MORRISON & FOERSTER LLP
MICHAEL A. JACOBS (Bar No. 111664)
mjacobs@mofo.com
KENNETH A. KUWAYTI (Bar No. 145384)
kkuwayti@mofo.com
MARC DAVID PETERS (Bar No. 211725)
mdpeters@mofo.com
DANIEL P. MUINO (Bar No. 209624)
dmuino@mofo.com
755 Page Mill Road, Palo Alto, CA 94304-1018
Telephone: (650) 813-5600 / Facsimile: (650) 494-0792

BOIES, SCHILLER & FLEXNER LLP
DAVID BOIES (Admitted *Pro Hac Vice*)
dboies@bsfllp.com
333 Main Street, Armonk, NY 10504
Telephone: (914) 749-8200 / Facsimile: (914) 749-8300
STEVEN C. HOLTZMAN (Bar No. 144177)
sholtzman@bsfllp.com
1999 Harrison St., Suite 900, Oakland, CA 94612
Telephone: (510) 874-1000 / Facsimile: (510) 874-1460

ORACLE CORPORATION
DORIAN DALEY (Bar No. 129049)
dorian.daley@oracle.com
DEBORAH K. MILLER (Bar No. 95527)
deborah.miller@oracle.com
MATTHEW M. SARBORARIA (Bar No. 211600)
matthew.sarboraria@oracle.com
500 Oracle Parkway, Redwood City, CA 94065
Telephone: (650) 506-5200 / Facsimile: (650) 506-7114

Attorneys for Plaintiff
ORACLE AMERICA, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

ORACLE AMERICA, INC. Plaintiff, v. GOOGLE INC. Defendant.	Case No. CV 10-03561 WHA **ORACLE'S MARKUP OF COURT'S BACKGROUND SECTION** Dept.: Courtroom 8, 19th Floor Judge: Honorable William H. Alsup

Oracle respectfully submits this markup of the Court's proposed background section, after having consulted with its technical staff. Two versions are attached for the Court's convenience: both redline and clear text.

The Court asked about the difference between statements and expressions. This short description explains the difference between the two:

An *expression* is a (typically) small piece of program text that instructs the computer to calculate a single value (*e.g.*, 1 + 3, 2 * pi * r, Math.max(a, b)). A *statement* can be just an expression, or it can be a control structure that contains one or more expressions or other statements. For example, an "if" statement expresses a conditional action:

```
if (x > y)
    System.out.println("greater");
```

This statement includes an expression (x > y) and another statement (System.out.println("greater")). It instructs the computer to compare the two variables x and y and then print the word "greater" if x is indeed greater than y. Statements are run in the sequence written. Statements are what tell the computer to do work.

Java syntax includes *separators* (*e.g.*, {, }, ;), *operators* (*e.g.*, +, -, *, /, <, >), *literal values* (*e.g.*, 123, 'x', "Foo"), and *keywords* (*e.g.*, if, then, else, while, return). These carry precise predefined meanings. Java syntax also includes *identifiers* (*e.g.*, x, String, java.lang.Object), which are used by developers to name specific values, fields, methods, and classes as described below. These are used to form statements, each statement being a single ~~command executed by the Java interpreter~~directive to take some action. Statements are run in the sequence written. Statements are what tell the computer to do work.

A *declaration* ~~is a type of statement. It~~ instructs the computer to recognize a name as a variable or as a method or as a class and to reserve memory accordingly. The word "declare" is not used to precede a declaration. A declaration is a line (or more) of code that, for example, declares the variable pi to be 3.141592 or declares a class or method ~~in its fully qualified form~~.

~~The next higher level of syntax is the method.~~ A method is a type of declaration. A method is a sequence of statements. Once defined, it can be invoked or "called~~"~~ on" elsewhere in the program. A method is like a subroutine. When a method is called upon, arguments are usually passed over to the method. These are the inputs. The output~~(s)~~ from the method are known as the return value~~s(s)~~. An example is a method that receives two numbers as inputs and returns the greater of the two as the output. Another example is a method that receives an angle expressed in degrees and returns the cosine of that angle. Methods can be much more complicated. A method, for example, could receive the month and day and return the Earth's declination to the sun for that month and day.

A method declaration consists of the method header and the method body. (At the trial, however, witnesses sometimes referred to the method header as the method declaration or the method signature, even though that was not technically correct, and so the Court will use the three synonymously to match the evidence~~signature of the method (discussed below) as the declaration~~.) A method declaration defines the entire routine to be followed when the method is called. A method call is a line of other code somewhere else in the program that calls up (or invokes) the method and specifies the arguments to be passed to the method for crunching. The return is returned for use as the program marches on after the method call.

A method ~~header, also called a method~~ signature~~,~~ consists of the name of the method and the number and types of formal parameters to the method, if any. More specifically, ~~the~~ a method ~~signature~~ header will contain the name of the method; the number, order, type and name of the parameters used by the method; the type of value returned by the method; the checked exceptions that the method can throw; and various method modifiers that provide additional information about the method.

A method body is a block of code that implements the method. If a method is declared to have a return type, then the method body must have a return statement and the statement must ~~be followed by~~ include the expression to be returned when that line of code is reached. During trial, many witnesses referred to the method body as the "implementation." It is the method body that does the heavy lifting, namely the actual work of taking the inputs, crunching them, and returning an answer. (This part was not copied from Java by Google.)

A class is another type of declaration ~~After a method, the next higher level of syntax is the class~~. A class ~~defines~~ may include a collection of fields that hold data values and methods that operate on those values. Classes are a fundamental structural element in the Java programming language. All Java programs are written as ~~a~~one or more classes. ~~All~~ Java statements appear within methods and all methods are implemented within classes. To write a Java program, it must be placed in a class.

