yes. And I think a broader deal around other things, you know, branding and cooperation and so on.

Q. How would a partnership with Sun have benefited Google, if at all?

A. Well, I think, you know, they spent a lot of time on the implementation of that code and that may have been useful to us.

Q. Now, after discussions with Sun broke off, did you believe that Google needed a license to use the APIs in Java?

A. No, I did not believe that.

Q. And tell the jury why.

A. The declarations of the APIs, yes.

Q. Okay. The declarations. Tell the jury why you felt that way.

A. I think it was established industry practice that the API, just the headers of those things, could be taken and basically reimplemented very carefully, not to use any of the existing implementation of those systems. That's been done many, many times.

Q. Okay. And you're familiar with that practice?

A. Yes.

Q. Is it common in the industry?

[1847] A. Yes. I think it's very common in the industry.

Q. Has it been going on for years?

A. Yeah. I think that's how you–

MR. BICKS: Objection. No foundation. This is now into an area where the Court has already cautioned.

MR. VAN NEST: Well, Your Honor, his state of –

MR. BICKS: He is not disclosed as a witness on custom and usage.

MR. VAN NEST: His state of mind. He was asked a lot of questions about licenses and so on on his cross, Your Honor. He has got a right to explain what the company did.

THE COURT: Well, Mr. Bicks, you did accuse him of willful conduct, basically, on your direct examination through the questions that you asked, and he's entitled to give his response on – so I think this is – this is legitimate within the scope of the examination.

Overruled. Please go ahead.

BY MR. VAN NEST:

Q. Go ahead, Mr. Page.

THE COURT: But you have to stick to what was in your mind at the time all this was going on and not veer off into a speech.

THE WITNESS: Okay. I think we acted very responsibly and carefully around these intellectual property issues.

[1848] BY MR. VAN NEST:

Q. And what was your understanding about the Java API method declarations?

A. They were free and open, and even after that fact, we know Sun publicly supported our use of Java and Android.

MR. VAN NEST: I have nothing further, Your Honor.

THE COURT: Anything more?

MR. BICKS: No, Your Honor.

THE COURT: All right. May the witness be excused and discharged?

MR. BICKS: Yes.

THE WITNESS: Thank you, Your Honor.

THE COURT: Thank you, Mr. Page, you are free to go.

MS. HURST: Your Honor, we have some read-ins before we recall Dr. Java.

THE COURT: Okay.

MS. HURST: It will go quick.

THE COURT: All right.

MS. HURST: "Request for Admission No. 285 from Oracle to Google: Admit that code in Android is not derived from code from the GNU Classpath project.

"Response: Google admits that its source code implementation for the Android core libraries is not derived from the GNU Classpath project.

"Request for Admission No. 287 from Oracle to Google:

* * *

[1851] Q. And there are many, many internal documents at Sun back in '05 and '06 reflecting the fact that they knew the market for mobile phones was changing; right?

A. I don't know that there were many. We have seen some.

Q. There certainly were some that you came upon in your review of the files and records in this case; right?

A. Yes.

Q. Could we have TX 7237 on the screen, please. And let's start with the cover page.

We did look at this yesterday, Your Honor. It's already in evidence. Just to establish the date again, it's September 29th, 2006.

Do you see Mr. Brenner's name on the cover of the slide deck?

A. I do.

Q. He was here yesterday to testify. That's the same person, isn't it?

A. That's my understanding.

Q. Okay. So let's go to page 3. And "Market Changes Threaten Our Position." That's the title of this slide deck. This is one of the documents that reflect that Sun was fully aware that changes were afoot for smartphones – for mobile phones, excuse me; right?

A. I'm not sure I understand the question. This is a document that says what it says.

[1852] Q. Okay. Let's go down to the third bullet. "Increase in device capability and networking shifting the market to advanced platforms. Growth is more than 5X feature phones."

You understood that as a reference to the fact that folks at Sun back in '06 knew that increase in device capability and networking is shifting to more advanced phone platforms; right?

A. You read it correctly.

Q. And that was your understanding from all the documents you have read, too?

A. That part, yes.

Q. Okay. And it also says below that "New competitors entering the advanced market. No entrenched incumbent"; right?

A. It says that.

Q. That's consistent with your view that this market was highly dynamic, changing, and very hard to predict; right?

A. Yes.

Q. Okay. Now, could we go – I think we saw yesterday, but let's go down to page 5.

As a result of the changes that Sun saw coming, they were predicting a potentially big drop in their ME revenue; right?

A. They made a number of different predictions, and as Mr. Brenner explained, they anticipated that under some circumstances, they could have this kind of revenue drop, yes.

Q. And it's – this is '06, so this is before Android was announced. You've seen documents in '07 and '09 and 2011 where

* * *

[1858] A. Yes.

Q. And you testified that at least one of them, Samsung, lost revenue – excuse me – that Oracle lost revenue because Samsung didn't renew its license at as big amount as it had; right?

A. Yes.

Q. And the reason for that is that Samsung wanted to build smartphones like Galaxy, not feature phones; right?

A. I would agree that they wanted to build smartphones. I would not agree that that was necessarily the reason why they were not licensing from Oracle.

Q. Well, you didn't do any investigation to find out exactly why Samsung did what they did; right?

A. That's true.

Q. But don't you assume, as an economist knowing what you know about this market, that the reason Samsung didn't re-up on Java ME is they wanted to build a smartphone and Java ME isn't capable of supporting a smartphone; right?

A. I don't agree with everything in what you just said.

Q. Okay. Let's unpack it. I asked two questions.

So we've established that the folks at Sun and Oracle have acknowledged that ME doesn't support a modern smartphone; right? We established that yesterday?

A. Yes.

Q. That's because ME was intended for resource-constrained [1859] devices much smaller and much simpler than a modern smartphone; right?

A. That's my understanding.

Q. Okay. And the reason that Samsung did not re-up at as high a number is they want to build smartphones like the Galaxy; right?

A. I agree that they want to build smartphones.

Q. Okay. And the same is true for HTC and Sony, they want to build smartphones, too?

A. Yes.

Q. Now, you're offering an opinion on market harm, and as we discussed yesterday, the test you're applying is whether or not there was market harm to the copyrighted work or derivatives of that; correct?

A. Yes.

Q. Okay. The copyrighted work in this case is Java SE?

A. Yes.

Q. And I didn't hear any testimony from you yesterday about any loss of revenues in the Java SE line of Oracle's business, did I?

A. Well, honestly, it's in my report. I don't remember as the direct point yesterday whether we covered that or not.

Q. Now, Java SE, the Standard – SE stands for Standard Edition; right?

A. Yes.

[1860] Q. It's traditional mark was desktops and servers; correct?

A. I think that's fair.

Q. And for years as Java SE was introduced and sold, it was sold primarily for use in desktops and servers and fairly powerful laptops; right?

A. That's my understanding.

Q. Okay. And as far as you know, that part of Oracle's business – by that I mean Java SE – is doing just fine; right?

A. Oracle continues to license SE in those markets, yes.

Q. And as a matter of fact, it's doing better than ever?

A. I don't recall.

Q. Did you look to see whether Java SE revenues are going up or going down as part of your work?

A. Well, as I testified yesterday, in considering market harm, I looked at the markets of mobile – mobile phones, tablets and E-readers and other device categories that I mentioned yesterday.

Q. But that wasn't my question, Dr. Jaffe. My question was did you look to see whether Java SE revenues were going up or going flat or going down?

A. I looked at that within the markets that I examined, which you're right, did not include desktop computers, laptops or servers.

Q. So in the traditional market for Java SE, as far as you

* * *

[1864] Q. Because you didn't even know what implementing code was at the time you formed your opinions; right?

A. That's correct.

Q. Are you relying on some other expert for this conclusion about the tipping point or that's all you?

A. I think it's both. I think there is some reliance on the technical experts like Dr. Schmidt regarding the role of the declarations, but I'm also as an economist thinking about the economic problem that Android needed to solve, which was having a developer community and getting the carriers on board, and, I mean, even Mr. Bloch last week referred to the declaring code as the nexus between the applications and the device, and so as an economist, I can understand that that means that having that is going to be important in bringing that developer community on board.

Q. Okay. So as an economist, you also know that SavaJe used the Java SE APIs; right?

A. I do know that, yes.

Q. And it was a big failure?

A. It was a failure.

Q. Huge; right?

A. It was a failure.

Q. Okay. So just having the Java – the method – actually, SavaJe had the whole kit and caboodle. It didn't just have the method headers; it had the implementing code and the whole [1865] thing, the virtual machine, all of the proprietary Sun code; right?

A. Yes.

Q. It was a flaming failure.

A. It was a failure.

Q. So just having all that doesn't guarantee success for a smartphone; right?

A. It does not guaranty success.

Q. And that's consistent with what you said yesterday, which was this is a very hard market to predict anything in; right?

A. Yes.

Q. I think you said nearly impossible to predict how this market comes out; right?

A. I did say that, yes.

Q. Okay. You also know that other smartphones that have been hugely successful don't use any Java at all; right?

A. Well, the only one that I know of is the iPhone.

Q. That's a pretty big, successful item; right?

A. It is. They had a different way of solving that economic problem.

Q. And they did it without any – they didn't use the Java programming language or the APIs or any of that; right?

A. That's correct.

Q. Apple is written in a completely different programming language?

* * *

[1897] All right. Go ahead.

BY MR. RAGLAND

Q. Dr. Leonard, did you form an opinion as to whether or not Android increased product variety?

A. I did.

It's – it has a lot of characteristics that no existing product in the marketplace at the time had; and, therefore, it expanded product variety and has been a boon for consumers.

Q. Now I would like to talk a little more detail, specifically regarding Java SE.

In the course of your work evaluating the evidence in this case, did you come to an understanding of what Java SE is?

A. Yes, I did.

Q. Can you please tell us what your understanding is.

A. Java SE is one of the Java applications programming platforms. And it's the one that specifically was designed for desktop computers.

Q. Did you reach an opinion as to whether or not Android had superseded Java SE in the market?

A. I have.

Q. Can you please tell the jury what that opinion is.

A. It has not superseded – Android has not superceded Java SE.

Q. Do you have reasons for reaching that opinion? If so can, you explain them?

[1898] A. Yes, I have two reasons.

The first is that the two products are on very different devices. As I just mentioned, Java SE is on personal computers. Android, on the other hand, is on smartphones.

Q. And what's the second reason?

A. Yeah, the second reason is that – I think we heard this before – is that Java SE is just an applications programming framework or platform. Whereas, Android is an entire mobile operating stack that runs a smartphone. Those are just very different types of products.

