KEKER & VAN NEST LLP
ROBERT A. VAN NEST - # 84065
rvannest@kvn.com
CHRISTA M. ANDERSON - # 184325
canderson@kvn.com
MICHAEL S. KWUN - # 198945
mkwun@kvn.com
633 Battery Street
San Francisco, CA 94111-1809
Tel: 415.391.5400
Fax: 415.397.7188

KING & SPALDING LLP
DONALD F. ZIMMER, JR. - #112279
fzimmer@kslaw.com
CHERYL A. SABNIS - #224323
csabnis@kslaw.com
101 Second Street, Suite 2300
San Francisco, CA 94105
Tel: 415.318.1200
Fax: 415.318.1300

KING & SPALDING LLP
SCOTT T. WEINGAERTNER
(*Pro Hac Vice*)
sweingaertner@kslaw.com
ROBERT F. PERRY
rperry@kslaw.com
BRUCE W. BABER (Pro Hac Vice)
1185 Avenue of the Americas
New York, NY 10036
Tel: 212.556.2100
Fax: 212.556.2222

IAN C. BALLON - #141819
ballon@gtlaw.com
HEATHER MEEKER - #172148
meekerh@gtlaw.com
GREENBERG TRAURIG, LLP
1900 University Avenue
East Palo Alto, CA 94303
Tel: 650.328.8500
Fax: 650.328.8508

Attorneys for Defendant
GOOGLE INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

ORACLE AMERICA, INC.,

Plaintiff,

v.

GOOGLE INC.,

Defendant.

Case No. 3:10-cv-03561 WHA

GOOGLE'S MAY 10, 2012 COPYRIGHT LIABILITY TRIAL BRIEF

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

TABLE OF CONTENTS

Page

I. Google requests that the Court hold that the structure, sequence and organization ("SSO") of the 37 API packages is not copyrightable. 1

II. Google's responses to the Court's questions 1

A. Question 1: The vocabulary and grammar of a computer language—as distinct from programs written in the language—are not copyrightable because the Copyright Act protects expression, not vocabulary. 1

1. Copyright does not protect vocabulary, and that lack of protection extends to the SSO of vocabulary. 1

2. The ECJ's decision in *SAS Inst. Inc. v. World Programming Ltd* supports the conclusion that the vocabulary of a programming language cannot be copyrighted. 4

B. Question 2: If each API method is treated as a program, then Google did not copy anything more than the name and declaration of that program, which is not copyrightable. 5

C. Question 3: The form of the fully qualified names of the methods in the 37 API packages is "package.class.method," where the package names start with "java." or "javax." and this form is required by the syntax of the Java language. 6

D. Question 4: Google could have come up with at least some different names and SSO yet still provided similar functionality in Android, but this would not have been consistent with industry custom and developer demand. 6

1. Many of the elements of the 37 API packages are *required* by the Java language. 7

2. Industry custom and developer demand require implementation of the 37 API packages. 8

3. When developing new APIs in the Java language, it is standard practice to build upon the standard J2SE APIs. 9

4. The Court should disregard Mr. Ellison's testimony about Spring. 9

E. Question 5: The input-output scheme of a method is not copyrightable. 10

F. Question 6: The "core" packages in 1996 included *at least* java.lang, java.io and java.util, and today include at least 34 of the accused packages. 10

G. Question 7: The Java language requires more than just the java.lang, java.io and java.util packages. 11

H. Question 8: All 37 accused API packages should be deemed "core"packages. 12

I. Question 9: There are cross-method, cross-class interdependencies *at the implementation level* in J2SE, and those implementation level interdependencies are not always duplicated in the Android implementations. 12

J. Question 10: At the name/declaration level, the only interdependencies in the Java language come from (1) the package/class/member organizational scheme; (2) inheritance via subclassing and subinterfacing; and (3) interface implementation. 13

K. Question 11: The Seventh Circuit's decision in *American Dental Ass'n v. Delta Dental Plans Ass'n*, was either wrongly decided, or unclear about the scope of its holding. 13

L. Question 12: *CDN, Inc. v. Kapes* was not about a method of operation or system. 15

M. Question 13: Android is compatible with the 37 API packages at issue because code written to use those APIs will compile and work properly on an Android device. 16

N. Questions 14-15: In the Java language, "inheritance" is a concept applicable to classes, not packages; by virtue of the Java language specification, a subclass inherits fields, methods, and other members from its superclass. 17

O. Question 16: Copyright protection does not extend to names, including fully qualified names, nor does it extend to input-output (argument-return) designations, exception types, subclass inheritances, or interface relationships. 18

III. The Court should hold that the SSO of the 37 API packages is not copyrightable. 18

TABLE OF AUTHORITIES

Page(s)

Federal Cases

Allen v. Academic Games League of Am., Inc.
89 F.3d 614 (9th Cir. 1996) 6, 10

ATC Distrib. Grp., Inc. v. Whatever It Takes Trans. & Parts, Inc.
402 F.3d 700 (6th Cir. 2005) 5, 14

Baker v. Selden
101 U.S. 99 (1879) 1, 6

Computer Assocs. Int'l, Inc. v. Altai, Inc.
982 F.2d 693 (2d Cir. 1992) 8

Engineering Dynamics, Inc. v. Structural Software, Inc.
46 F.3d 408 (5th Cir. 1995) 10

Feist Pubs., Inc. v. Rural Tele. Serv. Co.
499 U.S. 340 (1991) 3, 4, 15

Gates Rubber Co. v. Bando Chem. Indus., Ltd
9 F.3d 823 (10th Cir. 1993) 8

Herbert Rosenthal Jewelry Corp. v. Kalpakian
446 F.2d 738 (9th Cir. 1971) 6

Lotus Dev. Corp. v. Borland Int'l, Inc.
49 F.3d 807 (1st Cir. 1995),
aff'd by an equally divided court, 516 U.S. 233 (1996) 6

Sega Enters. Ltd v. Accolade, Inc.
977 F.2d 1510 (9th Cir. 1992) 1, 6, 8, 10, 16

Seng-Tiong Ho v. Taflove
648 F.3d 489 (7th Cir. 2011) 13

Southco, Inc. v. Kanebridge Corp.
258 F.3d 148 (3d Cir. 2001) 13

Swirsky v. Carey
376 F.3d 841 (9th Cir. 2004) 8

Other Cases

SAS Inst. Inc. v. World Programming Ltd
Case C-406/10 (ECJ May 2, 2012) 4, 5

Federal Statutes

17 U.S.C. § 102 1

17 U.S.C. § 102(b) ..... *passim*

Federal Rules

Fed. R. Evid. 201(b)(2) 9

Federal Regulations

37 C.F.R § 202.1(a) 2, 5

I. Google requests that the Court hold that the structure, sequence and organization ("SSO") of the 37 API packages is not copyrightable.

