CS 110, Programming Fundamentals I
Lab 2: Syntax errors, user input, operators

Computer Science

In last lab you wrote your first Java Program, *HelloWorld.java*. This week, you'll continue to gain familiarity with jGRASP, and you'll write several simpler programs. This lab is worth 100 points.

Preliminaries

1. Using your CWU login information log into the computer and Canvas.

2. For this lab, create another folder, called **lab2**.

I. Declare Variables and Print Their Values to Output

In a computer program you declare and initialize variables so that you can then refer to the values (data) in later parts of the code. In this part of the lab, you will write a program in which you declare two variables, one of which of type String. Strings are used in Java to hold text. As you have seen in lecture, a string literal is enclosed by double quotes, and text being saved into a variable of type String is also enclosed by double quotes. You'll learn more about Strings in the next few lectures.

1. To get started, create a new java file (refer to lab 1 if you've forgotten how).

2. Type the text in the green box below into the editor panel of the java file that you just created:

```
// Author:
// Date:
// File: Declaring and printing to output Integer variables

// A simple class that declares several variables and prints them
public class DeclaringVariables {

   // The main method
   public static void main(String[] args) {

      // variable declarations and assignment
      int eBurgZipCode = 98924;
      String universityName = "Central Washington University";

      // Invoke the print and println methods from System.out
      System.out.print("The zip code of " + universityName);
      System.out.println(" is " + eBurgZipCode + ".");
   }
}
```

3. One important thing that you should notice about the code:

   The type *String* is spelled with a capital *S*. The word *string* (with a lower-case *s*), is syntactically incorrect, so be sure to always use *String* instead of *string,* when declaring variables of type *String*.

4. Save the java file as *DeclaringVariables.java*. To save, click on the "disk" icon, or select File -> Save.

5. Compile the java code and run it. The output of the program should be similar to Figure 1:

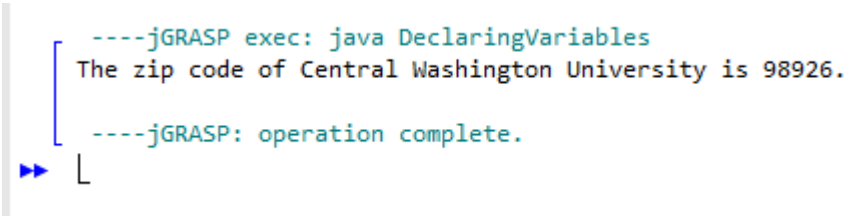

Figure 1: Output when the code in the green box is compiled and run.

II. Learning How to Locate and Resolve Syntax Errors

Because computer programs are complex, a first draft of your java code may contain syntax errors (you may have already dealt with such an error if in last week's lab you did not type the provided *HelloWorld.java* code correctly when creating your java file).

If you do not follow all of the java coding conventions when you compile your program, the java compiler will generate an error, to inform you that you've used incorrect java syntax. When the compiler encounters a syntax error, it stops converting your .java code to a .class file. When that happens, you must locate and fix the error, and attempt to compile again.

As you become more familiar with coding, you'll become adept at deciphering compilation (syntax) errors. In this part of the lab, you will be given java code that intentionally contains syntax errors. When you compile the code, we'll walk through the process of reading (and trying to understand) the compilation error to identify which part of the java code are faulty.

1. Download from the course schedule page the file *FindingSyntaxErrors.java* (shown in the light-blue colored box) and save it to your **lab2** folder.

2. Open the file in jGRASP – see how color formatting helps to recognize key words and how indentation make clear the overall structure of a java program.

```
// Author:
// Date:
// File: FindingSyntaxErrors

/* A simple class that contains syntax errors */
public class FindingSyntaxErrors; {

   // The main method
   public static void main(String[] args) {

      // Declare a variable of type integer
      int zipCode = 98924;

      // Invoke the print and println methods from System.out
```

```
      System.Out.print("Wellington P. Wildcat ");
      System.Out.println(
        " is the official mascot of our school.");
      System.Out.println("He lives at zip code " + zipCode)
   }
}
```

3. Compile your java program (the green "+" button). Because the java code that you've placed into your file contains intentional syntax errors, the java compiler will generate an error, which will be displayed in the *Compile messages panel*, as shown in Figure 2. The compiler error message includes a "^" that points to that part of the code that the compiler did not understand. In this case, the error message is informing you that the compiler was expecting a "{", but that instead it found a ";". To fix this error, remove the semicolon in the class declaration line before the "{". After you've made the edit, save your file.