Q. To what extent, if any, are you aware of efforts by Sun to develop a smartphone operating system based on Java SE?

A. Uhm, well, Sun made several attempts to do that. And they ultimately all failed.

Q. And are you aware of whether or not Oracle similarly made attempts to develop a smartphone operating system using Java SE?

A. They did. And they also failed.

Q. Have you reached any conclusion, Dr. Leonard, with regard to the market impact, if any, of Android on Java SE?

A. Well, my conclusion is that Android does not have any market impact on Java SE.

Q. Turning, Dr. Leonard, to Java ME, did you perform an analysis of Java ME?

A. I did.

[1899] Q. Why did you do that?

A. Well, because that really was the focus of Dr. Jaffe's analysis, that the market for Java ME had been harmed.

Q. Do you have a view as to whether or not Dr. Jaffe is correct on that?

A. Uhm, my view is that it has not been harmed by the use of the 37 – or the declaring code and SSO of the 37 APIs.

Q. Do you have any reasons for that conclusion?

A. I do.

Q. Can you explain them?

A. It's really, in a way, the same two reasons as it was for Java SE.

The products are used on different devices and, therefore, there's no overlap from that point of view. And, secondly, they're just very different types of products.

Q. Did you consider, in the course of your review of the evidence, Dr. Leonard, any evidence that at one time Java ME was used on feature phones?

A. I did, yes.

Q. And what conclusions, if any, did you draw based on that review of the evidence?

A. I mean, it doesn't really change anything because, again, once smartphones, the advent of smartphones, following first with the iPhone and thereafter, once those products were introduced to the market – and those were, of course, very [1900] different than what had preceded it, including feature phones, and those products were addressing an unmet need.

And just as with the wordprocessing software we had, people who switched to smartphones because they wanted those kind of functionalities, those people weren't going to then consider a feature phone to be a substitute.

So once the smartphone movement started, Java's – ME's days were numbered because, as we've heard, it is not able to run on a smartphone. It wasn't designed for it. And it just wasn't going to be able to compete in this new world of smartphones.

Q. In your area of expertise and market research, Dr. Leonard, do you occasionally study consumer behavior?

A. Yes, I do.

Q. And how might consumer behavior relate to this analysis?

A. Well, it's really – and consumers are driving the economic substitution.

It's really what I just mentioned, that once you have – you have people who want to do certain things on their phone like watch a YouTube video. And the current – the existing products before Android came on the market, there was the iPhone that could do it. But other products, feature phones in particular, weren't very good at it.

This is what Mr. Rubin talked about. He wanted to create something that would allow that kind of experience.

* * *

[1924] outside the context of this litigation?

A. These visualizations – and I think Professor Schmidt talked about how he created them using soft-

ware – aren't at all similar to what Sun and Oracle have provided for developers to understand the API packages. We saw one of those in the courtroom.

Q. And what was that, that we saw in the courtroom?

A. We saw something that we were told you could get from Amazon that could be hung on the wall.

In fact, when I first started programming in Java, I had that diagram, because it did show all the classes in one kind of wall art that you could use to understand the API packages. You saw their names and how they were related.

So that chart was something that Sun and then Oracle distributed to help developers understand the API packages.

Q. Professor Astrachan, did you see Dr. Reinhold's slide setting forth some example labels using the API packages?

A. Yes, I did.

MR. PAIGE: Slide 17 of Dr. Reinhold's demonstratives, please.

(Document displayed.)

BY MR. PAIGE

Q. What does that slide demonstrate, in your opinion, about the nature of the names used in the API package labels?

A. I thought this was a very clear example of how functional [1925] and descriptive the names for the declaring code, the packages and classes and methods, are.

MS. HURST: Your Honor, I'm going to object. This is beyond the reply report; contains no rebuttal of Dr. Reinhold.

MR. PAIGE: Dr. Reinhold did not submit a report, Your Honor. He was disclosed as an employee expert on February 29th, the day the reply reports were due. So Professor Astrachan hasn't had chance to submit anything in writing, as this was the first time, yesterday, the witness has been shown –

MS. HURST: All right. I'll withdraw the objection, Your Honor.

THE COURT: Thank you.

Go ahead.

THE WITNESS: As I was explaining, I think this shows exactly how descriptive and functional the names of the methods, classes and packages are in the Java programs.

BY MR. PAIGE

Q. Let's take an example of that.

Were you familiar with the API label for the authentication class in the java.net package before Tuesday?

A. No. It was kind of interesting.

So I knew about the java.net package. That's one of the things I talked about, that because it's name is java.net we would understand that it had network classes.

[1926] But in my own programming use, I hadn't used the authenticator class before. But I thought when I looked at this description, I would have a pretty good understanding of how that method worked simply because of how the labels, the names are described.

THE COURT: It's not clear to me.

What the jury is now seeing, is that something you prepared? Or is that something that another witness has shown the jury?

THE WITNESS: This is what Dr. Reinhold showed the jury.

THE COURT: This very document?

THE WITNESS: This very one, yes, sir.

THE COURT: All right. And the – under "Declaration," what is – what is that?

THE WITNESS: Okay. I was going to explain that –

THE COURT: All right.

THE WITNESS: – second line there.

So what we see there is the package name, java.net, the class name, authenticator. Now, the declaration, that's a method declaration. So if we look at java.net. authenticator.requestpasswordauthentication, that's the method declaration. It includes the package name, the class name, and the method name, which, for me, is hard to say, apparently, because it has an "s" and a "th" [1927] requestpasswordauthentication.

So I knew immediately, when I read that label name, what it was used for. It was used to request a password authentication.

And so I thought, when I saw Dr. Reinhold display this, I think I can understand pretty well what this method would do based simply on its names and its inputs and outputs.

I thought that was kind of a good example of functionality and descriptive nature of these labels.

BY MR. PAIGE

Q. Well, did you do anything to confirm your opinion about what that class did?

A. I did. After I saw this in the – in trial, I went and looked up the API specification, the comments that a developer would use to understand it.

And my own understanding that I had built simply from looking at this demonstrative, were kind of validated because when I read the description, it was the same, which I think isn't surprising based on these descriptive names that we see for both the method, the inputs, and the outputs.

Q. Can you explain to the jury what you concluded from looking at those elements of the declaration?

A. There's a lot on this line.

For a developer, it wouldn't be too hard to understand because developers are used to being able to find the pieces.

[1928] And when I described the pieces of the method, I said the word "limiting," "request password authentication," and then the inputs and the outputs.

The output comes after "public static." We can see "password authentication." Now, that might be hard for people who aren't programmers, but for me I realized right away that's the return. That's what this method gives you back. It's a password authentication object. So I knew right away that's what I would get.

The inputs are also described. And we can see many. Host, iNet address, protocol, prompt, scheme, requester type. And, again, for a developer, those descriptive names would be very easy to understand.

If you look, for example, at the one that says "URL," a URL is a uniform resource locater. It's how we find things on the Internet like www.cnn.com. That would be a URL.

So here, this method requires as one of its inputs a URL. So the name of the class, URL in capital letters, is very descriptive and functional about what it does. And then the name of the input itself, which is the lower case url.

So although this looks like it's complicated, each part is very descriptive and functional about what the method does.

Q. And what does the name there tell you about the SSO of this method, if anything?

A. Well, I talked about how the package name, class name and [1929] method name, that together is, in my opinion, the structure, sequence and organization.

Because Java – the Java Language requires that every method be in a class, and that every class be in a package. So we see right there what's called the SSO; the package name, the class name, the method name. Java.net.authenticator.requestpassportauthentication.

Q. Would you expect to find a class like this in java.net?

A. I would absolutely expect to find a class like this in java.net because it deals with network authentication.

So the classes in java.net would be those around networks. This is precisely where I would think to find it.

Q. So what's your conclusion as to what Dr. Reinhold's slide shows about the functional nature of these names?

A. I think it's a good example of how these names are functional and descriptive of their purpose in Java.

Q. Dr. Reinhold also expressed the opinion that declaring code was relatively more important to some develop than implementing code.

Do you remember that testimony?

A. I do, yes.

Q. What is your opinion regarding the relative importance of declaring code and implementing code to application developers?

A. I think that you can't really make valid conclusions about the relative importance of declaring code versus implementing

* * *

[1935] BY MR. PAIGE

Q. In your opinion, was putting the entirety of the Java SE on a mobile phone a good idea?

A. No, I don't think that was a good idea.

Q. Why not?

A. Using all 166 packages from Java SE, which is a platform designed for desktop and laptop computers, all of those wouldn't be appropriate for a mobile platform.

And we saw that, for example, with SavaJE. It didn't work in making a successful platform. They missed, kind of, a key step by taking all of Java SE.

Q. What's the key step you're talking about?

A. Well, the Google engineers took – they took the ideas and they used the labels of these 37 packages. So in using the labels of just 37 packages, they were able to create the Android platform by adding the functionality to it that they need.

So the key step there was selecting just the 37 labels and not using the entirety of the Java SE.

Q. So, Professor Astrachan, could you summarize your opinion on Google's use of the Java APIs?

A. Sure.

That first step that I just outlined was selecting the 37 packages and using those method declarations, the declaring code. And then the next step was implementing those [1936] declarations with code optimized for a mobile platform, and then adding to that the library specific for a mobile platform. Things like location and WiFi.

So at that point, the Google Android developers then brought in third-party libraries, these open source libraries for making Web browsers or graphics. And then they made a virtual machine, the Dalvik Virtual Machine. Again, specifically optimized for a mobile platform.

So starting with that selection and then going down through the virtual machine, all optimized for Android, and then building that on top of Linux, a version of the low-level operating system specific for this Android platform, that whole sequence led to this open source innovative Android platform.

MR. PAIGE: Thank you, Professor Astrachan.

I pass the witness.

THE COURT: All right.

CROSS-EXAMINATION

BY MS. HURST

Q. Professor Astrachan, I want to hand you a binder with your reports in it, Exhibit 7641.

A. Okay.

Q. And if you open the binder, you'll see that your reply report there is with a flag on it, on page 5. Do you see that, sir?

A. I – this is my opening report. Do you want me to look at

* * *

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA

No. C 10-03561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

v.

GOOGLE INC.,

Defendant.

FINAL CHARGE TO THE JURY (PHASE ONE) AND SPECIAL VERDICT FORM

1.

Members of the jury, it is now my duty to instruct you on the law that applies to the issue of fair use. A copy of these instructions will be available in the jury room for you to consult as necessary.

It is your duty to determine the facts from all the evidence in the case. To those facts, you will apply the law as I give it to you. You must follow the law as I give it to you whether you agree with it or not. You must not be influenced by any personal likes or dislikes, opinions, prejudices or sympathy. That means that you must decide the case solely on the evidence before you. You will recall that you took an oath promising to do so at the beginning of the case. In following my instructions, you must follow all of them and not single out some and ignore others; they are all equally important. You must not read into these

instructions or into anything the Court may have said or done as suggesting what verdict you should return — that is a matter entirely up to you.