A class ~~definition~~ declaration includes the name of the class and other modifiers ~~functional important~~ information that define the class. The body of the class includes fields and methods, constructors and initializers. Classes can have subclasses that inherit the functionality of the class itself while adding ~~specialized~~ functionality for the subclass. When a new subclass is defined, the ~~definition~~ declaration uses the word "extends" to alert the ~~computer~~ compiler that the fields and methods of the ~~patent~~ parent class are inherited ~~extended~~ automatically into the new subclass so that only the additional fields or methods ~~new and specialized codes~~ for the subclass need be ~~stated~~declared.

An interface is a special type of class, which is used to relate similar classes more flexibly than allowed by the strict subclass/superclass hierarchy. An interface contains method

declarations, but the declarations do not have bodies. If a class is declared to "implement" an interface, then for each method in the interface the class must either declare that method or inherit it from a superclass; a class can implement more than one interface.

Classes and interfaces can be grouped into packages in the same way we all group files into folders on our computers.

Here is a simple example that illustrates methods, classes and packages.

```
Package java.lang;
public class Math {
        public static int max (int x, int y) {
            if (x > y) return x ;
            else return y ;
        }
{}
```

A typical program would have more than one method in a class. All Java programs must have ~~a~~ at least one class. All programs, however, need not have packages, which are merely convenient ways to organize the classes.

To invoke this method from a program, the following could be included in the program:

```
int a= Math.max (2, 3);
```

Java syntax includes *separators* (*e.g.*, {, }, ;), *operators* (*e.g.*, +, -, *, /, <, >), *literal values* (*e.g.*, 123, 'x', "Foo"), and *keywords* (*e.g.*, if, then, else, while, return). These carry precise predefined meanings. Java syntax also includes *identifiers* (*e.g.*, x, String, java.lang.Object), which are used by developers to name specific values, fields, methods, and classes as described below. These are used to form statements, each statement being a single directive to take some action. Statements are run in the sequence written. Statements are what tell the computer to do work.

A *declaration* instructs the computer to recognize a name as a variable or as a method or as a class and to reserve memory accordingly. The word "declare" is not used to precede a declaration. A declaration is a line (or more) of code that, for example, declares the variable pi to be 3.141592 or declares a class or method.

A method is a type of declaration. A method is a sequence of statements. Once defined, it can be invoked or "called on" elsewhere in the program. A method is like a subroutine. When a method is called upon, arguments are usually passed over to the method. These are the inputs. The output from the method are known as the return value. An example is a method that receives two numbers as inputs and returns the greater of the two as the output. Another example is a method that receives an angle expressed in degrees and returns the cosine of that angle. Methods can be much more complicated. A method, for example, could receive the month and day and return the Earth's declination to the sun for that month and day.

A method declaration consists of the method header and the method body. (At the trial, however, witnesses sometimes referred to the method header as the method declaration or the method signature, even though that was not technically correct, and so the Court will use the three synonymously to match the evidence.) A method declaration defines the entire routine to be followed when the method is called. A method call is a line of other code somewhere else in the program that calls up (or invokes) the method and specifies the arguments to be passed to the method for crunching. The return is returned for use as the program marches on after the method call.

A method signature consists of the name of the method and the number and types of formal parameters to the method, if any. More specifically, a method header will contain the name of the method; the number, order, type and name of the parameters used by the method; the type of value returned by the method; the checked exceptions that the method can throw; and various method modifiers that provide additional information about the method.

A method body is a block of code that implements the method. If a method is declared to have a return type, then the method body must have a return statement and the statement must include the expression to be returned when that line of code is reached. During trial, many witnesses referred to the method body as the "implementation." It is the method body that does the heavy lifting, namely the actual work of taking the inputs, crunching them, and returning an answer. (This part was not copied from Java by Google.)

A class is another type of declaration . A class may include a collection of fields that hold data values and methods that operate on those values. Classes are a fundamental structural element in the Java programming language. All Java programs are written as one or more classes. Java statements appear within methods and all methods are implemented within classes. To write a Java program, it must be placed in a class.

A class declaration includes the name of the class and other modifiers information that define the class. The body of the class includes fields and methods, constructors and initializers. Classes can have subclasses that inherit the functionality of the class itself while adding functionality for the subclass. When a new subclass is defined, the declaration uses the word "extends" to alert the compiler that the fields and methods of the parent class are inherited automatically into the new subclass so that only the additional fields or methods for the subclass need bedeclared.

An interface is a special type of class, which is used to relate similar classes more flexibly than allowed by the strict subclass/superclass hierarchy. An interface contains method declarations, but the declarations do not have bodies. If a class is declared to "implement" an interface, then for each method in the interface the class must either declare that method or inherit

it from a superclass; a class can implement more than one interface.Classes and interfaces can be grouped into packages in the same way we all group files into folders on our computers.

Here is a simple example that illustrates methods, classes and packages.

```
Package java.lang;
public class Math {
        public static int max (int x, int y) {
                if (x > y) return x ;
                else return y ;
        }
}
```

A typical program would have more than one method in a class. All Java programs must have at least one class. All programs, however, need not have packages, which are merely convenient ways to organize the classes.

To invoke this method from a program, the following could be included in the program:

```
int a= Math.max (2, 3);
```

Respectfully submitted,

Dated: May 9, 2012

MORRISON & FOERSTER LLP

By: /s/ Michael A. Jacobs

Attorneys for Plaintiff
ORACLE AMERICA, INC.