As directed by the Court, Google provides responses to the sixteen questions the Court has posed. *See* Dkts. 1057, 1062, 1088. Google also offers further argument in support of the conclusion that the SSO of the 37 API packages is not copyrightable.

For the reasons expressed below and in Google's prior filings, and based on the trial record, the SSO of the 37 API packages is not copyrightable. This conclusion of law is independently supported by (1) Oracle's concession that the Java language is free and open for anyone to use; (2) the "system" and "method of operation" exclusions in section 102(b) of the Copyright Act; (3) the "functional requirements for compatibility" interpretation of section 102(b) adopted by the Ninth Circuit in *Sega Enters. Ltd v. Accolade, Inc.*, 977 F.2d 1510, 1522 (9th Cir. 1992); (4) constraints on the SSO imposed by the requirements of the Java language and the merger doctrine; and (5) expectations of developers and industry and the *scenes a faire* doctrine.

II. Google's responses to the Court's questions:

A. Question 1: The vocabulary and grammar of a computer language—as distinct from programs written in the language—are not copyrightable because the Copyright Act protects expression, not vocabulary.

1. Copyright does not protect vocabulary, and that lack of protection extends to the SSO of vocabulary.

Well over a century ago, in a decision that is now codified at 17 U.S.C. § 102(b), the Supreme Court announced a fundamental principle dividing copyright and patent protection: "To give the author of the book an exclusive property in the art described therein, when no examination of its novelty has ever been officially made, would be a surprise and a fraud upon the public. That is the province of letters-patent, not copyright." *Baker v. Selden*, 101 U.S. 99, 102 (1879). Throughout the first phase of this trial, Oracle has repeatedly attempted to circumvent *Baker*'s holding by claiming copyright protection in elements of its overall platform, through claims to the Java vocabulary and vague claims to that vocabulary's "structure, sequence and organization."

The Copyright Act protects expression, not systems for expression. *See* 17 U.S.C. § 102; *Baker*, 101 U.S. at 102. Copyright does not protect vocabulary—for example, it does not protect

names, words or short phrases. 37 C.F.R § 202.1(a); Copyright MSJ Order [Dkt. 433] at 7:25-8:4. Nor does it help Oracle's position to argue that the particular structure, sequence and organization of its vocabulary should somehow be protected, even if the individual elements of that vocabulary are not protected.

First, the "structure" or "organization" of words in relation to other words is not protectable. In any language, words are related to other words, and concepts are related to other concepts. Without such interrelations, a vocabulary has no value. Some words are, by definition, more specific versions of others (e.g., a "plane" is a particular type of "vehicle"), just as a subclass in the Java language is a more specific version of its superclass. Indeed, this is precisely the analogy Dr. Reinhold used in explaining the concept of subclass inheritance. RT 587:10-588:11 (explaining how a "Vehicle" class could be the superclass for the "Car," "Train" and "Plane" classes). Dr. Reinhold similarly explained interfaces, fields and methods by reference to properties of real-world items that people describe with words every day. For example, cars have engines that can be started, horns that the driver can "blow," and lights that can be turned on. RT 588:12-20 (explaining methods that the Car class could have). He explained interfaces by analogy to real-world items, noting that there are things other than vehicles that have horns, and thus it is possible to have an interface that reflects this fact. RT 590:5-17 (discussing "ThingWithHorn" interface). The "interdependencies" that Oracle has relied upon so frequently are, in the end, nothing special, and nothing protectable under copyright law. In any useful vocabulary, there are similar relationships that can be drawn, as Dr. Reinhold's analogies demonstrate. Thus, the placement of API elements in packages or classes, and the hierarchical arrangement of those elements via subclassing and interface implementation, all in accordance with the requirements of the Java language, does nothing to make them more copyrightable than any words or short phrases in the English language.

Second, the "sequence" of the elements in the API does not amount to creative expression. Nothing in the record suggests that Google copied the sequence in which the APIs are implemented within the source code. For example, the methods in java.lang.Math are implemented in entirely different orders in Android and J2SE. *Compare* TX 47.101 (Android

Gingerbread version of "Math.java") *with* TX 623.101 (J2SE 5.0 version of "Math.java"). The first ten public methods declared in the Android version are:

double abs(double d)
float abs(float f)
int abs(int i)
long abs(long l)
double acos(double d)
double asin(double d)
double atan(double d)
double atan2(double y, double x)
double cbrt(double d)
double ceil(double d)

See TX 47.101 at lines 61, 83, 100, 113, 133, 151, 171, 211, 230, 250. The first ten public methods declared in the J2SE version are very different:

double sin(double a)
double cos(double a)
double tan(double a)
double asin(double a)
double acos(double a)
double atan(double a)
double toRadians(double angdeg)
double toDegrees(double angrad)
double exp(double a)
double log(double a)

See TX 623.101 at lines 103, 118, 135, 153, 169, 186, 200, 216, 236, 257. The methods are ordered *alphabetically* in the documentation for both Android and J2SE, but that "choice" is so unoriginal that it is not protectable. *See Feist Pubs., Inc. v. Rural Tele. Serv. Co.,* 499 U.S. 340, 362 (1991) (alphabetical list of subscribers is "devoid of even the slightest trace of creativity").