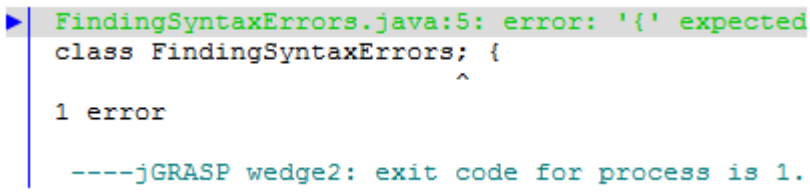

Figure 2: A sample compilation error message, where the ^ indicates the location of the error.

4. Compile your java program again. This time, the compiler will find the next error in the code. Notice that when you compiled your program the first time, only one error message was generated. That is because when the first error was found, the compiler stopped compiling the code. Only when the first error was fixed, could the compiler continue, and it found the second error. Use the error message in the *compile message panel* to locate and fix the second syntax error in your code. Hint: remember that Java is case-sensitive, so capitalization is important. After you've edited your code, save the file, and compile it. Keep on compiling and editing, until the program is error-free and compiles correctly. Then, run the program. The quotation should then be output correctly to the screen.

III. Your own program with Arithmetic Operators and reading input from the keyboard

For this part of the lab, you'll write your own java program, that retrieves input from the keyboard, and calculates the *circumference* of a circle (this is helpful for homework #2). You'll need to use the final constant PI, that is part of the Math class. The code for reading input from the keyboard is given to you and will be discussed in detail in the next two lectures.

1. Download from the course schedule page the file *CircumferenceOfCircle.java* (also shown in the white box on the next page) and save it to your **lab2** folder. The java code has no errors, but you need to complete the code.

2. Open the file *CircumferenceOfCircle*.java in jGRASP. Read the comments and follow the instructions to complete the program. There are three things that you must add to the file, indicated by "Step 1", "Step 2" and "Step 3". The code for creating a Scanner object (reading from keyboard) has been given to you.

3. Compile the program (fix syntax errors, if you find any), and run the program. The output should be similar to the following:

```
Enter a decimal number, and press return  1.2
The circle with radius 1.2 has a circumference of 7.5398223686155035
```

```
// Name
// Date
// Description
import java.util.Scanner;
public class CircumferenceOfCircle {

   public static void main(String[] args) {

      // create an object of type Scanner
      Scanner keyboard = new Scanner(System.in);

      // Step 1: declare a variable, call is radius, of type double




      // Ask the user to input a number with decimals
      System.out.print("Enter a decimal number, and press enter  ");

      // place the user's input into the variable radius
      //radius = keyboard.nextDouble();

      // Step 2: Declare a variable, call it diameter, of type double,
      // and assign it the value of twice the variable radius. For this
      // part, do NOT write "double diameter = 9.2". You must use the
      // variable "radius". For example, if you were assigning the value
      // to diameter to be twice the radius, you would write:
      //
      // double diameter = 2 * radius;




       // Step 3: Define a variable, circumference, of type double, and
      // use the Math library's PI value to assign it a value that is
      // the circumference, which has the formula pi * diameter.
      // Hint: to use the Math Library's pi value, you use: Math.PI.



      // Output the result to the screen
      System.out.println("The circle with radius " + radius +
            " has a circumference of " + circumference);
   }
}
```

Figure 3

IV. What to Hand In

Please upload your three .java files for lab 2 to Canvas:

1. Each .java file must be commented. Be sure to include your name at the top of each file. Also, if your code does not compile because you've been unable to fix a syntax error, then the comments that you provide in your code will allow you to receive partial credit. If you do not provide comments and your code does not compile, then it is very hard for a grader to determine if you were on the right track to the correct solution.

2. The code in each .java file must be indented, so that the code is easy to read.

3. Variables should be appropriately named. For example, *amountMoneyInBank* is a good variable name, but *variableName* is not, because it is not descriptive enough.

4. **Always remove my instructional comments (they are there to help you with an assignment) and replace them with your own.**

For this lab, make sure that you upload the following files to the Lab2 assignment in your Canvas account:

DeclaringVariables.java
FindingSyntaxErrors.java
CircumferenceOfCircle.java

There will be additional files (the **.class** files, that you've generated when compiling your code) in your lab2 folder, but **don't upload them** to Canvas. Those files are not graded; they are just the byte code files that are used by the Java Virtual Machine.

V. Rubric

File / Lab	Points
DeclaringVariables.java compiles and generates correct output when run	20
FindingSyntaxErrors.java compiles and generates correct output when run	20
CircumferenceOfCircle.java compiles and generates correct output when run	20
All .java files are commented(with your own comments), well formatted, and contain a header with your name and date	20
Variable names are adequate and descriptive	20
Total	100