2.

The evidence from which you are to decide what the facts are consists of:

1. The sworn testimony of witnesses, on both direct and cross-examination, regardless of who called the witness;

2. The exhibits which have been received into evidence;

3. The sworn testimony of witnesses in depositions and other proceedings, read into evidence;

4. Any facts to which the lawyers have stipulated. You must treat any stipulated facts as having been conclusively proved;

5. Answers to interrogatories and requests for admission read to you during trial; and

6. Any facts that I have instructed you must be treated as having been established.

3.

Evidence may be direct or circumstantial. Direct evidence is direct proof of a fact, such as testimony by a witness about what that witness personally saw or heard or did. Circumstantial evidence is proof of one or more facts from which you could find another fact. By way of example, if you wake up in the morning and see that the sidewalk is wet, you may find from that fact that it rained during the night. However, other evidence, such as a turned-on garden hose, may explain the presence of water on the sidewalk. Therefore, before you decide that a fact has been proved by

circumstantial evidence, you must consider all the evidence in the light of reason, experience, and common sense. You should consider both kinds of evidence. The law makes no distinction between the weight to be given to either direct or circumstantial evidence. It is for you to decide how much weight to give to any evidence. You should base your decision on all of the evidence regardless of which party presented it.

4.

In reaching your verdict, you may consider only the types of evidence I have described. Certain things are not evidence, and you may not consider them in deciding what the facts are. I will list them for you:

1. Arguments and statements by lawyers are not evidence. The lawyers are not witnesses. What they have said in their opening statements, closing arguments, and at other times is intended to help you interpret the evidence, but it is not evidence. If the evidence as you remember it differs from the way the lawyers have stated it, your memory of it controls.

2. A suggestion in a question by counsel or the Court is not evidence unless it is adopted by the answer. A question by itself is not evidence. Consider it only to the extent it is adopted by the answer.

3. Objections by lawyers are not evidence. Lawyers have a duty to their clients to consider objecting when they believe a question is improper under the rules of evidence. You should not be influenced by any question, objection, or the Court's ruling on it.

4. Testimony or exhibits that have been excluded or stricken, or that you have been instructed to disregard, are not evidence and must not be

considered. In addition, some testimony and exhibits have been received only for a limited purpose; where I have given a limiting instruction, you must follow it.

5. Anything you may have seen or heard when the Court was not in session is not evidence. You are to decide the case solely on the evidence received at the trial.

5.

The weight of the evidence as to a fact does not necessarily depend on the number of witnesses who testify. Nor does it depend on which side called witnesses or produced evidence. You should base your decision on all of the evidence regardless of which party presented it.

6.

You are not required to decide any issue according to the testimony of a number of witnesses, which does not convince you, as against the testimony of a smaller number or other evidence, which is more convincing to you. The testimony of one witness worthy of belief is sufficient to prove any fact. This does not mean that you are free to disregard the testimony of any witness merely from caprice or prejudice, or from a desire to favor either side. It does mean that you must not decide anything by simply counting the number of witnesses who have testified on the opposing sides. The test is not the number of witnesses but the convincing force of the evidence.

7.

A witness may be discredited or impeached by contradictory evidence or by evidence that, at some other time, the witness has said or done something or

has failed to say or do something that is inconsistent with the witness' present testimony. If you believe any witness has been impeached and thus discredited, you may give the testimony of that witness such credibility, if any, you think it deserves.

8.

Discrepancies in a witness' testimony or between a witness' testimony and that of other witnesses do not necessarily mean that such witness should be discredited. Inability to recall and innocent misrecollection are common. Two persons witnessing an incident or a transaction sometimes will see or hear it differently. Whether a discrepancy pertains to an important matter or only to something trivial should be considered by you.

However, a witness you think is willfully false in one part of his or her testimony is to be distrusted in others. You may reject the entire testimony of a witness who willfully has testified falsely on a material point, unless, from all the evidence, you believe that the probability of truth favors his or her testimony in other particulars.

9.

In determining what inferences to draw from evidence you may consider, among other things, a party's failure to explain or deny such evidence.

10.

You may have heard from a witness that there was a prior trial in this case. It is true that there was a prior trial. We have heard evidence in this trial of a prior proceeding, which is the earlier trial that occurred in this case. Do not speculate about what happened in the prior trial. No determination on fair

use was made one way or the other in that trial. It is up to you, the jury, to determine fair use based on the evidence you have heard in this trial and my instructions of the law.

11.

In this case, members of the jury, you have heard two types of witnesses. *First*, you have heard fact witnesses. These are people who were part of the story on trial and have testified to the facts they experienced firsthand. *Second*, you have heard expert witnesses. Unlike fact witnesses who were part of the story on trial, the various expert witnesses have been retained by both sides after-the-fact to testify to opinions based on their specialized training or experience. To take an example from a more routine case, in a traffic case, a fact witness is someone who saw or heard the accident or was part of it, whereas an expert witness is someone like an accident reconstruction specialist who offers an opinion of the car's speed based on skid marks.

12.

In deciding the facts in this case, you may have to decide which testimony to believe and which testimony not to believe. You may believe everything a witness says, or part of it or none of it. In considering the testimony of any witness, you may take into account:

1. The opportunity and ability of the witness to see, hear, or know the things testified to;

2. The quality of the memory of the witness;

3. The manner of the witness testifying;

4. The interest of the witness in the outcome of the case and any bias or prejudice;

5. Whether other evidence contradicted the testimony of the witness;

6. The reasonableness of the witness' testimony in light of the evidence; and

7. Any other factors that bear on believability.

13.

With respect to expert witnesses, the main reason we allow their testimony is because they may have specialized training and experience with insights that may help the jury understand a field of specialized knowledge and how it applies to the case at hand. Usually, these witnesses are paid by their respective sides in litigation. Two important caveats for experts are as follows:

> No expert witness should ever vouch for which side's fact scenario is correct. No retained expert was present at the events in question. None has firsthand knowledge. Experts may rely on particular documents and testimony and may make an assumption that the document or testimony is correct and then give an opinion based on that assumption, but the opinion is only as good as the factual assumption and that foundational fact question is always for you, the jury, to resolve, not for the experts. Put differently, experts should not invade the province of the jury by purporting to tell the jury which side's fact version is true.
>
> Similarly, no expert witness should attempt to tell the jury what someone had in mind or was thinking. The mental state and intent of the characters in our story on trial is for you

to decide, not for the experts to decide. It is, however, permissible for experts to quote testimony or documents and then to assume that the statements therein were accurate and then based thereon to apply their expertise to render an opinion.

14.

With this in mind, I will now suggest to you some further inquiries for your evaluation of the testimony of experts.

1. To what extent, if at all, has the expert witness overstepped his or her role and tried to usurp the function of the jury by vouching for the truth of one side's witnesses versus the other or by giving opinion on the mental state of the characters involved in the case?

2. To what extent is the expert witness' opinion actually anchored in his or her specialized knowledge and training as opposed to just partisan argument, which you are just as qualified to make or reject as him or her?

3. To what extent is the expert witness' opinion supported by facts you find have been independently proven?

4. To what extent is the opinion contradicted by the facts?

5. To what extent has the expert witness relied upon a source of factual information that is biased?

6. To what extent has the expert witness "cherry picked" the factual record to highlight material helpful to his or her opinion while downplaying the facts that undercut his opinion?

7. To what extent has the expert witness forthrightly conceded points versus stubbornly refused to concede a point you think he or she should?

8. To what extent has the expert witness been influenced by money compensation paid by the side presenting him or her?

These are merely considerations. It is always up to you, the jury, to decide how much weight to give, if any, to any testimony or evidence, including from expert witnesses.

15.

Under the law, a corporation is considered to be a person. It can only act through its employees, agents, directors, or officers. Therefore, a corporation is responsible for the acts of its employees, agents, directors, or officers performed within the scope of authority.

You have heard testimony that Oracle Corporation bought Sun Microsystems, Inc., in 2010 and changed the name of the corporation from "Sun Microsystems, Inc." to "Oracle America, Inc." This means that Sun and Oracle America, the plaintiff in this case, are the same legal entity.

16.

In these instructions, I will often refer to a party's "burden of proof." Let me explain what that means. When a party has the burden of proof on any claim by a preponderance of the evidence, it means you must be persuaded by the evidence that the claim is more probably true than not true. To put it differently, if you were to put the evidence favoring a plaintiff and the evidence favoring a defendant on opposite sides of a scale, the party with the burden of proof on the issue would have to make the scale tip somewhat toward its

side. If the party fails to meet this burden, then the party with the burden of proof loses on that issue. Preponderance of the evidence basically means "more likely than not."

17.

If you find that Google carried its burden of proof as to fair use, your verdict should be for Google. If you find that Google did not carry its burden of proof, your verdict should be for Oracle.

18.

I will now remind you of some important established facts regarding the copyrighted works at issue in this case.

The Java platform is a software application platform that is used to write and to run programs in the Java programming language. The Java programming language is free and available to use without permission from anyone. The Java platform includes, among other things, the Java Virtual Machine and the Java API packages. "API" stands for "Application Programming Interface."

What is at issue in this case are the Java API packages, which are sets of prewritten computer programs used to perform common computer functions without a programmer needing to write code from scratch. These prewritten computer programs assist developers in writing applications. These prewritten programs are organized into packages, classes, and methods. An API package is a collection of classes. Each class contains methods and other elements.

The packages, classes, and methods are defined by declaring code. The declaring code is the line or lines of source code that introduce, name, and specify the

package, class, or method. The declaring code allows programmers to understand and make use of the prewritten programs in the API packages to write their own programs.

The declaring code for the packages, classes, and methods reflects the structure, sequence, and organization (or "SSO") for the Java API packages. The SSO specifies the relationships between and among the elements of the Java API packages and also organizes the classes, methods, and other elements in the package. The term structure, sequence, and organization is a concept used by lawyers and courts in connection with copyright. It is not a term used by computer scientists.

Each individual method performs a specific function. The declaring code for a method is sometimes referred to as the "method declaration," "header," or "signature." The declaring code for a method tells the programmer the information the method needs (the inputs) to perform the desired function.

Each method also contains implementing code. The implementing code provides step-by-step instructions that tell the computer how to perform the function specified by the declaring code. The declaring code and the SSO of the 37 Java API packages at issue are protected as part of the overall work protected by copyrights owned by Oracle. The copyright protection does not extend to the idea of organizing functions into packages, classes, and methods, but the copyright protection does cover the SSO as expressed in the 37 Java API packages.