Third, even if the SSO of a vocabulary could theoretically be protected—and on this record, there is no evidence suggesting that *Oracle's* SSO should be protected—the SSO would only be protected *as a whole.* "No matter how original the format, however, the facts themselves do not become original through association." *Feist,* 499 U.S. at 349. Oracle has not argued, however, that Google copied its SSO *as a whole.* Instead, it argues that Google adopted *part of* that SSO, and integrated it with further API packages of Google's own design. Indeed, that fact—the alleged "fragmentation" caused by Android—is the crux of Oracle's claim of harm. That is, Oracle complains that Google *did not* adopt the SSO of Oracle's 166 API packages, but instead forged forward with a *different* SSO of Google's own design. Google was free to do so:

"Notwithstanding a valid copyright, a subsequent compiler remains free to use the facts contained in another's publication to aid in preparing a competing work, so long as the competing work does not feature *the same* selection and arrangement." *Feist*, 499 U.S. at 349 (emphasis added).

2. The ECJ's decision in *SAS Inst. Inc. v. World Programming Ltd* supports the conclusion that the vocabulary of a programming language cannot be copyrighted.

Interpreting the same broad legal principles that are at issue in the present case, the European Court of Justice recently held that computer programming languages are not copyrightable. *SAS Inst. Inc. v. World Programming Ltd*, Case C-406/10 (ECJ May 2, 2012) [Dkt. 1047-1].[1] One of the questions before the ECJ was whether a "Second Program" that "replicates the functions of [a] First Program" infringes the copyright in the First Program. *Id.* ¶ 28(1). More specifically, the ECJ was asked to consider the situation where the First Program "interprets and executes programs written by users of the First Program in a programming language devised by the author of the First Program and a syntax devised by the author of the First Program" and where the Second Program is "written so as to interpret and execute such application programs using the same keywords and the same syntax." *Id.* ¶ 28(3). This is, in essence, the question in the present case. Developers use the methods, fields, constructors and initializers of the 37 API packages to write programs in the Java language, and Android has been written such that it can interpret and execute those programs to the extent they use methods, fields, constructors and initializers found in the 37 packages. RT 2171:24-2172:11 (Astrachan); RT 2292:25-2293:14 (Mitchell).

The ECJ concluded that "the programming language and the format of data files" are "elements of [the First Program] by means of which users exploit functions of that [First P]rogram." Dkt. 1047-1 ¶ 42. The programming language is not "a form of expression of that program" for purposes of copyright law. *Id.* ¶ 39. Any other conclusion "would amount to making it possible to monopolise ideas, to the detriment of technological progress and industrial

[1] The European Committee for Interoperable Systems—of which Oracle is a member, *see* http://www.ecis.eu/about-ecis/—lauded the decision. *See, e.g.*, Reuters, *EU court limits copyright protection for software* (May 2, 2012) (including quote from Thomas Vinje, a spokesperson for the ECIS, supporting the ECJ's decision).

development." *Id.* ¶ 40.[2]

While the ECJ was applying European law, the same principles apply here. Moreover, the ECJ's conclusion was that the programming language cannot be protected by copyright law, because it is on the unprotectable idea side of the idea/expression dichotomy. It necessarily follows that no copyright protection prevents others from adopting *parts* of a programming language. Similarly, copyright law cannot prevent Google from adopting *parts* of the 166 J2SE API packages.

B. Question 2: If each API method is treated as a program, then Google did not copy anything more than the name and declaration of that program, which is not copyrightable.

Google used nothing more than the name and declaration of each API element. For example, Google used the same method declarations[3]—which necessarily means Google used the same names. A method declaration includes the method's name, the parameters it accepts as input, and the type of thing that it returns. RT 785:25-787:8 (Bloch); *see generally* TX 984 (*The Java Language Specification,* Third Edition) at 209-37. A method declaration can also include modifiers such as "public" or "final." *See* TX 984 at 214-20. In addition, a method declaration can use the term "throws" to indicate "exceptions" (errors) that the method can communicate. *See* TX 984 at 221-23. The method body—sometimes referred to at trial as the *implementation* of the method, *see, e.g.*, RT 790:20-23 (Bloch); RT 1566:23-1567:10 (Schmidt); RT 2186:3-12 (Astrachan)—is "a block of code that implements the method . . . ." TX 984 at 223. Aside from the nine-line rangeCheck method, there is no dispute that Google's implementing code is different from Oracle's. RT 1309:8-1313:11 (Mitchell); RT 2182:13-2183:1 (Astrachan). The declarations are, in essence, the titles of the things they declare. Titles are not copyrightable. 37

[2] That said, if the defendant copied *implementing code* from the First Program, that could constitute infringement. *Id.* ¶ 43 ("it should be made clear that, if a third party were to *procure the part of the source code or the object code* relating to the programming language or to the format of data files used in a computer program, and if that party were to create, *with the aid of that code,* similar elements in its own computer program, that conduct would constitute partial reproduction") (emphasis added).

[3] As the Court has noted, the formal definition of a "method declaration" includes the "method body"—the implementation. TX 984 at 209-10. At trial, however, witnesses excluded the method body from their definition of "method declaration" and distinguished between the "declaration" and the "implementing code."

C.F.R. § 202.1(a).

Moreover, the declarations simply state the functional characteristics of the API elements—the names by which they can be invoked, the parameters they must be given as input, and so on. RT 1773:25-1774:25 (Bornstein); RT 2106:13-2109:2 (Astrachan). In order to create API implementations that are compatible with the J2SE APIs—that is, implementations that will function in the same way when called by code written by developers—these functional elements are precisely the ones that Google needed to use. RT 2159:23-2160:2 (Astrachan). Because the declaration can only be written one way in the Java programming language—because the *idea* underlying the declaration changes if the declaration is changed— any arguable expression in the declaration merges with the underlying idea, and cannot be protected by copyright. *Herbert Rosenthal Jewelry Corp. v. Kalpakian,* 446 F.2d 738, 742 (9th Cir. 1971); *Baker,* 101 U.S. at 104; *Allen v. Academic Games League of Am., Inc.,* 89 F.3d 614, 617-18 (9th Cir. 1996). The elements of the declarations are also functional requirements for compatibility with the J2SE API packages, and thus cannot be protected by copyright for this separate reason. *Sega,* 977 F.2d at 1522 (citing 17 U.S.C. § 102(b)); *see also Lotus Dev. Corp. v. Borland Int'l, Inc.,* 49 F.3d 807, 815 (1st Cir. 1995), *aff'd by an equally divided court,* 516 U.S. 233 (1996).

