COSC–111 Introduction to Computer Science I
Lab 02: Errors and Arithmetic
Due: Friday, February 15, 8.59am

1 Understanding Error Messages

You may work with a partner for this first part.

First, connect to the remote server via SSH (see the Instructions for Lab 01 if you need to refresh how to do it).

Then, make a new directory for this lab, change into that directory, and copy a number of Java files:

```
$ mkdir lab-2
$ cd lab-2
$ wget -q -i http://bit.ly/111s19L02A
```

Your directory should now contain 12 files whose names start with `Err` and end with `.java` (check using `ls -l`, i.e., *ell ess space dash ell*). The code in each of these files has at least one *syntax error*.

For each file, compile the source code and look at the error message. See if you can figure out what it means. Then open the source file using emacs and try to you correct the error. **After** you have corrected the error, look at the explanation of the problem (see below) and make sure you understand what is happening. *Understanding and acting upon* the messages given by the compiler `javac` is of key importance to become fluent in Java.

Err1.java

In this example, the compiler gives you the answer. It expects a ';' and the compiler tells you (using the ∨) exactly where it should be.

Err2.java

Again, the compiler tells you exactly what is wrong. It expects a ')', and it points to the place.

Err3.java

Let's look just at the first error message that is printed. This time the compiler is reasonably clear about the error, but it is not pointing to the location of the error. Rather, it is pointing to the place where the unclosed String starts. A "literal" is a constant value of a particular type, as opposed to a variable. Once you have corrected the first error, all of the other error messages should disappear: the additional errors are not real, and only appeared because the first error caused the compiler to be confused about how to read the rest of the program. Advice: when in doubt, fix the first error, recompile, and see what happens.

Err10.java

This may seem like an oddly phrased error message (class Err9a is public, should be declared in a file named Err9a.java). The real error is that the name of our file is Err9.java, and the name of the class contained within this file must also be Err9 (alternatively, the file name and the class name could be Err9a.java and Err9a respectively). The reason for the phrasing of the error message is that it is possible to have a non-public class, for which there are different rules. You do not need to know about this yet, but you should be aware that it may be difficult to understand some of the error messages at your current level of knowledge.

Err9.java

This error message (class, interface, or enum expected) is less helpful. The location is correct, but the message is off base. The compiler is confused by the extraneous word "Public", which is does not recognize as a misspelling of "public". Java is case sensitive!

Err8.java

Here the compiler seems to be hedging: it tells us that variable i "might" not have been initialized. What javac means is that it is not sure that i has been initialized (in this case it is clear from looking at the code that i has not been initialized; next week we will see some examples of programs in which it will be less clear).

Err7.java

Again, this is straightforward. The compiler tells you that a variable i already has been defined.

Err6.java

In this case the compiler is very clear both about the nature of the problem and the location. It cannot find the symbol j because no variable called j was ever declared.

Err5.java

This error message is perhaps not as clear. Why is it complaining about a "possible" problem? It turns out that it is only a possible problem because the double might contain an integer value, in which case we are fine. But the double might contain a value that is not an integer, in which case it cannot be assigned to an int variable. There are several possible ways to fix this, including declaring i to be a double or declaring j to be an int. It depends on what you really intended.

Err4.java

We see two errors, so let's handle the first one first. This is a little misleading because it points to the right place, but has the wrong expectation about what it wants to see. What is missing is a '{', not a ';'. As in the Err3 example, once we correct this error the second error message disappears.

Err11.java

Indeed, ')' is an illegal start of an expression, but the error message isn't particularly helpful. Instead, the problem is that javac expects an expression after the '+', but instead there is a ')'.

Err12.java

The first error message is technically correct; "`System.out.println("hello");`" is not a statement, but the compiler doesn't tell you that the problem is that there is a '{' instead of a '('. Once we fix this error, the compiler comes up with an entirely different error. This is common. You fix an error and the compiler can understand the program better, so it finds another error.

When you are done, **put both of your names in a comment at the top of Err1.java** and save the file. If you do not put your name in the file and you are working from your partner's account, I will have no way of knowing that you have completed this part of the assignment.

Submit all of your java files:

```
$ cssubmit *.java
```

and select "Lab 02A: Errors" as the assignment.

2 Geometry Computations

Work **on your own** for this part of the assignment.

Download a file `Lab02.java`:

```
$ wget -q -O Lab02B.java http://bit.ly/111s19L02B
```

Compile the `Lab02B.java` file and run the Lab02B program using the `javac` and `java` commands, respectively. When you run the program, it should prompt you to enter the base and height of a rectangle, and then it should print out the area.

You may notice that if you type in something that is not a number when the program asks you to enter the base or the height, the program will stop running and an error message will print. We will see later how to handle this; for now you can assume that your user is well behaved and will enter sensible numbers.

Your job is to calculate various geometric properties of different shapes. The table on the next page lists the shapes and the values you should compute, as well as the data you will need to read from the keyboard. For each shape, you should read from the keyboard the values listed in the "Input" column, compute all of the values listed in the "Value" column using the given formulas, and then print the results. For example, the output for the right triangle might look something like:

```
 Please enter the length of the base: 3 Please enter the height:
4 Right triangle area is: 6.0 Right triangle hypotenuse is: 5.0 Right triangle
perimeter is: 12.0
```

Your program should read in *new* input values for each shape (i.e., the user should be able to enter one base and height for the rectangle, then a different base and height for the triangle). So in the end, your program should read in 9 different values from the keyboard (one for each of the quantities in the "input" column) and print out 11 different results (one for each of the quantities in the "value" column).

Shape	Input	Value	Formula
Rectangle	Base (b) Height (h)	Area (A)	$A = bh$
		Perimeter (P)	$P = 2b + 2h$
Triangle	Base (b) Height (h)	Area (A)	$A = \frac{bh}{2}$
Right Triangle	Base (b) Height (h)	Area (A)	$A = \frac{bh}{2}$
		Hypotenuse (c)	$c = \sqrt{b^2 + h^2}$ *
		Perimeter (P)	$P = b + h + c$
Circle	Radius (r)	Area (A)	$A = \pi r^2$ **
		Circumference (C)	$C = 2\pi r$
Regular polygon	# sides (n) Side length (l)	Perimeter (P)	$P = nl$
		Apothem (a) ***	$a = \frac{l}{2\tan(\pi/n)}$ ****
		Area (A)	$A = \frac{aP}{2}$

***Note: you can use** `Math.sqrt(x)` **to compute the square root of a variable** `x`.
****Note: you can use** `Math.PI` **to get the value of** π.
*****Note: the apothem is the distance from the center of any side of a regular polygon to the polygon's midpoint.**
******Note: you can use** `Math.tan(x)` **to compute the tangent of a variable** `x`.

When you are done, submit your program:

```
$ cssubmit Lab02B.java
```

and select "Lab 02B: Geometry" as the assignment.