19.

Sun developed the Java programming language and made it free for all to use. Sun further developed the

copyrighted Java API library of prewritten code, including implementing code, to carry out more advanced functions and made it available for all to use with a license, although the question for you to decide is the extent to which, if at all, the declaring code and SSO may be copied without a license under the statutory right of fair use. Anyone using the Java programming language may write their own library of prewritten programs to carry out various common functions. They may even write their own library to cover the same functions as covered by the copyrighted works. This is because copyright protects a particular set of words or expression, but it does not and cannot cover ideas or functions. However, even in writing their own programs to carry out the same functions, Java programmers may not begin their methods, classes, or packages with the identical line (or lines) of declaring code as used in the copyrighted works — unless such use of the declaring lines constitutes a fair use. Nor may they organize their methods into the same packages and classes as in the copyrighted works unless to do so qualifies as fair use.

20.

Now, I will turn to the law that applies to this case. In this trial, it has already been established that the Android versions in question used aspects of Java 2 Standard Edition Version 1.4 and Java 2 Standard Edition Version 5.0, specifically using the declaring code and the structure, sequence, and organization of 37 Java API packages. The pertinent Android versions are: 1.0, 1.1, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, Kit-Kat, Lollipop, and Marshmallow. Google's use of the declaring lines of code and the structure, sequence, and organization of those 37 API packages constituted

copyright infringement unless you find that Google has carried its burden as to the defense of fair use. In other words, for purposes of this trial, it is a given, already established, that Google used certain aspects of copyrighted works, and the question remaining for you to decide is whether or not Google's use was a fair use. There is no contention, however, that Google copied the implementing code for the 37 API packages. The point of contention is over the declaring lines of code within the 37 API packages, also referred to as declarations or header lines, which Google concededly used in Android, which reflect the structure, sequence, and organization for the Java API packages.

21.

Now, I will explain what fair use means under the law.

One policy behind our copyright law, of course, is to protect the compositions of authors from exploitation by others. When it applies, however, the right of fair use permits the use of copyrighted works by others without the copyright owner's consent. The policy behind the right of fair use is to encourage and allow the development of new ideas that build on earlier ones, thus providing a counterbalance to the copyright policy to protect creative works. Since the doctrine of fair use is an equitable rule of reason, no generally accepted definition is possible, and each case raising the question must be decided on its own facts. And, in this dispute between Oracle and Google, that question falls to you for decision.

22.

Under the Copyright Act, an author owns the exclusive right to use or to license his or her writings or images or other copyrightable works with the

statutory exception that anyone may make fair use of even a copyrighted work and may do so without anyone's permission and without payment of money to anyone. Specifically, the Act states (and I will quote it exactly):

> The fair use of a copyrighted work for purposes such as criticism, comment, news reporting, teaching (including multiple copies for classroom use), scholarship or research, is not an infringement of copyright. In determining whether the use made of a work in any particular case is a fair use the factors to be considered shall include
>
> 1. The purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes;
>
> 2. The nature of the copyrighted work;
>
> 3. The amount and substantiality of the portion used in relation to the copyrighted work as a whole; and
>
> 4. The effect of the use upon the potential market for or value of the copyrighted work.

I have just quoted for you the right of fair use exactly as enacted by Congress. As you just heard, the statute includes several examples of some types of uses that may be found to be fair uses, but that list is not exhaustive or exclusive. In your deliberations, you must decide whether or not Google has met its burden in this trial to prove that its copying was a fair use. Now I will further explain each of the four statutory factors.

23.

The first statutory factor concerns the purpose and character of the accused use. This factor includes these issues: (1) whether and to what extent the accused use serves a commercial purpose, which weighs against fair use, versus a nonprofit educational purpose, which weighs in favor of fair use, and (2) whether and to what extent the accused work is "transformative," which supports fair use. Although the Act does not explicitly use the word "transformative," our courts uniformly hold that the first statutory factor calls for an evaluation whether and to what extent the purpose and character of the accused work is transformative.

24.

What does transformative mean? A use is transformative if it adds something new, with a further purpose or different character, altering the first use with new expression, meaning, or message rather than merely superseding the objects of the original creation. New works have been found transformative when they use copyrighted material for purposes distinct from the purpose of the original material. A use is considered transformative only where a defendant changes a plaintiff's copyrighted work or, where the copyrighted elements remain unchanged from the original, a defendant uses them in a different context such that the original work is transformed into a new creation. A work is not transformative where the user makes little or no alteration to the expressive content or message of the original work and uses it in the same or similar context. The extent of transformation may vary from case to case. The greater the transformation, the more likely an accused use will qualify as a fair use, and the less the transformation, the less likely an accused use will qualify as a fair use.

25.

To qualify as transformative, the material copied need not be modified in the new work, so long as the material and the context in which the material is used qualifies as transformative under the test stated above. In this case, Google contends that it used the exact lines of declaring code at issue and their SSO together with new implementing code (and additional technology) as part of a new platform for mobile devices. Oracle contends that Sun was already using, licensing, and adapting the copyrighted works in mobile and other devices. It is up to you to decide the extent to which Google's use qualifies as transformative under the test stated above, but you may not disqualify it from being transformative merely because the declaring code and SSO were carried over without change. On the other hand, even if you find that the accused use was transformative, you must weigh that and the extent of the transformativeness against the commercial purpose of the use and its extent, which I will now discuss.

26.

In evaluating the first statutory factor, the extent of the commercial nature of the accused use must be considered. In this case, all agree that Google's accused use was commercial in nature but disagree over the extent. Commercial use weighs against a finding of fair use, but even a commercial use may be found (or not found, as the case may be) to be sufficiently transformative that the first statutory factor, on balance, still cuts in favor of fair use. To put it differently, the more transformative an accused work, the more other factors, such as commercialism, will recede in importance. By contrast, the less

transformative the accused work, the more other factors like commercialism will dominate.

27.

Also relevant to the first statutory factor is the propriety of the accused infringer's conduct because fair use presupposes good faith and fair dealing. Where, for example, the intended purpose is to supplant the copyright holder's commercially valuable right of first publication, good faith is absent. In evaluating the question of the propriety of Google's conduct, meaning good faith or not, you may only consider evidence up to the commencement of this lawsuit on August 12, 2010, and may not consider events thereafter. Your decision as to fair use, however, will govern as to all versions of Android at issue in this case, regardless of their date of issue. Again, in evaluating good faith or not, you should limit your consideration to events before August 12, 2010, and disregard any evidence you have heard after that date. This evidence cut-off date applies only to the issue of good faith or not.

In evaluating the extent to which Google acted in good faith or not, you may take into account, together with all other circumstances, the extent to which Google relied upon or contravened any recognized practices in the industry concerning re-implementation of API libraries.

You have heard evidence concerning the possibility of Google seeking a license from Oracle. Under the law, if the accused use is otherwise fair, then no permission or license need be sought or granted. Thus, seeking or being denied permission to use a work does not weigh against a finding of fair use.

Similarly, you have heard evidence about various licenses from the Apache Foundation, the Apache

Harmony Project involving Java, and the General Public License. These are relevant in some ways, but Google concedes it had no license from Sun or Oracle, and it is important to remember that Google makes no claim that its use was pursuant to a license from Sun or Oracle, directly or indirectly. Instead, Google claims that its use was a fair use and therefore required no license at all.

28.

The second statutory factor is the nature of the copyrighted work. This factor recognizes that traditional literary works are closer than informational works, such as instruction manuals, to the core of intended copyright protection. Creative writing and expression lie at the very heart of copyright protection, so fair use is generally more difficult to establish for copying of traditional literary works than for copying of informational works. The focus of this statutory factor is on how close the used material is to the core values of copyright protection. The less the used material implicates the core values of copyright protection, the more viable will be fair use and vice versa.

29.

In this case, it is undisputed that the declaring code and the structure, sequence, and organization of the 37 API packages at issue were sufficiently creative and original to qualify for copyright protection. "Original," as the term is used in copyright, means only that the work was independently created by the author (as opposed to copied from other works) and that it possesses at least some minimal degree of creativity. The extent to which the 37 API packages in question here involved greater creativity than the minimum required to obtain copyright is disputed and is open for you to

examine. That is, you should consider the extent to which the used materials were creative versus functional. The more creative the work, the more this factor disfavors fair use, and the more functional the work, the more this factor favors fair use.

30.

Even though a computer program performs functions and has functional elements, the structure, sequence, and organization of a computer program may be (or may not be) highly creative. When there are many possible ways to structure, sequence, and organize a program, the particular way chosen for a copyrighted program and individual lines of declaring code may be (or may not be) highly creative. On the other hand, when the declaring code and the structure, sequence, and organization are dictated by functional considerations such as efficiency, compatibility, or industry standards, then less creativity is indicated and the core values of copyright protection are less implicated. When purely functional elements are embedded in a copyrighted work and it is necessary to copy associated creative elements in order to utilize those functional elements, then this circumstance also favors fair use. Conversely, copying creative expression that is not necessary to perform the functions cuts against fair use.

31.

Google, of course, had the right to write its own code to perform any function it wished because no one can get a copyright on a general method of operation (other than to get a copyright on its specific implementation for that function). Unless it was a fair use, however, Google did not have the right to use the exact lines of declaring code and the overall structure, sequence,

and organization of the 37 API packages, as copyrighted by Sun (and now owned by Oracle).

32.

Because Google was free to use the Java programming language to write Android, you should also consider the extent to which you find it was necessary for Google to use any or all of the declaring code and structure, sequence, and organization of any of the 37 API packages to write in the Java language. Such a finding, to that extent only, would support fair use; to the extent you find it was not necessary, however, that finding would disfavor fair use. It is established that 170 lines of code at issue are technically necessary to use the Java programming language. Those 170 lines of declaring code are listed in Trial Exhibit 9223. Because that declaring code is necessary to use the language, it is established that Google's use of the declaring code in Trial Exhibit 9223 was a fair use. It is for you to determine the extent to which other additional declaring code beyond those lines identified in Trial Exhibit 9223 either was or was not necessary for use of the Java programming language. To the extent you find they were not necessary, you still must consider whether their use was (or was not) a fair use in light of the statutory factors for fair use. This consideration also bears on the third statutory factor, to which I will now turn.

33.

The third statutory factor is the amount and substantiality of the portion used in relationship to the copyrighted work as a whole, which concerns how much of the overall copyrighted work was used by the accused infringer. Analysis of this factor is viewed in the context of Oracle's copyrighted works, namely

Java 2 Standard Edition Versions 1.4 and 5.0. For this factor, the total number of lines in Android is irrelevant. The fact, if true, that a substantial portion of an infringing work was copied verbatim is evidence of the qualitative value of the copied material, both to the originator and to whoever seeks to profit from marketing someone else's copyrighted work. Wholesale copying does not preclude fair use per se but it militates against a finding of fair use. Even a small part may be qualitatively the most important part of a work. If, however, the secondary user only copies as much as is necessary for a transformative use, then this factor will not weigh against him or her. The extent of permissible copying varies with the purpose and character of the use, which relates back to the first statutory factor.