C. Question 3: The form of the fully qualified names of the methods in the 37 API packages is "package.class.method," where the package names start with "java." or "javax." and this form is required by the syntax of the Java language.

The fully-qualified name of a method has three parts: its package name, its class name and its method name. RT 770:1-4 (Bloch). For example, the "cos" method (for calculating the cosine of an angle) in the "Math" class in the "java.lang" package would be named "java.lang.Math.cos". RT 770:5-9 (Bloch). This format is required by the Java language. RT 770:10-12 (Bloch); TX 984 at 126-38.

D. Question 4: Google could have come up with at least some different names and SSO yet still provided similar functionality in Android, but this would not have been consistent with industry custom and developer demand.

It would have been possible in many instances for Google to have created APIs with different names and/or SSO that would have provided similar functionality. That, however,

would have effectively undermined the ability to utilize the Java language, which language provides a familiar environment for developers. By analogy, it is technically possible to create an alternative version of the English language that follows the same rules of grammar, but in which all the *words* are different. While such a scenario is possible, it would make little sense, and would be contrary to the expectations of any reasonable person conversant in English. Moreover, even if Google had done so, it still would have had to implement at least the parts of the 37 packages that are undisputedly *required* in order to implement the Java language, as explained below.

1. Many of the elements of the 37 API packages are *required* by the Java language.

In order to implement the Java language, Google was required to include many elements of the 37 API packages. Dr. Reinhold admitted that at least 61 classes from the 37 API packages are *required* by the Java language specification. RT 684:14-685:2; TX 1062 (Reinhold summary); *see also* RT 1286:14-22 (Mitchell) (agreeing with Dr. Reinhold's analysis). These 61 classes are not merely mentioned by the language specification—"they are part of the specification rather than being part of an example." RT 677:15-16 (Reinhold); *see also* RT 679:18-21. In addition, Oracle's Java language compiler depends on the presence of over 30 classes, including several that are not included in Dr. Reinhold's list of the 61 required classes in TX 1062. *See* RT 679:22-681:21 (Reinhold); TX 1063 (Reinhold summary).

Although the Java language specification *requires* the presence of these classes, it does not specify their details. RT 679:20-21 (Reinhold). Instead, those classes are specified in the *API* specifications, which specify that those classes require over 750 methods and fields. RT 776:21-777:9 (Bloch). Moreover, due to dependencies, implementing those 750 classes, method and fields, requires implementation of 177 classes, with over 2,000 public methods and fields, spread across ten of the accused API packages. RT 779:13-780:15 (Bloch).

Thus, *at least* portions of 10 of the 37 accused API packages must be implemented simply to implement the Java language as required by the language specification, including both classes required by the language specification, and those that the required classes depend upon.

2. Industry custom and developer demand require implementation of the 37 API packages.

Without the APIs, the Java language is a "primitive thing." RT 686:15 (Reinhold). Without APIs, the Java language can be used to "waste time," but "that's pretty much it." RT 782:9-11 (Bloch); *see also* RT 782:12-783:18 (Bloch); RT 683:14-684:4 (Reinhold).

Developers expect the presence of the 37 API packages when they write programs in the Java language. RT 2202:6-11 (Astrachan); RT 1782:6-1783:10 (Bornstein). Those packages are needed in order to meet industry expectations. RT 2203:11-15 (Astrachan); RT 2291:1-8 (Mitchell); RT 519:21-23 (Screven). Indeed, programmers often memorize the names and organization of members of these packages in order to help them write programs more efficiently. RT 767:1-17 (Bloch); RT 2169:25-2170:13 (Astrachan); RT 2289:24-2290:3 (Mitchell). The 37 API packages are also necessary to make practical use of the language. RT 2196:7-2201:17 (Astrachan).

Sun recognized these facts, and indeed promoted widespread use of the Java language and APIs. *See* RT 1957:24-1958:4, 1961:13-19, 1962:2-9 (Schwartz); RT 1474:24-1475:10, 1477:2-1478:9 (Schmidt). Sun worked hard to dispel any suggestion that the SSO of the 37 API packages was proprietary or protected. RT 1966:1-12 (Schwartz). Sun's goal was to ensure that the Java language was widely adopted by encouraging its teaching in colleges and universities. RT 1476:9-14 (Schmidt); RT 1958:5-20 (Schwartz). Java language developers have always understood that the Java API packages, along with the Java language, are free to use. RT 962:4-14 (Swetland); RT 861:9-23 (Lindholm); RT 1769:18-1770:1 (Bornstein).

In view of this undisputed testimony, the APIs in the 37 packages are the Java language equivalents of *scenes a faire*—and therefore uncopyrightable. *See Computer Assocs. Int'l, Inc. v. Altai, Inc.*, 982 F.2d 693, 706-10 (2d Cir. 1992)*; Swirsky v. Carey*, 376 F.3d 841, 850 (9th Cir. 2004); *Sega*, 977 F.2d at 1524 (constraints on the defendant are relevant to whether copyright protection allowed); *Gates Rubber Co. v. Bando Chem. Indus., Ltd*, 9 F.3d 823, 838 (10th Cir. 1993).

3. When developing new APIs in the Java language, it is standard practice to build upon the standard J2SE APIs.

Dr. Reinhold testified that even when people design their own APIs, "[t]hey are building typically *on top of all of the standard Java APIs* and creating their own APIs for whatever problem they are trying to solve." RT 685:12-14 (emphasis added). Thus, for example, when Wall Street firms create their own APIs for financial trading, "those are *strictly built on top of all of this Java platform stuff* we have been speaking about." RT 685:18-20 (emphasis added).

