CS 211, Lab 1 – Exercise (due: Mon 1/29, 11:59pm)
Installations and Introductions

Our goal is to get all tools installed, and then write a simple program. We will have a few very simple test cases, and you will learn how to run the test cases before turning in your work. This will prepare you for the exact process we use in all of our lab exercises and projects.

Files:

- **download** these files for testing into a folder named with your userid and lab number: gmason76_L1_230

  http://cs.gmu.edu/~marks/211/labs/Tester1e.java
  https://cs.gmu.edu/~marks/211/share/junit-cs211.jar

- Grading: entirely test-case based.

- mac/unix: we use a colon to separate items on the classpath.

```
javac -cp .:junit-cs211.jar *.java
java -cp .:junit-cs211.jar Tester1e
```

- Windows: we use a semi-colon to separate items on the classpath.

```
javac -cp .;junit-cs211.jar *.java
java -cp .;junit-cs211.jar Tester1e
```

As this is an **Exercise**, you can read any and all materials, ask us questions, talk with other students, and learn however you best learn in order to solve the task. Just create your own solution from those experiences, and turn in your work.

Installing Java

We need to get Java installed before we can do anything. Specifically, we need the **Java SDK**, not the JRE. We also need version 8 of Java; at the end of the semester we will finally use the newer features in Java 8, so go ahead and install it now.
(Java 9 is out recently, but is not yet supported by DrJava; stick with 8 for now.)

We already have a good short guide on installing Java in our free CS211 textbook:
https://cs.gmu.edu/~marks/211/textbook/

Follow the link, and get java and an editor installed, such as DrJava (officially supported in class).

Read as far as installing java and an editor, but realize that you have to complete different work (and test it!) to complete this lab.

Task

Begin by copying in this sample code into a file named **HelloLab.java**
(the file name and the class name have to match)

```
public class HelloLab {

  public static void main(String[] args){
    System.out.println("Hello, Lab!");
  }
}
```

For now that's just some boilerplate code you should start with for any new class, other than the name (**HelloLab**) and what your code does (like printing a message, here). Before too long you should know the meaning and purpose of each word here. Java is quite explicit, requiring every detail to be written into the code, and we don't know all of those details yet. That's okay.

Next, create the file **CSWisdom.java** using the same template, but replacing **HelloLab** with **CSWisdom**.

We can still use **"\n"** as the representation of a newline character. Modify your **CSWisdom** code so that it prints the following quote, including the blank line and attribution:

```
Computer science is
no more about computers
than astronomy is
about telescopes.

 - Edsger Dijkstra
```

You may have to play carefully with spacing to ensure you match the tester's output. Pay close attention to any error messages you get, and remember that **System.out.println(someString)** will always add a newline at the end of the requested content.

Running the Tester

You can run the tester independently of any IDE, directly on the command line. JUnit is a testing framework, not directly a part of java, so you will always need to have a copy of the JUnit code available when running any JUnit test suite. This is provided as a .jar file, which is secretly just a mislabeled .zip file of java code in disguise! Download junit-cs211.jar at the provided link above, and put it next to your code. Open up a terminal, and browse to the same location of your files. Then, run the following commands:

(Windows users, you must replace colons (:) with semi-colons (;) in the commands below)

- compile all .java files, with the "class path" including the current directory (.) and the junit jar file (junit-cs211.jar) as places to look for needed code.

```
javac -cp .:junit-cs211.jar *.java
```

- with the class path again containing the current directory and the junit jar file, find a compiled class named **Tester1e**, and call its **main** method.

```
java -cp .:junit-cs211.jar Tester1e
```

or, with DrJava, click the magic Test button when the **Tester1e.java** file is open.

Turning in your Assignment

Each time we turn in code for this class, we will create a folder to hold all your documents, a quick identifying document, and then we'll turn in a .zip file of this folder.

- the folder name must be your lab section number, your userID, and the assignment kind/number separated by underscores. For example, the correct pattern for this assignment would look like this for George Mason (gmason76) enrolled in lab section 230:

  230_gmason76_L1/
- For this assignment, put both your files in that folder:
  - HelloLab.java
  - CSWisdom.java
- Also create a textfile named **`ID.txt`**, containing the following information tailored to yourself. You will need this for every single project/lab you turn in.

```
Full Name: George Mason
userID: gmason76
G#: 00000001
Lecture section: 003
Lab Section: 230
```

- Next, create a .zip file from this folder.
  - Mac: just right-click and select "compress folder…", and it will generate it for you.
  - Windows: right-click the folder, "Send to…", "Compressed (zipped) folder", and it will generate it for you.
  - Linux: you've already decided you like to do things your own way, so I'm sure you've got this.
- Last, upload this zip file to Blackboard under the correct assignment. This time, it has three files in it: HelloLab.java, CSWisdom.java, and ID.txt.

Important Notes

- Whenever using the command line, just retype things instead of trying to copy from a pdf. Dashes and quotes and other fancy characters usually get mangled in the process, and unfortunately you often can't even tell by looking at it!
- We don't need the tester file; please do ***not*** include any files we provided you in your submissions.
- We can't inspect your work with the compiled (**`.class`**) versions of your code; those receive zero credit! **We always, always, always need the `.java` files.**
- Make sure you "Submit" the assignment on Blackboard. If you only "save", then you are "saving it for later but not giving it to anyone right now". And you'll completely miss the deadline and get no points and be sad.
- You can submit your work as many times as needed. You can also check your submission(s) and re-download them, to make sure you turned in the correct copy. **It is your responsibility as an adult taking a college class to make sure you actually turn in the correct work!** Just make double-checking your submission be part of your normal process.