INTRODUCTION TO COMPUTER SCIENCE I
LAB 1
Errors and Arithmetic

1 Understanding Error Messages

For this part of the lab, you should **work with a partner**.

Use the following command to download a number of Java files:

```
$ wget -nv -i https://goo.gl/7xEAwp
```

Your directory should now contain 12 files whose names start with `Err` and end with `.java`. Each of these files has at least one syntax error in it. For each one, compile the program, look at the error message, and try to determine what it means. Then open the file using `emacs` and see if you can 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.

- `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 exact 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.

- `Err4.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.

- `Err5.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.

- `Err6.java`

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

- `Err7.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).

- `Err8.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!

- `Err9.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.

- `Err10.java`

We see two error messages, 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`

Javac is correct, '`)`' is indeed 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");`" indeed is not a statement, but the compiler does not give you the additional detail 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.

Completing the assignment: Put **both of your names** in a comment in `Err1.java`, and submit all of the corrected programs. (Instructions on submission are in Section 3.)

2 Geometry Computations

For this part of the lab, you must **work on your own**.

Download another file:

```
$ wget -nv -i https://goo.gl/NDvbeJ
```

Compile and run the `Lab2B` program using the `javac` and `java` commands. 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 (you should ask the user for each input once per shape).

When you are done, submit your program, per Section 3.

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)	$A = \frac{aP}{2}$

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

3 How to submit your work

For the first part of the assignment, submit your modified `Err1.java` through `Err12.java` files; for the second part of the assignment, submit your `Lab2.java` file. You may use either of the following two methods to use the CS submission system:

- **Web-based:** Visit the submission system web page. If you did your work on your own computer, open this link in your computer's browser; if you did your work on `remus/romulus`, then open this link in a browser opened within *remote desktop*.
- **Command-line based:** Use the `cssubmit` command at your shell prompt, as below, following the prompts it presents about which class and assignment you are submitting:[1]

```
$ cssubmit Err*.java
$ cssubmit Lab-2.java
```

This assignment is due on Thursday, September 21, 11:59 pm.

[1] This command works only while connected to `remus/romulus`.