1.05 Desk Check: HelloWorld.java

The source code for the "Hello World" program is shown again below; however, notice that line numbers have been added. Line numbers will never appear in an actual program, but they will be included in desk checks to direct your attention to specific parts of a program. (If line numbers were included in the actual source code, an error message would be generated.) The highlighting is also added to point out important information.

```
< 1>  /**
< 2>  * Display a "Hello World!" message on the screen
< 3>  *
< 4>  */
< 5>  public class HelloWorld
< 6>  {
< 7>        public static void main(String[] args)
< 8>        {
< 9>              System.out.println("Hello, Virtual World!");
<10>        }//end of main method
<11>  }//end of class
```

As you start to examine this program, don't focus on the individual lines; look for a structural pattern first. (You already noticed that the program resembles an outline.) The upper section of code contains asterisks (*) while the bottom section does not. Now that the big picture is coming into focus, it's time to pay attention to the details.

The **highlighted** code designates reserved words in Java's vocabulary, which are always written in lowercase. Java is case-sensitive, so Class and class are not the same. Watch out for case mistakes, they will be one of the most frequent sources of typographical error. Next, let's dissect the code line-by-line.

Line	**Purpose**
< 1>	Beginning of comment section.
< 2>	Comment explaining purpose of the program.
< 3>	Blank comment creating white space.
< 4>	End of comment section.
< 5>	Class declaration giving the program a name.
< 6>	Curly brace marking the beginning of the class (matches up with Line <11>).
< 7>	Main method establishing where program execution begins.
< 8>	Curly brace marking the beginning of the class (matches up with Line <10>).
< 9>	Print statement to display "Hello, Virtual World" on screen.
<10>	Curly brace marking the end of the main method (matches up with Line <8>).
<11>	Curly brace marking the end of the class (matches up with Line <6>).

There is a specific purpose for every line of code. Each line must follow Java's syntax rules and appear in the proper sequence. Now we can dig a little deeper.

Lines <1> through <4> are comments. All comments in a program are ignored by the

computer; it's as if they weren't even there (see Java Methods Section 5.2). If the computer ignores comments, why are they included? Comments provide internal documentation that programmers often need in order to understand the purpose of a program and details about how it works.

Line <5> declares HelloWorld to be the name of the class. Notice that the first letter of any word in the name is capitalized. When this source code is saved, it will automatically be named after the class (i.e. HelloWorld.java).

Lines <6> and <11> are a pair of opening and closing curly braces that indicate where the class begins and ends. Braces always occur in pairs.

Line <7> is the **`main()`** method. This is an important line because program execution begins from the **`main()`** method.

Lines <8> and <10> are another pair of curly braces marking the beginning and end of the **`main()`** method.

Line <9> is a print statement to display "Hello, Virtual World" on the screen using the **`println()`** method. (This is pronounced "print line.") Notice that there are two parts to this statement, inside and outside of the parentheses, and that the line ends with a semicolon (;).

- Anything inside of quotation marks is referred to as a String literal, which in this case is the message "Hello, Virtual World!" that will be printed as output when the program executes. It is important to remember that anything inside quotes in a print statement will be printed exactly (WYSIWYG) as it appears within the quotation marks.
- The code at the beginning of the line (System.out) is Java's way of directing output to the computer screen. The **`println()`** method is a member of the System class.
- Executable Java statements must end with a semicolon (;). Forgetting to include the semicolon on this line will result in a syntax error. Observe, however, that not every line in the program ends in a semicolon. Accidentally including a semicolon when it is not needed is also a syntax error.

This simple program has now been analyzed inside and out. Eleven lines of code to print one message on the screen! Did you think such a simple program would require so much explanation? But now we are actually ready to run it.