4. The Court should disregard Mr. Ellison's testimony about Spring.

When Larry Ellison was asked whether the Java APIs are needed to use the Java language, Google objected on the ground that the question called for expert testimony. RT 290:15-19. The Court overruled the objection, but only after Mr. Ellison assured the Court that he was testifying based on personal knowledge. RT 290:20-24. Mr. Ellison then testified that a UK company named Spring had built its "own Java environment" called Spring, which used the Java language, but not the Java APIs. RT 290:25-291:6.

Mr. Ellison's testimony was incorrect. The Spring framework is open source software, and the documentation for the Spring framework is readily available on the Internet.[4] This documentation demonstrates that Mr. Ellison's testimony was incorrect, and that the Spring framework *uses the J2SE APIs.* For example, the Spring package "org.springframework.ui" has a class named "ModelMap," which is a subclass of "java.util.HashMap"[5]—a class that is in the accused java.util package. This Spring class implements the interfaces Serializable (part of the accused java.io package), Cloneable (part of the accused java.lang package) and Map (part of the accused java.util package).[6]

This evidence, which flatly contradicts Mr. Ellison's testimony, is not in the trial record. The accuracy of the cited documentation, however, cannot reasonably be questioned under the circumstances, and thus the Court may take judicial notice of these facts. Fed. R. Evid. 201(b)(2). Moreover, if the Court requests, Google will submit a declaration from Professor Astrachan

[4] *See* http://static.springsource.org/spring/docs/2.0.x/api/index.html.

[5] *See* http://static.springsource.org/spring/docs/2.0.x/api/org/springframework/ui/ModelMap.html.

[6] *See id.*

explaining that the source code for the Spring framework depends on no less than 20 of the 37 accused J2SE API packages. In light of these facts, Google requests that the Court not rely on any of Mr. Ellison's unsupported testimony about Spring. *See* RT 290:25-291:6, 304:13-22.

E. Question 5: The input-output scheme of a method is not copyrightable.

The input-output scheme of a method cannot be copyrighted because it represents an *idea*, not creative expression. 17 U.S.C. § 102(b). Moreover, the input-output scheme is required for compatibility with code that "calls" or "invokes" the method, which is another reason it cannot be protected by copyright. *See supra*, Part II.B; *Sega*, 977 F.2d at 1522.

In addition, there are only a limited number of ways in which to devise an input-output scheme for any given method. Independent of *Sega*, the merger doctrine bars copyright protection for input-output schemes. *See Allen*, 89 F.3d at 617-18; *Engineering Dynamics, Inc. v. Structural Software, Inc.*, 46 F.3d 408, 409-10 (5th Cir. 1995), *clarifying* 26 F.3d 1335 (5th Cir. 1994); *see also* Google 4/3/12 Br. [Dkt. 852] at 8:19-9:20.

F. Question 6: The "core" packages in 1996 included *at least* java.lang, java.io and java.util, and today include at least 34 of the accused packages.

Google agrees that *at least* java.lang, java.io and java.util were "core" packages *in 1996.* Sun's documentation from that time stated that java.lang, java.io and java.util "must be included in all general purpose Java systems." TX 2564 (*The Java Language Specification*, First Edition) at 31. This same book referred to those three packages as "core packages." TX 2564 at 23. Another Sun book published in 1996, *The Java Application Programming Interface, Volume 1*, described those packages as "the foundation of the Java language" and as "general purpose libraries fundamental to every Java program." TX 980 at 528. This book defined the "core packages" to be those three packages, plus java.net. TX 980 at xix, back cover. As the Court has recognized, the case against copyright protection and for fair use for the "core" packages is even stronger than it is for the rest of the accused packages. RT 3388:12-3389:6, 3389:21-3390:11.

Later versions of J2SE, however, recognize many more packages as "core" packages. For example, the documentation for J2SE 1.4 defines the "core" packages to include *all but four of the accused packages*—every accused package except java.lang.annotation, java.security.acl,

java.sql and javax.sql. *See* TX 622 (source code for J2SE 1.4, which also includes documentation).[7]

The first of the four remaining API packages, java.lang.annotation, was not introduced until J2SE 5.0. It is one of the few packages that is *directly required* by the third edition of *The Java Language Specification*, which includes an entire section about "Annotations." TX 984 at 270-86. This section states that "[t]he direct superinterface of any annotation type is always [java.lang.][8]annotation.Annotation"—i.e., the Annotation interface in the java.lang.annotation package. *Id.* at 272. The section also discusses four predefined annotation types—Target, Retention, Inherited and Override—that are part of the java.lang.annotation package. *Id.* at 277-79. For J2SE 5.0, java.lang.annotation is unquestionably a core package.

This leaves three accused packages, java.security.acl, java.sql and javax.sql. The first of these, java.security.acl, appears to have been left out of the list of "core" packages in J2SE 1.4 only by accident. Each of the other subpackages of java.security (java.security.cert, java.security.interfaces and java.security.spec) is listed. *See* TX 622.[9]

The final two packages, java.sql and javax.sql, are not identified as "core" packages in the J2SE 1.4 documentation. However, these two packages cover such basic concepts for data storage and retrieval that they should be considered, as a practical matter, to be required elements of the Java language. RT 2198:25-2199:16 (Astrachan).

G. Question 7: The Java language requires more than just the java.lang, java.io and java.util packages.

The Java language requires methods, classes and packages beyond java.lang, java.io and java.util. This is because the 61 classes that are *directly* required in order to implement the Java language themselves are dependent on other classes. The full chain of interdependencies requires

[7] The relevant documentation can also be accessed on the web at http://docs.oracle.com/javase/1.4.2/docs/guide/core/index.html. Following the links on that page leads to webpages that include lists of API packages. There does not appear to be an analogous list present in the J2SE 5.0 documentation.

[8] In *The Java Language Specification*, where no package is expressly identified, "the intended reference is to the class or interface . . . in the package java.lang." TX 984 at 6.