In assessing this third statutory factor, both the quantity of the material used and the quality or importance of the material used should be considered.

34.

The fourth and final statutory factor is the effect of the accused infringer's use on the potential market for or value of the copyrighted work. This factor militates against fair use if the accused use materially impairs the marketability or value of the copyrighted work. This is the single most important statutory factor, but it must be weighed with all other factors and is not necessarily dispositive. This factor considers whether the accused work is offered or used as a substitute for the original copyrighted work. This factor considers not only the extent of any market harm caused by the accused infringer's actions but also whether unrestricted and widespread use of the copyrighted materials of the sort engaged in by the accused infringer would result in a substantially adverse impact on the potential

market for the copyrighted work. Market harm to the value of the copyrighted work may be a matter of degree, and the importance of this factor will vary not only with the amount of harm shown, but also with the relative strength of the showings on the other factors.

35.

In connection with the fourth statutory factor, the term "potential market for or value of refers to the value of the entire copyrighted work itself and licensing opportunities for the copyrighted work and its derivative works. A derivative work is a work based in whole or in substantial part upon one or more preexisting copyrighted works, such as a musical arrangement or dramatization based on a book, to name only two specifics, or any other form in which a work may be recast or adapted. In this case, the copyrighted works in suit are Java 2 Standard Edition Versions 1.4 and 5.0, so the only derivative works that count are those derived from those two works.

36.

In making your evaluation under the fourth factor, you should assess the harm, if any, to the potential market for or value of the copyrighted work itself and to its licensing value for it and its derivative works. You may consider the broader potential market for products that feature independent elements in addition to the copyrighted material and their successes and/or failures only insofar as they shed light on the licensing or market value of the copyrighted work itself and its derivative works. In doing this, moreover, you must ignore benefits from the use to the copyright owner outside the genre claimed to have been harmed.

37.

Actual present harm need not be shown. Nor is it necessary to show with certainty that future harm will result so long as some meaningful likelihood of future harm exists to the market value of the copyrighted work or the licensing value for the copyrighted work and its derivative works in traditional, reasonable, or likely to be developed markets. If the intended accused use is for commercial gain, that likelihood may be presumed except where the second use is transformative because in cases of transformation, market substitution is at least less certain and market harm may not be so readily inferred.

38.

I have now completed my explanation of the four factors in the Act. You might ask, are we limited to these four factors? The Act states that the factors to be considered "include" the four statutory factors, and the law holds that those four factors are not exclusive and you may consider any additional circumstances and evidence, pro or con, that, in your judgment, bear upon the ultimate purpose of the Copyright Act, including protection of authors and the right of fair use, namely, to promote the progress of science and useful arts.

39.

It is up to you to decide whether all relevant factors, when considered fully and together, favor or disfavor fair use. All of these factors must be explored, discussed, and evaluated by you. No single factor is dispositive. Your evaluation of all factors must be weighed together in light of the purpose of copyright, which as our Constitution states in enumerating the legislative power of Congress, is to promote the progress of science and useful arts. Some factors may

weigh in favor of fair use and some against fair use, and you must decide, after giving the factors such weight as you find appropriate based on the evidence and my instructions, whether or not, on balance, Google has shown by a preponderance of the evidence that they predominate in favor of fair use.

40.

When you begin your deliberations, you should elect one member of the jury as your foreperson. That person will preside over the deliberations and speak for you here in court.

You will then discuss the case with your fellow jurors to reach agreement if you can do so. Your verdict must be unanimous. Each of you must decide the case for yourself, but you should do so only after you have considered all of the evidence, discussed it fully with the other jurors, and listened to the views of your fellow jurors.

Do not be afraid to change your opinion if the discussion persuades you that you should. Do not come to a decision simply because other jurors think it is right. It is important that you attempt to reach a unanimous verdict but, of course, only if each of you can do so after having made your own conscientious decision. Do not change an honest belief about the weight and effect of the evidence simply to reach a verdict.

I will give you a special verdict form to guide your deliberations.

41.

Some of you have taken notes during the trial. Whether or not you took notes, you should rely on your own memory of what was said. Notes are only to assist your memory. You should not be overly influenced by

the notes. When you go into the jury room, the Clerk will bring in to you the trial exhibits received into evidence to be available for your deliberations. The Clerk will also provide you with an index to them.

42.

As I noted before the trial began, when you retire to the jury room to deliberate, you will have with you the following things:

1. All of the exhibits received into evidence;

2. Indices of the exhibits, one in chronological order, one in order of exhibit number, and one index of the exhibits shown in video depositions.

3. A work copy of these jury instructions for each of you;

4. A work copy of the verdict form for each of you;

5. An official verdict form; and

6. A cart with a computer which holds exhibits that exist only in electronic form.

When you recess at the end of a day, please place your work materials in the brown envelope provided and cover up any easels with your work notes so that if my staff needs to go into the jury room, they will not even inadvertently see any of your work in progress.

43.

A court security officer will be outside the jury-room door during your deliberations. If it becomes necessary during your deliberations to communicate with me, you may send a note through the officer, signed by your foreperson or by one or more members of the jury. No member of the jury should ever attempt to

communicate with me except by a signed writing, and I will respond to the jury concerning the case only in writing or here in open court. If you send out a question, I will consult with the lawyers before answering it, which may take some time. You may continue your deliberations while waiting for the answer to any question. Remember that you are not to tell anyone — including me — how the jury stands, numerically or otherwise, until after you have reached a unanimous verdict or have been discharged. Do not disclose any vote count in any note to the Court.

44.

You have been required to be here each day from 7:45 A.M. to 1:00 P.M. Now that you are going to begin your deliberations, however, you are free to modify this schedule within reason. For example, if you wish to continue deliberating in the afternoons after a reasonable lunch break, that is fine. The Court does, however, recommend that you continue to start your deliberations by 8:00 A.M. If you do not reach a verdict by the end of today, then you will resume your deliberations tomorrow and thereafter.

It is very important that you let us know via the officer what hours you will be deliberating so that the lawyers may be present in the courthouse at any time the jury is deliberating.

45.

You may only deliberate when all of you are together. This means, for instance, that in the mornings before everyone has arrived or when someone steps out of the jury room to go to the restroom, you may not discuss the case. As well, the admonition that you are not to speak to anyone outside the jury room about this case still applies during your deliberation.

46.

Once you render a verdict on the fair use question, we may proceed to the shorter and final phase of the trial on damages issues, depending on your answer to the fair use question. This would still be within the June 10 end date stated earlier. Please do not allow any desire to complete trial sooner to influence your thinking. Once you render your verdict on the fair use issue, it will be final and may not be re-visited or modified during the second phase.

47.

After you have reached a unanimous agreement on a verdict, your foreperson will fill in, date and sign the verdict form and advise the Court that you have reached a verdict. The foreperson should hold onto the filled-in verdict form and bring it into the courtroom when the jury returns the verdict. Thank you for your careful attention. The fair use issue is now in your hands. You may now retire to the jury room and begin your deliberations.

Dated: May 23, 2016.

/s/ William Alsup
WILLIAM ALSUP
UNITED STATES DISTRICT JUDGE

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA

No. C 10-03561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

v.

GOOGLE INC.,

Defendant.

SPECIAL VERDICT FORM

YOUR ANSWER MUST BE UNANIMOUS.

Has Google shown by a preponderance of the evidence that its use in Android of the declaring lines of code and their structure, sequence, and organization from Java 2 Standard Edition Version 1.4 and Java 2 Standard Edition Version 5.0 constitutes a "fair use" under the Copyright Act?

Yes _____ (finding for Google)

No _____ (finding for Oracle)

Dated: May __, 2016.

_____

FOREPERSON

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA

No. C 10-03561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

v.

GOOGLE INC.,

Defendant.

SPECIAL VERDICT FORM

YOUR ANSWER MUST BE UNANIMOUS.

Has Google shown by a preponderance of the evidence that its use in Android of the declaring lines of code and their structure, sequence, and organization from Java 2 Standard Edition Version 1.4 and Java 2 Standard Edition Version 5.0 constitutes a "fair use" under the Copyright Act?

Yes ____✓____ (finding for Google)

No _____ (finding for Oracle)

Dated: May 26, 2016.

/s/ [Illegible]
FOREPERSON

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA

No. C 10-03561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

v.

GOOGLE INC.,

Defendant.

FINAL JUDGMENT

Based upon the unanimous verdict by the jury, FINAL JUDGMENT IS HEREBY ENTERED in favor of defendant Google Inc., and against plaintiff Oracle America, Inc.

IT IS SO ORDERED.

Dated: June 8, 2016.

/s/ William Alsup
WILLIAM ALSUP
UNITED STATES DISTRICT JUDGE

[224] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOOGLE, INC.,

Defendant.

San Francisco, California
April 17, 2012
Before the Honorable William H. Alsup

TRANSCRIPT OF JURY TRIAL PROCEEDINGS

---

* * *

[286] part of the business, is intellectual property important?

A. What we do is create intellectual property. We create hardware designs and we create, in this case, software designs.

And we, again, design computer software, they are computer programs. So we design computer programs. And then we build those computer programs.

Both the design of the program and the program itself are both intellectual property.

Q. Now, we're going to talk in this trial both about patent and copyrights, but I just want to focus on copyrights right now.

Does Oracle use copyrights to protect its intellectual property?

A. We use copyrights to protect both our software designs and the programs—and the computer programs themselves.

Q. Is it expensive to design software programs and develop software programs?

A. Oracle spends about $5 billion a year on research and development. 90 percent of that is spent on creating—designing and programming and creating computer software.

Q. And would it be possible to make that kind of investment if you did not have copyright protection for the intellectual property, the software that you created?

A. Well, no. If people could copy our software, in other words create cheap knock offs of our products, we wouldn't get paid for our engineering and we wouldn't be able to continue to

* * *

[290] programmer writes in the Java language, and then the Java programmer gets to reuse these building blocks, these programs, and include them and build a still larger program.

Q. When you refer to "these building blocks," these prewritten programs, what are you referring to?

A. That's, again, this library of programs that you access through APIs.

Again, the Java—the Java program environment includes these two parts: The Java language and this library of prewritten programs.

And those prewritten programs, again, are—the command structure of those prewritten programs are these APIs.

So when you program in Java, you write language statements, and you use the APIs to these prewritten programs.

