Lab 01 Specification – Lab Tools Review; Compilers
Due (via your git repo) no later than 8 a.m., Tuesday, 11 September 2018.
50 points

Lab Goals

- Quick review of GitHub and git commands.
- Take a closer look at the output of the Java compiler.

Suggestions for Success

- Take a look at the suggestions for successfully completing the lab assignment, which is available at: `https://www.cs.allegheny.edu/sites/amohan/resources/suggestions.pdf`

Learning Assignment

If you have not done so already, please read all of the relevant "GitHub Guides", available at https://guides.github.com/, that explain how to use many of the features that GitHub provides. In particular, please make sure that you have read guides such as "Mastering Markdown" and "Documenting Your Projects on GitHub"; each of them will help you to understand how to use both GitHub and GitHub Classroom. To do well on this assignment, you should also read

- PLP chapter 01, section (1.1 - 1.3)

1. **[GitHub Setup.]** Take a look at the detailed documentation for getting started with GitHub, which is available at: `https://www.cs.allegheny.edu/sites/amohan/resources/github.pdf`
2. **[Java Experiment 1.]**
   (a) Create the following Java program in a fresh directory in your repository. I will also place a copy on the lab repository.

```
public class Lab1 {
        public static void main(String[] args) {
                int i = 5, j = 6, k = 127, l = 128, m = 255,
                n = 32767, o = 32768;
                System.out.println("i="+i);
                System.out.println("j="+j);
                System.out.println("k="+k);
                System.out.println("l="+l);
                System.out.println("m="+m);
                System.out.println("n="+n);
                System.out.println("o="+o);
        }
}
```

Add header comments to include your name and the Honor Code pledge, but do not change anything else in the file! Compile and execute it as you normally do (”javac Lab1.java”, ”java Lab1”) just to make sure it’s working.

(b) In the terminal window, type the command ”jbe &”. You should see a window something like Figure 1. Use the ”File/Open” command to open ”Lab1.class” (NOT Lab1.java!), then from the menu on the left expand ”Methods/main/Code” (see Figure 2).

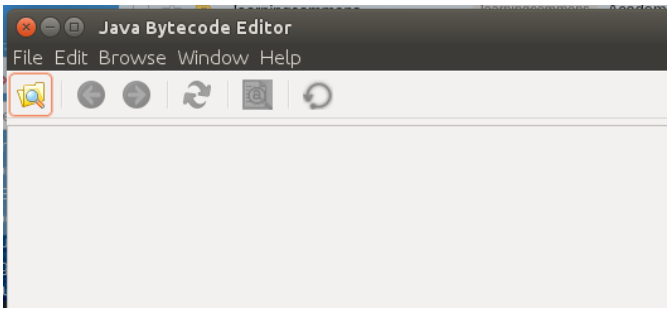

Figure 1: Result of typing jbe

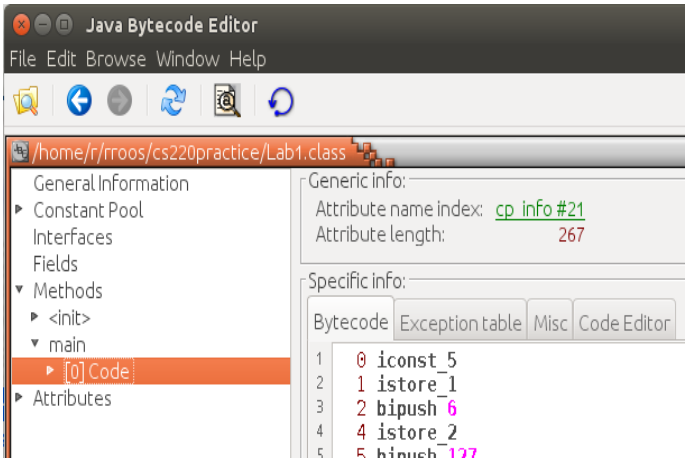

Figure 2: Getting to the bytecode

(c) Click on the ”Code Editor” button and locate the line that says ”iconst_5”. Change it to ”iconst_m1” and then click the ”Save Method” button. Dont worry if you get a warning in the terminal window about JDK 1.5just ignore it.

(d) In your terminal window, run the program again. DON’T recompile it! You should see a different result for i. Congratulations! You have just edited some Java bytecode!

(e) You will notice that different constant initial values compile into different JVM commands. For instance, all values from -1 through 5 have one-word bytecode commands (”iconst_m1” through ”iconst_5”), while values between 6 and 127 translate into ”bipush” commands. Another change occurs between 32767 and 32768. Use the jbe code editor so that the bytecode will set variable i to -2, variable j to 5000, variable k to 65000, variable l to 3. Don’t change the Java program, only change the bytecode! Run your program to see if it works.

(f) **[To Hand In.]** Commit the Lab1.java (with modified header comments) and the modified Lab1.class files to your repo. Be sure that the .class file reflects the changes you made to the Java bytecode in the previous step. Be sure you have not changed the code in the .java file (apart from the header comments).

3. **[Java Experiment 2.]**
   Now imitate what you just did in a brand new file, e.g,. "Lab1Part4.java" (do not modify the file from the previous part), but with double rather than int variables and constants. Through experimentation, try to determine the different ways that double constants are assigned to double variables. You needn't try to be exhaustive, but find at least two distinct JVM commands that assign double values to double variables. What happens if you assign an int constant to a double?

   Edit the bytecode so that your double variables are assigned different values from what is in the source code (as in the previous exercise) and verify that running the program without recompiling produces the new results.

   Write up your findings in a plain text document. Be sure to put your name in the document and to write in the Honor Code pledge. Commit your Java program and your modified .class file in your repository for submitting work. Make sure you do not recompile your Java program after you have edited the bytecode.

4. **[Speculate.]**
   At the bottom of your text document from the previous step, speculate as to the reasons why the Java compiler uses several different types of instructions for the assignments in the previous two questions.

5. **[Java Experiment 3.]**
   Experiment with simple integer arithmetic operators (addition, multiplication, etc.). What is the Java bytecode for adding two int variables? What is the Java bytecode for multiplying two int variables? What about subtraction and division? Place your answers at the end of your text file. In your text file, paste in the (short) portions of the source code and byte code that show the operators. You do not have to submit the Java program or the class file.

6. **[Java Experiment 4.]**
   Do some research on "constant folding". Does the Java compiler do any "constant folding". Write a program where constant folding could be performed, then generate the bytecode and see if the optimization occurred. Add the relevant pieces of the Java source code and the corresponding bytecode to the end of your text file, along with your answer.

Submission Details

1. Push your programs into the repository you shared with me.
2. Your two programs must be uploaded to your repository no later than 8 a.m., Tuesday, Sep. 11.
3. Questions about the lab? Bring them to class on Wednesday morning!

Grading Rubric

1. Task 1 in this assignment is a preliminary requirement to complete this lab assignment. There is no explicit points awarded for completion of these tasks. It is highly recommended to take this seriously and complete it as per the instructions, so that it can set you up nicely for the entire semester.
2. If you complete Task 2 completely as per the requirement outlined above, you will receive 20 points.
3. If you complete Task 3 completely as per the requirement outlined above, you will receive 10 points.
4. If you complete Task 4 completely as per the requirement outlined above, you will receive 5 points.
5. If you complete Task 5 completely as per the requirement outlined above, you will receive 10 points.
6. If you complete Task 6 completely as per the requirement outlined above, you will receive 5 points.
7. Failure to upload the lab assignment code and text file to your git repo, will lead you to get no points awarded for the lab.