[9] For the Court's convenience, the specific page is also available on the web at http://docs.oracle.com/javase/1.4.2/docs/guide/security/index.html.

over 177 classes, with over 2,000 public methods and fields, drawn from ten of the accused API packages. *See supra,* Part II.D.1.

H. Question 8: All 37 accused API packages should be deemed "core"packages.

Based on the documentation for J2SE 1.4, and the third edition of *The Java Language Specification,* 34 of the accused API packages are concededly core packages. One of the remaining three, java.security.acl, appears to have been left off of the list of core packages in J2SE 1.4 only by mistake. The final two packages, java.sql and javax.sql, cover such fundamental concepts for modern software applications that they, too, should be deemed core packages. *See supra,* Part II.F.

I. Question 9: There are cross-method, cross-class interdependencies *at the implementation level* in J2SE, and those implementation level interdependencies are not always duplicated in the Android implementations.

The Court asked the parties to address, in their April 22, 2012 briefs, whether "any of the Sun compiled lines in the 37 APIs call upon part or all of another API as a step" and, if so, whether Android's implementing code "likewise call upon the same other API." *See* Order re Brief Due Sunday [Dkt. 951] at 1. Google's brief responded that, yes, the J2SE implementations do reference other methods and classes, but, no, the Android implementations do not necessarily follow the same pattern. *See* Dkt. 955 at 13:8-14:6. Oracle's brief addressed references in class, method, and field *declarations* (which are necessarily similar for compatibility reasons), rather than references in the *implementations* (which can be different while remaining compatible). *See* Dkt. 956 at 13:15-14:10.

At trial, there was no testimony on these points. The source code that is in evidence in native format, however, confirms that the positions Google advanced in its April 22. 2012 brief—positions Oracle has not denied—are correct. *See* TX 623 (J2SE 5.0 source code), TX 46 (Android, "Froyo" release, source code). If the Court requests, Google will submit a declaration attaching printed excerpts from those trial exhibits demonstrating that, for example, Android's implementation of the URL class in the java.net package makes use of the ObjectOutputStream class from the java.io package, while the J2SE implementation makes use of the OutputStream class from the java.io package.

J. Question 10: At the name/declaration level, the only interdependencies in the Java language come from (1) the package/class/member organizational scheme; (2) inheritance via subclassing and subinterfacing; and (3) interface implementation.

There are three types of interdependencies at the name/declaration level. First, there is the organizational scheme that groups class members (such as methods) into classes, and classes into packages. Second, classes and interfaces can inherit characteristics from their superclasses and superinterfaces. Third, classes can implement interfaces. RT 2187:18-2188:24 (Astrachan); RT 584:8-603:6 (Reinhold); TX 1028 (key, showing only class/package, interface, and subclass/superclass relationships). These are the only name/declaration level interdependencies that were identified at trial.

K. Question 11: The Seventh Circuit's decision in *American Dental Ass'n v. Delta Dental Plans Ass'n,* was either wrongly decided, or unclear about the scope of its holding.

The Seventh Circuit's *American Dental Ass'n* decision purported to address whether the ADA's "taxonomy" for dental procedures could be copyrighted. 126 F.3d 977, 977 (7th Cir. 1997). But in concluding that the "taxonomy" was expressive, the Seventh Circuit relied on the *text descriptions* the ADA employed. *See id.* at 979. That suggests that the court's decision was about the copyrightability of the ADA's *book describing the taxonomy,* not the taxonomy separate and apart from those descriptions. If so, the decision is not relevant to the copyrightability determination that the Court must make in this case. If, however, the Seventh Circuit did intend to hold that the taxonomy itself was copyrightable, its reliance on text descriptions of the taxonomy to conclude that the taxonomy itself was copyrightable was nonsensical, and renders the decision of suspect persuasive value.

No Ninth Circuit case has ever cited *American Dental Ass'n*. Indeed, the Seventh Circuit's decision has only been cited three times by *any* court. The Seventh Circuit itself has cited the *American Dental Ass'n* decision only once, and only for propositions unrelated to the "taxonomy" holding. *Seng-Tiong Ho v. Taflove,* 648 F.3d 489, 497-500 (7th Cir. 2011). The Third Circuit has cited the decision once, but found the facts of the case before it factually distinguishable. *See Southco, Inc. v. Kanebridge Corp.,* 258 F.3d 148, 155 (3d Cir. 2001).

Finally, the Sixth Circuit *criticized* the decision, calling the Seventh Circuit's reasoning "opaque." *ATC Distrib. Grp., Inc. v. Whatever It Takes Trans. & Parts, Inc.*, 402 F.3d 700, 708 (6th Cir. 2005).

Even if the Court were to assume that a taxonomy can be copyrighted—and to do so would be legal error, because the ADA taxonomy is an unprotectable system, *see* 17 U.S.C. § 102(b)—that does not mean that the SSO of the 37 API packages should be protected by copyright. The Seventh Circuit rejected the argument that the "taxonomy" was an unprotectable system, noting that "[t]his taxonomy does not come with instructions for use, as if the Code were a recipe for a new dish." 126 F.3d at 980. The 37 API packages, however, *do* come with instructions for use—that is precisely what the API specifications are. The APIs in the 37 accused packages are a system by which Java language developers express themselves, and as such are an uncopyrightable system. 17 U.S.C. § 102(b).

Finally, although the Seventh Circuit *purported* to hold that a "taxonomy" can be copyrighted, nothing in the decision supports the conclusion that a numbering system can be copyrightable separate and apart from descriptions of the parts of the system. In *American Dental Ass'n*, the Seventh Circuit ultimately held only that the defendant could not copy "the Code itself" or distribute derivative works based on "the Code." 126 F.3d at 981. But the opinion earlier defined "the Code" to be *a book. See id.* at 977 ("The American Dental Association has created the *Code on Dental Procedures and Nomenclature.* The first edition was published in 1969; the Code has been revised frequently since, in response to changes in dental knowledge and technology."). The court also referred to "the numbering system and short descriptions from the ADA's Code." *Id.* If the phrase "the Code" referred to the numbering system itself, separate from the descriptions, then the reference to "the numbering system . . . from the ADA's Code" would mean "the numbering system . . . from the ADA's numbering system," which is recursively meaningless, referring to "the numbering system" as something that can be extracted from "the numbering system."