Q. Now, is it necessary to use the Java APIs that Sun has created in order to use the Java programming language?

A. Absolutely not.

MR. VAN NEST: Objection, Your Honor. Calls for expert testimony.

THE COURT: Do you know the answer to the question?

THE WITNESS: Yes, Your Honor.

THE COURT: From personal knowledge?

THE WITNESS: Yes, Your Honor.

THE COURT: Overruled. Please answer.

THE WITNESS: There's a company in the UK that built [291] its own Java environment. And they used the Java programming language, but they created

their own set of APIs, prewritten programs. And that other environment is called Spring.

So Spring uses the Java programming language, but it doesn't use the Sun-created APIs. They have their own set of APIs and their own set of prewritten programs.

Furthermore, there are lots of programming languages that are just programming languages and don't have any prewritten library or programs for reuse.

BY MR. BOIES:

Q. Now, is it difficult/expensive to create APIs?

A. Uhm, arguably, it's one of the most difficult things we do at Oracle. When you design a program, the very first thing you do is create or define the APIs of the program. That's a task that's done by our most senior experienced and talented software engineers.

Q. Does Sun, and now Oracle, offer licenses to people who want to use the APIs for Java that Sun has created or Oracle has now created?

A. We do have a variety of licenses for Java.

Q. Can you explain what those types of licenses are?

A. Yeah. There are three kinds of licenses. There's the GPL open source license. There's a specification license. And then there's a commercial license.

Q. I want to go through each of those licenses and talk about

* * *

[293] open source license.

Q. Now, was a Java GPL open source license available to Google?

A. Absolutely.

Q. Now, you said a second type of license was a Java specification license. Do you recall that?

A. Yes.

Q. And would you explain what the nature of a Java specification license is.

A. Of course.

The Java specification license lets you look at all of the source code—excuse me. Excuse me. Let me be clear. The Java specification license—that is incorrect, what I said. It doesn't let you look at the source code. Let me back up.

The Java specification license lets you look at the Java documentation. Not the source code. All of the Java documentation, something that's in English. It's printed out on a sheet of paper. Let's look at all of those specifications, those design specifications.

And, then, using those design specifications you can build your own version of Java. So you can use our—you can use the designs and all the specs. You cannot look at the code. Very specifically, you are not allowed to look at the code. And then using those specifications, you can then build [294] your own version of Java.

Once you have built that version of Java, you must run a—what's called a compatibility test, to make sure that it is Java, to make sure—because we want

everyone's version of Java—IBM, by the way, did this. IBM has its own version of Java. Oracle has a version of Java. SAP has a version of Java.

There are lots of companies that have built their own versions of the Java environment; both the environment for writing programs and the environment for running programs.

Lots of people have done this, and they got a specification license. But they must—part of the specification license requires them to run this compatibility test called the TCK. I think it's Test Compatibility Kit. They have to run this TCK. And Oracle—before, Sun—charges for this compatibility test.

You can build your own version of Java using the specifications, but you must pass the compatibility test. When you do pass the compatibility test, you then are granted a license for Java copyrights and Java patents. But not until you pass the compatibility test.

Q. Is the Java specification license itself free?

A. The Java specification license itself is free. What we charge for is the TCK, the compatibility test kit.

Q. In order to use a Java specification license, you must buy

* * *

[298] You don't have to do that at this moment. We'll take 15 minutes. Thank you.

MR. BOIES: Thank you, Your Honor.

(Recess taken from 9:33 to 9:52 a.m.)

THE COURT: Be seated. Let's go back to work.

Just so everyone will know, the rule against talking to the witness only applies on cross-examination. So we aren't there yet.

Please, bring in the jury.

(Jury enters at 9:52 a.m.)

THE COURT: All right. Be seated, please.

Mr. Boies, please continue.

MR. BOIES: Thank you, Your Honor.BY MR. BOIES:

Q. Mr. Ellison, before the break you were talking about the importance of compatibility for Java. Do you recall that generally?

A. I do.

Q. Now, the jury has heard and will hear terms about fragmenting Java and forking Java. And could you explain what those terms mean.

A. It means creating incompatibility versions of Java and—that creates two problems.

One, it fragments the developer community because they have to, if you will, learn how to program in industry [299] standard Java and then this incompatible version of Java. So you would have to learn how to program two slightly—you know, somewhat different ways. So it fragments the developer community.

Second, it ends the notion of write once, run anywhere. No longer can a programmer write a program once and expect it to run on all the different computers that run the Java runtime environment.

So it fragments the developer community and it breaks the write once, run anywhere promise.

Q. Now, you mentioned earlier that IBM had its own version of the Java; Oracle has got its own version of Java; SAP, another big software company, has its own version of Java. Are those different versions of Java all compatible?

A. They are all compatible. The ones that you mentioned, IBM, SAP, Oracle, Red Hat, Sun—we can go on and on—they are all compatible.

Q. And I think you said the Java specification license required that a person taking that license, in order to get the copyrights and the other intellectual property that they got, had to agree to create a compatible version of Java. Is that correct?

A. Right. They got a specification license. They created their compatible version of Java, and they proved it was compatible by running the compatibility test.

* * *

[303] Q. Can you give me some examples of companies that are on the Java executive committee.

A. IBM—Oracle, IBM, SAP, HP, Red Hat. All of whom are our competitors, by the way. We compete, but we still cooperate around Java. And Google is also on the executive committee for Java.

Q. And from time to time are new versions of Java created through this community process?

A. Yeah. The people ask for improvements for Java, so they basically—they'll come up and they'll say, we'd like to add this feature to Java, we'd like to add that feature to Java.

And these recommendations then go before the executive committee. The executive committee votes on them. Eventually, we have a collection of improvements, and we come up with a new version of Java.

And Java 7, recently the executive committee voted on and approved Java 7.

Q. And is Java 7 a new version of Java?

A. Java 7 is the new version of Java.

Q. And when was that approved by the executive committee?

A. Several months ago.

Q. And when this Java 7 was up for approval, did all of the members of the executive committee have an opportunity to vote?

A. Yes. Everyone—everyone on the executive committee had an opportunity to vote.

[304] Q. And did anybody vote against this new version?

A. Everyone voted for it except for one company. The only company that voted against Java 7 was Google.

Q. With respect to the APIs that you've mentioned earlier, is using the APIs that Sun and Oracle created an advantage to a company that wants to program in the Java programming language?

A. We—we think these—this library of prewritten programs—and they use those prewritten programs through their APIs.

We think this library of prewritten programs, it's a good library of programs. We think it makes programmers much more productive if they use the library, they use our library.

Q. You mentioned there was this company, I think you said Spring, who had written their own APIs?

A. Yes.

Q. Does it take a period of time and expense and resources if you're going to go that route?

A. Yeah. Spring had to design their own APIs, and then they had to teach the developer community about these new APIs. And they had to persuade them that their collection of APIs, their library of programs, was in some ways better than the library of programs that Oracle and Sun had produced.

Q. And there came a time when you became aware that Google was using Java APIs that Sun had copyrighted and Oracle now owned the copyrights to; is that correct?

[305] A. That's correct.

Q. And did you take any action to try to address that?

A. I met with Eric Schmidt when he was CEO of Google, and I met with Larry Page, the current CEO of Google. And I tried to persuade them to build—to be compatible with the industry standard version of Java.

Q. And did they agree to do that?

A. No.

MR. BOIES: Your Honor, we have no more questions at this time.

THE COURT: Before we—thank you.

Before we go to the cross, someone out there has a very loud and noisy keypad. Who is that? I'm going to ask you to stop typing because it is distracting. And if you don't, the marshals will remove you.

I said before, when the lawyer has the floor there will be no distractions. And I mean that.

This is an important case to these parties, and the jury is going to hear every word without a bunch of tick, tick, tick, tick.

I don't know who it was, but I know the direction it was coming from. I don't think it was your table, Mr. Van Nest. I'm not accusing you. It's somebody out there in the public seating.

I'm sorry to be so strong about this, but this is

* * *

[389] the Java Community Process, number one.

Number two, we also took over the engineering of the—what we call the reference implementations and design specifications for many of the Java standards. Things like the Java Standard Edition, the design specifications for Java Enterprise Edition, et cetera.

And, third, we invested additionally, we also became the sponsors for the Java user groups, what's called the Java, you know, developer community,

which has a number of forums and other things where we go out and do events to get developers excited about where Java is going and also train new developers on Java technology so that we can keep the developer ecosystem vibrant.

Q. Do you have an estimate for how much Oracle has invested in Java since the acquisition?

A. Annually, we spend hundreds of millions of dollars on Java.

Q. And any plans to change that?

A. Not—no.

Q. Did you at some point get involved in analyzing Android for purposes of determining what Oracle would do about Android?

A. I evaluated Android to understand largely from the point of view of trying to understand, one, how we could make Android compliant with the Java specification, and, secondly, what technically would be involved to help make that possible.

* * *

[430] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOOGLE, INC.,

Defendant.

San Francisco, California
April 18, 2012
Before the Honorable William H. Alsup

TRANSCRIPT OF JURY TRIAL PROCEEDINGS

---

* * *

[601] I've got a problem, how am I going to solve it?" you don't care about all the Exception Classes. Eventually you might care when you find an error condition could happen, but if you care about that, then you can read the more detailed documentation.

BY MR. JACOBS:

Q. And can you focus for a minute on what this tells us about relationships between classes?

A. Sure. So just in the diagram, as in the diagram I was showing, there are tree structures on the left relating classes, and then there are also structures on the right showing interfaces, and dotted lines between the classes and the interfaces showing the relationship between the classes and the interface.

Now, moreover, as we were—as I was saying earlier, some classes are shown in a package even if they are defined in some other package, and that's just for reference. So here in the java.nio.channels Package, there is the Object Class. Then there is a little blue icon here indicating that Object is actually defined in the package with that particular blue icon. That blue icon is for the java.lang Package and here is Object (indicating).

Similarly, in the Channels Package we have reference to the Closable interface. As I said, that's not defined in the Channels Package. That's defined over in java.io which is right above, so here is the definition of the Closable [602] interface (indicating).

Q. Now, again, does the poster shows us methods or fields?

A. It does not show us methods or fields.

Q. Did methods or fields have relationships themselves that would extend to different interfaces or classes?

A. Methods and fields can be—are related to other interfaces or classes.

Q. And can you maybe return to your—

A. Yes.

(Witness resumes stand.)

Q. I think you have slides to illustrate this?

A. Yeah. If we can go to the next slide, please? That one.

(Document displayed)

A. Right. So let's go back for just a moment to the Car Class that we started with. So car has Stop, Start and blowHorn methods. Suppose it had another method called Paint.

Can you click that in, please?

