Recognizing Syntax Errors

When you make syntax errors in your program the compiler gives error messages and does not create the bytecode file. It saves time and frustration to learn what some of these messages are and what they mean. Unfortunately, at this stage in the game many of the messages will not be meaningful *except* to let you know where the first error occurred. Your only choice is to carefully study your program to find the error. In the following you will introduce a few typical errors into a simple program and examine the error messages.

1. Type the following program into a file called Hello.java. (This is the traditional first program a computer scientist writes in a new language.)

```
// ****************************************
//   Hello.java
//
//   Print a Hello, World message.
// ****************************************

public class Hello
{
    // ----------------------------------
    // main method -- prints the greeting
    // ----------------------------------
    public static void main (String[] args)
    {
        System.out.println ("Hello, World!");
    }
}
```

   Compile and run the program to see what it does. Then make the changes below, answering the questions as you go.

2. **Class name different from file name.** Delete one l (el) from the name of the class (so the first non-comment line is *public class Helo*), save the program, and recompile it. What was the error message?

3. **Misspelling inside string.** Correct the mistake above, then delete one l from the Hello in the message to be printed (inside the quotation marks). Save the program and recompile it. There is no error message—**why not?** Now run the program. What has changed?

4. **No ending quotation mark in a string literal.** Correct the spelling in the string, then delete the ending quotation mark enclosing the string Hello, World!. Save the program and recompile it. What error message(s) do you get?

5. **No beginning quotation mark in a string literal.** Put the ending quotation mark back, then take out the beginning one. Save and recompile. How many errors this time? Lots, even though there is really only one error. **When you get lots of errors always concentrate on finding the first one listed!!** Often fixing that one will fix the rest. After we study variables the error messages that came up this time will make more sense.

6. **No semicolon after a statement.** Fix the last error (put the quotation mark back). Now remove the semicolon at the end of the line that prints the message. Save the program and recompile it. What error message(s) do you get?

Correcting Syntax Errors

File *Problems.java* contains a simple Java program that contains a number of syntax errors. Save the program to your directory, study it and correct as many of the errors as you can find. Then compile the program; if there are still errors, correct them. Some things to remember:

- Java is case sensitive, so, for example, the identifiers *public*, *Public*, and *PUBLIC* are all considered different. For reserved words such as *public* and *void* and previously defined identifiers such as *String* and *System*, you have to get the case right. You will learn the conventions about case soon, but for now you can look at a sample program in the text to see what case should be used where.
- When the compiler lists lots of errors, fix the first one (or few) and then recompile—often the later errors aren't really errors in the program, they just indicate that the compiler is confused from earlier errors.
- Read the error messages carefully, and note what line numbers they refer to. Often the messages are helpful, but even when they aren't, the line numbers usually are.
- When the program compiles cleanly, run it.

```
// ****************************************
//   Problems.java
//
//   Provide lots of syntax errors for the user to correct.
//
 ****************************************

public class problems
{

    public Static main (string[] args)
    {

      System.out.println ("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
      System.out.println (This program used to have lots of problems,");
      System.0ut.println ("but if it prints this, you fixed them all.")
      System.out.println ("             *** Hurray! ***);
      System.out.println ("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");

    }
}
```