Thus, the better reading of the decision is that "the Code" refers to *the ADA's book describing the numbering system*, complete with "the short descriptions" of each procedure. *See*

id. So understood, the Seventh Circuit's holding that "the Code" is copyrightable, *see id.* at 979, means only that the ADA's *book describing its numbering system* is copyrightable. If that is the extent of the Seventh Circuit's holding, then the decision is irrelevant to the present case. If the Seventh Circuit instead intended to hold that the *numbering system*, separate from its descriptions, was protected by copyright, the decision is unclear, poorly reasoned, and should not be followed absent Ninth Circuit precedent for doing so—and there is none.

L. Question 12: *CDN, Inc. v. Kapes* was not about a method of operation or system.

In *CDN, Inc. v. Kapes*, the Ninth Circuit did not address whether the SSO of CDN's price list was protectable, because there was no allegation that the SSO had been copied. 197 F.3d 1256, 1259 (9th Cir. 1999) ("CDN does not allege that Kapes copied the entire lists, as the alleged infringer had in *Feist*. . . . Thus Kapes' argument that the selection is obvious or dictated by industry standards is irrelevant."). The case therefore sheds no light on whether the SSO of the 37 API packages is copyrightable.

Instead of SSO, the issue in *CDN* was whether CDN's estimates of coin prices were "sufficiently original *as compilations* to sustain a copyright." *Id.* (emphasis added). The Ninth Circuit held that CDN relied on pricing information provided by others (i.e., that the coin prices in its guide were not of its own creation). *Id.* at 1260. CDN then *chose* which coin prices to *keep*, "retain[ing] only that information it considers to be the most accurate and important." *Id.* "The prices CDN creates are compilations of data that represent its best estimate of the value of the coins." *Id.*

This theory of copyrightability is not available to Oracle. In its April 3, 2012 brief, Oracle denied that its APIs were a compilation. *See* Oracle 4/3/12 Br. [Dkt. 853] at 1:7-8 ("The 37 APIs should not be viewed as a compilation under section 101 of the Copyright Act."). After briefly attempting to switch course, Oracle again conceded that it was not arguing for protection under a collective work theory. *See* RT 2134:11-17. At no time since has Oracle argued that the elements of the 37 API packages should be protected under a compilation theory.

Moreover, *CDN* did not address section 102(b)'s system and method of operation

exclusions, which are central to the Court's copyrightability determination in this case. The Ninth Circuit opened its decision by stating, "We must decide whether prices listed in a wholesale coin price guide contain *sufficient originality* to merit the protection of the copyright laws." 197 F.3d at 1257 (emphasis added). However, Kapes also argued that CDN's prices were unprotectable *ideas. See id.* at 1261. The Ninth Circuit disagreed, holding that CDN's "prices fall on the expression side" of the idea/expression dichotomy. *Id.* at 1262.

CDN, however, did not seek copyright protection for a *system* of coin pricing, or a *method of operation* for coin pricing. Had it done so, its argument would have failed, because CDN could not "claim protection for its idea of creating a wholesale price guide . . . ." *Id.*; *see also* 17 U.S.C. § 102(b). Moreover, Kapes did not argue that functional requirements for compatibility compelled him to use the same coin prices that CDN did. *See Sega*, 977 F.2d at 1522.

M. Question 13: Android is compatible with the 37 API packages at issue because code written to use those APIs will compile and work properly on an Android device.

The Court has asked the parties to address what they mean when they refer to "compatibility." Android is compatible with the APIs from the 37 API packages and, as a necessary part of that compatibility, it has substantially the same SSO as the 37 API packages.

Compatibility is not an all or nothing proposition. Two things can be compatible in some respects, but not others. The key issue in the present case is not whether Android is fully compatible with J2SE in all respects, but whether it is compatible with the APIs in the 37 API packages in the computer science sense—not "compatible" in Sun's or Oracle's business plan sense. The parties' experts agreed that the platforms are compatible from the perspective of computer science, because code written using the APIs in those packages will work on both platforms. RT 2171:24-2172:11 (Astrachan); RT 2292:25-2293:14 (Mitchell). For example, Professor Astrachan wrote the following program during trial:

```
package simple;
/** @author ola */
public class WebReader {

    public static void main (String[] args) {
        java.net.URL site=new java.net.URL("http://cnn.com");
        java.io.InputStream source=site.openStream();
```

```
            System.out.print(source.read());
        }
    }
```

TX 3536 (demonstrative); RT 2162:12-2167:25 (Astrachan). This code defines a package named "simple," which includes a class named "WebReader." RT 2163:5-22. Within that class is a short method that (1) creates a "URL object" associated with the URL http://cnn.com; (2) creates an "InputStream" object by opening a stream to the CNN website; and (3) displays the first character from that website. RT 2165:17-2167:25, 2170:22-2171:3. This program depends upon elements and invokes methods from three of the accused API packages, java.net, java.io and java.lang. RT 2169:20-24. Because the methods that the program invokes and the API elements it relies on are implemented both in J2SE and Android, this program is compatible with both of those platforms. RT 2171:19-23; *see also* RT 2292:25-2293:8 (Mitchell) (the three lines of code Professor Astrachan wrote that use the J2SE APIs would work on both the J2SE and Android platforms). This understanding of "compatibility" is not a position adopted just for this litigation. Indeed, Oracle's expert called this definition of compatibility "a great definition of 'compatible' . . . ." RT 2293:9-14 (Mitchell).

N. Questions 14-15: In the Java language, "inheritance" is a concept applicable to classes, not packages; by virtue of the Java language specification, a subclass inherits fields, methods, and other members from its superclass.