So if you want to paint a car, you need to specify what color you want. And so the Paint method has what we call an input parameter and that input parameter is the color you want the car to be painted.

Now, to talk about a color, we need some way to represent that. And the way—the most natural way to do that is to create another class called Color. And maybe, maybe members of the Color Class have fields that are, for example,

* * *

[622] what are the core classes and interfaces and methods I would want in the API? And it's important fairly quickly to get to a high level summary of that so that you can understand, you know, what a possible structure may be and so that you can show it to other people and get feedback on it.

And when you do that, you actually—I mean, what I do, what every Java API developer I know does, is you start writing in fragments of actual Java programming language code. You would sketch out in a file—maybe it's just an email. It's not an actual source file. You sketch out in a file, well, there is going to be

this class called Channels and it's going to have a NewInputStream method and maybe some other method in it, and sketch out a few other classes, and you send that around to get comments from your colleagues, other people you're working with.

As time goes on, as you get feedback, you revise that design. It starts to get a little bit longer because it's getting more real. Maybe you start to insert some of the English prose. It's really sketchy right now, but you started to do that.

And then another thing to get to, not too quickly, but as quickly as you can, is to write actual instructions in the method, write actual code into those so that you can compiled this file, which is now a Java source file. You can compile it and share that compiled version of this class with [623] other developers to get their feedback.

Some developers are really good at looking at just a design and saying, "Oh, well, yea, I could use that," or, "No, that doesn't solve my use case." Other developers really like to have code that they can run and write their own code to use directly to see how it works. So it's important to have that.

Another reason that working on the implementation at the same time is important is that when you're designing an API, working on the implementation at the same time often identifies bugs in the API design. You might notice, for example, that the point I said earlier about an API requiring—sort of requiring bad performance. You might write some code, do a few tests and see, Well, no, I don't actually need to allocate an object every time. Let me change the API to

work in a different way so that bad performance isn't required.

Q. How long does it take you to design a substantial Java API package?

A. So to continue with the java.io Package example, that was an effort that took almost exactly two years. I was probably working on it around half time during that two-year period.

THE COURT: Just you or did you have helpers?

THE WITNESS: I had helpers. I had a couple of other engineers on the team helping me with the design and the implementation, and I also had a group of experts who were

* * *

[628] counting them.

You need to—you need to choose names. Choosing names is really important in an API. Sometimes a name is suggested by the context in which it's going to be used. But other times a name—finding the right name for something requires a lot of thought.

I'm reminded of what the old mystics used to say: If you know the name of the thing, you have power over the thing.

And so it's really important in an API for it to be easy to learn, to choose good names.

Good names can be really hard because if we think of the space of names as real estate, many of the good names are already taken, and you can't reuse them.

Especially the good short names. So if you're designing an API, you want to take into careful

consideration, well, if I'm going to define a class, is this a class that many developers are going to use, or—and so I should really work hard to find a great descriptive short name for it? Or is this a class that is not going to be used that much, and so it's okay if it has a longer and uglier name? So there are a lot of choices to be made there.

But it's not just about the names. It's also about the structure. You know, how should classes be organized under other classes? How should interfaces be organized under other interfaces? How should classes and interfaces relate? Where [629] should the methods be? What should the methods be named? What kinds of inputs do the methods take? What kind of outputs do the methods provide for the fields? How do with they relate? Is the value in a field a color, or is it just a number, or is it a string, or is it something else?

So there are many, many design choices to be made.

THE COURT: Can I ask a question before you leave this subject.

This committee, do any of the—if someone on the outside of Sun wanted to propose an API or a new method to go in an old API, did that ever happen?

THE WITNESS: Oh, yes. That happens all the time.

THE COURT: Give us an example of how that would come down.

THE WITNESS: So there have been quite a few JSRs that were not initiated by Sun, and now not initiated by Oracle. I don't remember offhand what the statistics are, sir.

In the case of someone outside the company who wants to propose just a small new thing, that would generally come in through a process we have for collecting input from any software developer as a—a request for enhancement.

When a developer submits an idea like that, they might actually include a little bit of code. And when they submit that idea, there's—there's a button I believe they have to click on where they agree that they're contributing any [630] IP that might be in that idea, that code, whatever it is, so that Sun or now Oracle can use it.

An additional way things can come in more recently is through the OpenJDK community, where we have outside contributors who are actually able to suggest, review, and actually put the code in themselves, because they've demonstrated they have the experience and knowledge and judgment to do that in the right way.

THE COURT: Thank you. Go ahead.

BY MR. JACOBS:

Q. Could you give an example of a package in the Java API that has a different—that has a different version of it out there with a different structure?

A. So pretty much any Java API package you could look at and find something out in the world that's sort of like it. But—excuse me—a good example is a package called java.util.logging. L-o-g-g-i-n-g.

Logging is a facility that is often used in programs that run for a very long time. Like maybe on a server, computer, something that's processing. Bank transaction is—hopefully, it's going to run without crashing

for at least a day during the banking day. During that time, many different things could happen. If something goes wrong, you want to be able to diagnose what went wrong with it.

So in a long-running program, it's useful to create a [631] log of its activities. In a log for a banking application, you might record each transaction, just in text form. What's going on with this transaction? Is it completed yet? Whose account is it for? And so forth. So if something does go wrong, you can go back and look at that log.

Anyway the java.util.logging API package is a simple facility for logging messages. Around the time it was introduced, there was a competing package called Log4J. This was created by developers outside of Sun.

There's actually, to this day, a little bit of tension in the community because the people who like Log4J, they really hate java.util.logging. And the people who like java.util.logging don't much like Log4J.

But if you look at them from a functional perspective, they solve exactly the same kinds of problems. But, they are very different APIs. They have different class names, different method names, different interfaces, and different relationships.

Q. Has the number of Java APIs changed over time?

A. The number of Java APIs has grown dramatically over time.

Q. How many API packages were in the first release of Java, in 1996?

A. In 1996, Java 1.0 had seven API packages.

Q. And remind us how many there are in SE 5 and in Java 7.

A. Java SE 5 had 166, and Java 7 has 209.

* * *

[648] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOOGLE, INC.,

Defendant.

San Francisco, California
April 19, 2012
Before the Honorable William H. Alsup

TRANSCRIPT OF JURY TRIAL PROCEEDINGS

---

* * *

[747] your presentation. "Code should read like prose."

Do you see that on your presentation?

A. Yes.

Q. What were you driving at?

A. Well, writing a program is very much a creative process. And if you have good words to use, if the API gives you good words that really mean what it is that you're doing with the API, then once it comes time for

you to take that API and write a program with it, the program will read like English text.

For example, you know, suppose you have a car class. This is the example I have here. If you have a method called speed, and you have—then you can say if car.speed is more than twice the speed of light, generate alert, watch out for cops.

Now, even though you aren't programmers, you know what that mean.

THE COURT: "Speed of light." I think you meant speed limit.

THE WITNESS: Speed limit. I'm sorry.

(Laughter)

MR. JACOBS: You're talking quickly.

THE WITNESS: You're violating laws if it's faster than the speed of light. Guarantee that.

(Laughter)

THE WITNESS: The point is, if you look at this code

* * *

[751] A. Nominally, Professor Doug Lee. There's something called JSR-166x. This was sort of a continuing sort of pseudo JSR that's been going on for the past decade or so, that allows a group of colleagues …

Q. But that person who came up to you, had obviously left a vivid impression in your mind, was saying to you: API design is a noble and rewarding craft. You changed my life with the quality of your craftmanship. Correct?

A. Yes.

Q. And, "Here's what good API design can do," you said in your presentation, "It can improve a lot of programmers and users and companies." Correct?

A. Yes.

Q. And "API design is tough." That's what you were also saying?

A. Yeah. Designing a good API is tough.

Q. Like any work of craftmanship?

A. I agree with that.

Q. Creating a beautiful painting is tough?

A. I'm not sure if that's craftsmanship or art, but I guess that's a fine distinction.

Q. And API design, you said and believe, is a noble and rewarding craft. Correct, sir?

A. Yes, I certainly believe that.

Q. And, in fact, people have told you that this presentation

* * *

[1133] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOOGLE, INC.,

Defendant.

San Francisco, California
April 23, 2012
Before the Honorable William H. Alsup

TRANSCRIPT OF JURY TRIAL PROCEEDINGS

---

* * *

[1218] Review with us what this poster is showing.

A. This is a great poster. This shows a number of packages from the Java Class Library. For each package there's a really nice, graphical notation showing the relationships between elements in that package; and, also, there's a color-coding system used to show connections and relationships between separate packages.

Q. What's the basic concept in the Java class libraries illustrated on the poster?

A. The most basic concept is a class. And a class is used in object-oriented programming to create objects. Each object created by a class has the methods. Those are operations on objects of that class. And fields, each object has fields which are, again, associated with objects of the class.

Q. Can classes be related to each other?

A. Yes, they can. The most fundamental relationship between classes is the subclass relationship. When one class has all of the methods and fields of another, it can be declared to be a subclass of the first class.

Q. So let's look at a portion of this poster. We're looking at the lower right-hand corner, down here, of this exhibit. Can you tell us what this is showing?

A. This is the graphical notation. To me, it kind of reminds me of musical notation because there are lines and dots in it. On the left side there are classes. And here there's [1219] a vertical line underneath a class with horizontal solid lines connecting to subclasses of that class. So that illustrates the class/subclass relationship that's fundamental to object-oriented programming.

Another relationship illustrated here is a relationship between classes and interfaces. An interface lists a set of methods. And a class is related to that interface if the class provides all the methods and other elements listed in that interface.

Here, this word "interface" is used in a particular way specific to the Java language. It's different from

the more generic use of the word "interface" in the term API or application program interface.

Q. How are the classes and interfaces grouped?

A. Uhm, the classes are hierarchical, and grouped under each other. The interfaces can be hierarchical. Classes, interfaces, and other elements can be organized into packages. And packages, again, can be hierarchical. A package can have a number of subpackages that are related to it in some way.

Another thing that's shown on this legend for the poster is the way a class might—in one package, might be a subclass of a class that comes from another package. So that's another kind of relationship between packages and classes.

Here, this illustration shows a class with a blue dot with a star in it. That's to indicate that that class in the

* * *

[1239] a subclass hierarchy to make it easier to write the implementation and to make it easier for programmers who use the library to understand how the different concepts in the library work together.

Interfaces are used to make it possible to write code that operates over many different classes in the same way showing commonality across those classes.

And then the organization into packages is helpful in informing programmers how to understand where each solution, each program they might want to use in buildings their system, sits in the library. And to make a summary and conceptual organization clear across the library.

Q. And how are those choices illustrated on the slide 9 we saw before and is up again?

A. If—if you remember, there are—each class has methods. So this shows an example of the methods in one class. The hierarchy of Buffer, ByteBuffer, MappedByteBuffer and how these classes are related to each other is shown in this slide.

And then it's a little bit gray in the slide, but this package—these classes shown here are in the nio package. That package is part of a larger library of interrelated packages with classes that refer to each other in various ways.

I don't think I mentioned yet that each method in a class has parameters, data that must be supplied to the method [1240] in order for it to do its job, and returns a value of a given type.

The types of the parameters and the return type of the method can be classes from anywhere in the library. So that's another way that code in one portion of a library, in one package, can depend on and use code in another.

The arguments to a method, the parameters to the method and the return can be classes from other packages.

Q. How does designing an application programming interface compare with other aspects of writing software?

A. When you start to design an API for a portion of a library, you really start with a clean slate. So at that point nothing has really been determined. The high-level decisions that govern the organization of the

whole software system can start with the API. So that's the starting point.

And then from there, as more and more decisions are made, the remaining decisions are more constrained. That's not to say there isn't creativity involved at all levels, but there certainly is quite a bit at the beginning in order to map out the organization of a system.

Q. Let's take a look at a—an example of an API concept as architected in different contexts.

Can you explain to us what's shown on slide 10?

A. Certainly. This is one of my favorite examples. I usually teach the—some elements or aspects of the Smalltalk

* * *

[1266] language ever, right?

A. It certainly is very popular, yes.

Q. And it was released back in 1996?

A. I believed that's correct.

Q. And it became popular very quickly?

A. Yes.

Q. Thousands of developers were very soon writing the Java programming language, correct?

A. Yes.

Q. They built a developer conference for the developers trained in the language to come to every year called JavaOne?

A. Yes.

Q. And that became a larger and larger event every year?

A. Right. Yeah, I believe it was, even in those early years, the largest developer conference at that time.

Q. And soon the Java language was being taught in colleges and universities around the country—

A. Yes.

Q. (Continuing)—right?

Thousands of students learning the Java language every year?

A. Yes.

Q. You were teaching—you were teaching the Java language in your class?

A. I have covered it. It's also been covered in the—in [1267] other programming classes at our university and others.

Q. And so a large group of people writing in the Java language has grown up over time and now there's a very large group, hundreds of thousands of people writing in the Java programming language around the world?

A. There are many developers who are familiar with the language and use it.

Q. And the language itself has been among the top three programming languages for a long time, right?

A. The statistics probably show that, yes.

Q. And you set forth in your report that even in 2011 the Java programming language was number

one in terms of popularity of programming languages around the world?

A. You're probably referring to what's called of Tiobe Survey. It's based on examination of web pages.

Q. And in preparing your report, you assumed that Oracle was making no claim for copyright infringement based on the use of the language, right?

A. I have heard that said, yes.

Q. Actually, you relied on it in your report, correct?

A. I wasn't asked to look into issues of copyright of the language specifically.

Q. In other words, in preparing your opinions you started from the premise that the Java language itself was free to use and there was no claim in this case of infringement based on

* * *

[1879] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOOGLE, INC.,

Defendant.

San Francisco, California
April 26, 2012
Before the Honorable William H. Alsup

TRANSCRIPT OF JURY TRIAL PROCEEDINGS

---

* * *

[2062] A. First page.

Q. Last paragraph.

A. Yes, the financials—okay. I do see that.

Q. What did you mean by that?

A. We—we had a very lucrative revenue stream from J2ME, which was the handset version of Java, that we had licensed to just about every smart phone carrier, except Apple, around the world.

So it was—it was a very strong revenue stream for us. It was royalties which have no cost to goods sold, so it was fundamentally nearly pure profit, that revenue stream. So it was quite valuable.

Q. Let me ask you to look next at Trial Exhibit 563, which we believe is in evidence, but I want to check with counsel.

MR. VAN NEST: Yes, it is.

THE WITNESS: Can I use the screen? It's empty.

THE COURT: Sure, you can use the screen.

(Document tendered to the witness.)

BY MR. BOIES:

Q. This is an e-mail from you to Mr. Schwartz, and then a response from Mr. Schwartz to you. Correct?

A. Yes, it is.

Q. And you wrote this in or about March of 2007; is that correct?

A. That's correct.

* * *

[2128] UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

No. C 10-3561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOOGLE, INC.,

Defendant.

San Francisco, California
April 27, 2012
Before the Honorable William H. Alsup

TRANSCRIPT OF JURY TRIAL PROCEEDINGS

---

* * *

[2229] THE COURT: All right. Go ahead.

BY MR. JACOBS

Q. In court we've heard some critique of the blueprint analogy, that APIs are not like a blueprint because you can't actually use them to build anything. Do you agree with that?

A. No, I don't. In a way, the whole, the whole point of the Java community process is exactly to be designing APIs, blueprints, so that different organizations,

different companies, can create competing implementations.

Q. So how does a software developer—or how in the Java community do software developers go about using Java API for a blueprint for a class library?

A. So when you already have a good API designed, it already has an established gone-through hierarchy of packages and classes and methods and fields, together with English descriptions of how everything is supposed to work together.

Once you've got that, to do an implementation from scratch is a relatively easier job. You start by copying the declarations from the API into your source code. And then you fill in the methods with actual instruction code that will go at runtime. And you might need to write some subsidiary internal classes, but those are strictly not part of the API.

Q. Is implementing—based on your experience in this business, is implementing an existing API design less or more work than creating the API design in the first place?

[2230] A. It's almost always less. You've already got a map worked out of what you need to do. You follow that map. You fill in the details. There's room for creativity, but only within the scope of the existing API design.

Q. Dr. Reinhold, you started working on Java at Sun in what year?

A. 1996.

Q. Was there a JCP, a Java Community Process, in 1996?

A. No.

Q. When was it formed.

A. 1998.

Q. There has been testimony in this case, including from you, about the process that takes place at the JCP, about approving a new specification, called the JSR. And you described that, in working on the JSR, usually an expert group is formed that gives advice and comments on the new spec.

Do you recall that testimony?

A. Yes.

Q. Were these experts required to sign any kind of agreement to participate in the expert group?

A. Yes, they are.

Q. What kind of agreement is that?

A. So that's an agreement called the JSPA, the Java Specification Participation Agreements.

Q. And how strict was Sun about requiring the JSPAs to be

* * *

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

Case No. CV 10-03561 WHA

ORACLE AMERICA, INC.,

Plaintiff,

vs.

GOOGLE, INC.,

Defendant.

Dept. Courtroom 8, 19th Floor
Judge: Honorable William H. Alsup

JOINT RESPONSE TO COURT'S REQUEST FOR CHART OF ELEMENTS IN ACCUSED PACKAGES (ECF NO. 1124)

Package Name	J2SE 5	
	Classes plus Interfaces	Methods from Classes and Interfaces
java.awt.font	18	175
java.beans	35	209
java.io	62	569
java.lang	43	813
java.lang.annotation	1	4
java.lang.ref	5	9
java.lang.reflect	17	135
java.net	40	440
java.nio	10	236
java.nio.channels	20	112
java.nio.channels.spi	5	33
java.nio.charset	5	75
java.nio.charset.spi	1	2
java.security	62	362
java.security.acl	5	27
java.security.cert	35	267
java.security.interfaces	13	27
java.security.spec	25	75
java.sql	25	662

Package Name	J2SE 5	
	Classes plus Interfaces	Methods from Classes and Interfaces
java.text	27	342
java.util	65	866
java.util.jar	10	43
java.util.logging	17	137
java.util.prefs	7	95
java.util.regex	3	48
java.util.zip	15	117
javax.crypto	18	166
javax.crypto.interfaces	4	6
javax.crypto.spec	15	44
javax.net	2	11
javax.net.ssl	29	184
javax.security.auth	7	33
javax.security.auth.callback	9	32
javax.security.auth.login	4	10
javax.security.auth.x500	2	11
javax.security.cert	2	20

Package Name	J2SE 5	
	Classes plus Interfaces	Methods from Classes and Interfaces
javax.sql	14	111
TOTAL	677	6508

Package Name	Android Froyo	
	Classes plus Interfaces	Methods from Classes and Interfaces
java.awt.font	2	11
java.beans	5	23
java.io	62	570
java.lang	43	805
java.lang.annotation	1	4
java.lang.ref	5	8
java.lang.reflect	17	132
java.net	40	433
java.nio	10	236
java.nio.channels	20	112
java.nio.channels.spi	5	33
java.nio.charset	5	75

Package Name	Android Froyo	
	Classes plus Interfaces	Methods from Classes and Interfaces
java.nio.charset.spi	1	2
java.security	62	358
java.security.acl	5	27
java.security.cert	35	267
java.security.interfaces	13	27
java.security.spec	25	75
java.sql	25	662
java.text	27	342
java.util	65	868
java.util.jar	10	44
java.util.logging	17	137
java.util.prefs	7	95
java.util.regex	3	49
java.util.zip	15	119
javax.crypto	18	166
javax.crypto.interfaces	4	6
javax.crypto.spec	15	44
javax.net	2	11
javax.net.ssl	29	184

Package Name	Android Froyo	
	Classes plus Interfaces	Methods from Classes and Interfaces
javax.security.auth	5	27
javax.security.auth.callback	3	6
javax.security.auth.login	0	0
javax.security.auth.x500	1	6
javax.security.cert	2	20
javax.sql	12	104
TOTAL	616	6088

Trial Exhibit 1072

Accused API Packages and Files in Android

API Packages

1. java.awt.font
2. java.beans
3. java.io
4. java.lang
5. java.lang.annotation
6. java.lang.ref
7. java.lang.reflect
8. java.net
9. java.nio
10. java.nio.channels
11. java.nio.channels.spi
12. java.nio.charset
13. java.nio.charset.spi
14. java.security
15. java.security.acl
16. java.security.cert
17. java.security.interfaces
18. java.security.spec
19. java.sql
20. java.text
21. java.util

22. java.util.jar
23. java.util.logging
24. java.util.prefs
25. java.util.regex
26. java.util.zip
27. javax.crypto
28. javax.crypto.interfaces
29. javax.crypto.spec
30. javax.net
31. javax.net.ssl
32. javax.security.auth
33. javax.security.auth.callback
34. javax.security.auth.login
35. javax.security.auth.x500
36. javax.security.cert
37. javax.sql

Files

TimSort.java

ComparableTimSort.java

AclEntryImpl.java

AclImpl.java

GroupImpl.java

OwnerImpl.java

PermissionImpl.java

PrincipalImpl.java

PolicyNodeImpl.java

AclEnumerator.java

CodeSourceTest.java

CollectionCertStoreParametersTest.java