In the Java language, if "you define one class to be a subclass of another, then the subclass inherits all the methods of the superclass . . . ." RT 1225:13-15 (Mitchell); *see also* RT 2188:10-11 (Astrachan). The inheritance relationship exists at the class level. RT 2243:6-7 (Astrachan) ("the class declaration, which shows the inheritance relationships and the interface relationships"). There is no such thing as "inheritance" of a package. *See* TX 984 at 617 (index; no discussion of "package" under "inheritance"); *see also id.* at 154 (recognizing that hierarchical naming structure for packages "has no significance in itself" and that similarly-named packages have no "special access relationship "to each other").

Inheritance is a characteristic of a class or interface that results from the superclass/subclass and superinterface/subinterface relationships that are part of the Java language. *See* TX 984 at 188, 190. A class or interface inherits "public" members (fields,

methods, etc.) of its superclass or superinterface, as well as any members declared in the class or interface itself. *See id.*

O. Question 16: Copyright protection does not extend to names, including fully qualified names, nor does it extend to input-output (argument-return) designations, exception types, subclass inheritances, or interface relationships.

The Court has asked what Google allegedly copied aside from names (including fully-qualified names) and input-output (argument-return) designations. In addition to those items, the SSO of the Android APIs for the 37 packages at issue adopts substantially the same "exceptions" for methods (i.e., the methods "throw" or generate the same error messages), subclass and subinterface structures for classes and interfaces (i.e., inheritance) and interface implementations for classes. However, the exception, inheritance and interface relationships are all part of the "ideas" underlying the defined methods, classes and interfaces. These relationships are therefore uncopyrightable by virtue of the merger, *scenes a faire* and/or functional requirements for compatibility doctrines, and/or the system and method of operation exclusions of section 102(b) of the Copyright Act.

III. The Court should hold that the SSO of the 37 API packages is not copyrightable.

The parties have submitted voluminous briefing on this legal issue, and much of the evidence during Phase One of the trial related to the issue of copyrightability. There are many reasons why the Court should conclude that the SSO of the 37 API packages is not copyrightable, but the Court has no doubt read the prior briefs carefully, and Google will not repeat each of its arguments here once more.

The simple point, however, is that Oracle seeks to misuse copyright to control a language—to restrict dramatically the right to use the established vocabulary of a freely-available language and to require that others who want to use the language create an entirely new vocabulary. Whether the APIs at issue are part of the "Java language" is important to deciding the effect of Oracle's concession that it does not assert copyright protection in this case over the Java language itself. Regardless of how the Court resolves that point, however, the APIs are part of *a* language. There is no dispute that when Java developers use the APIs, they do so to express themselves. Indeed, without APIs, the Java language is, in all practical effect, useless. RT

683:14-684:4, 707:18-21 (Reinhold); RT 782:9-14 (Bloch); RT 1477:2-13 (Schmidt); RT 1960:4-8 (Schwartz).

Oracle argues that developers can create *new* APIs, and then use those new APIs *instead* of the J2SE APIs, and thus can still express themselves in the Java language without using the J2SE APIs. That is, at least to some extent, wrong, because the Java language cannot be implemented without at least *some* of the APIs at issue. RT 1274:16-24 (Mitchell); RT 684:16-685:2, 679:12-21 (Reinhold); RT 776:12-777:9, 777:19-778:9, 779:13-780:18 (Bloch). But it is also irrelevant. The language that developers use when they write programs in the Java language with the J2SE APIs—whether that language is called the "Java language" or "the Java language with the J2SE APIs"—is a functional system that can be used for expression and is known to millions of developers. That system cannot be protected by copyright. 17 U.S.C. § 102(b).

Allowing Oracle to use copyright law to control the J2SE APIs would in effect grant Oracle a monopoly, allowing it to prevent millions of Java developers from using their skills to write applications for platforms that Oracle has not approved. This goes beyond protecting *Oracle's* expression, and would grant Oracle control over how *developers* are allowed to express themselves.

It is no answer to say that there are *other* systems that developers can use to express themselves. Of course that is true—we know it is true, because there are many different programming languages that are the result of many different design choices made by language designers. But section 102(b) does not exclude systems from copyright protection only when there are no alternatives. It states that "[i]n no case" shall copyright protection extend to a system. 17 U.S.C. § 102(b).

And there is a strong policy reason for that stark prohibition: choosing one system over another is not a matter of expression, it is a matter of choosing one idea over another. The J2SE APIs include a "collections framework" that was designed by Josh Bloch at least in part when he was a Sun employee. RT 750:5-7, 750:16-751:4 (Bloch). The approach Dr. Bloch adopted for the collections framework was, in at least one person's estimation, life changing. RT 750:8-12 (Bloch). The notion of "collections" was not new—other, older languages implemented

collections frameworks of their own, which conceptualized collections in different ways. RT 1242:16-1243:3 (Mitchell). The reason Dr. Bloch's collections framework was life-changing was that, at least for that one developer, Dr. Bloch's idea was better than those that had come before. His collections framework was a better mousetrap—a better idea.

Because copyright does not grant a monopoly over ideas, when someone comes up with a good idea, others are free to adopt it absent some other legal principle—such as patent law—protecting the idea. Good ideas rarely come easily. Indeed, they may take years of effort and experimentation. They are still, however, ideas, and still unprotected by copyright law.

The record shows that the SSO of the 37 packages—to the extent it includes anything other than the "structure" imposed by the Java language hierarchical naming rules—is a method of operation, a system of communicating with software, and functionally required for compatibility. The record also shows that the form of expression in the source code that reflects the SSO is constrained by the requirements of the language and therefore unprotectable under the merger doctrine. Finally, the record shows that any arguable expression in the SSO has come to be expected by programmers and industry, is required for compatibility purposes and reflects widely-accepted programming practices—and cannot therefore form the basis for an infringement action. For all the reasons given in its prior briefs, as well as the reasons given above, Google respectfully requests that the Court hold that the SSO of the 37 API packages at issue is not copyrightable.

Dated: May 10, 2012

KEKER & VAN NEST LLP

/s/ Robert A. Van Nest

By: ROBERT A. VAN NEST

Attorneys for Defendant
GOOGLE INC.