CMPSC 111
Introduction to Computer Science I
Fall 2015

Lab 2
Assigned: January 27, 2016
Due: Wednesday, February 3, 2016 by 2:30 pm

Objectives

To develop a template for a Java program to use during this and future labs; learn standard ways of organizing and preparing the lab; write a program to perform a simple calculation using variables. Finally, to continue to practice using the Git version control system and the Bitbucket site.

General Guidelines for Labs

- **Work on the Alden Hall computers.** If you want to work on a different machine, be sure to transfer your programs to the Alden machines and re-run them before submitting.
- **Update your repository often!** You should add, commit, and push your updated files each time you work on them. I will not grade your programs until the due date has passed.
- **Review the Honor Code policy.** You may discuss programs with others, but programs that are nearly identical to others will be taken as evidence of violating the Honor Code.
- **Review the other guidelines.** Study the additional guidelines that were given on the first page of the last laboratory assignment sheet; ask the instructor if you have any questions.

Reading Assignment

Start learning more about the basics of implementing Java programs by reading Sections 1.4 and 1.5 and Sections 2.2 through 2.6 of your textbook. In addition, please review the slides that we have used during the recent class discussions. If you have questions about these reading assignments, please see the course instructor or a teaching assistant or departmental tutor during their office hours. Posting a question on Slack is another option for ensuring that your concern is resolved.

Create a Program Template

Every program you write will have header comments, a "`main`" method, some statements that print your name and the date, etc. Rather than typing this in every time you have to write a Java program, you are going to create one "template file" that can be copied into each of your future laboratory files. As we learn more Java, you will be making changes to this file to accommodate new features. You will store this template in your directory/repository, named `cs111S2016-<your user name>`, that you created during practical 1, making a copy of it for each laboratory assignment.

Go to the `cs111S2016-<your user name>` directory and type the command "`gvim Template.java`". Create a Java program template that you can fill in for each laboratory assignment. See the following program template for an example of what your template should look like. Please note that this program will not compile and cannot be run! You only need to create this template file once, thus allowing you to re-use it in the future laboratory sessions.

```
//************************************
// Honor Code: The work I am submitting is a result of my own thinking and efforts.
// Your Name  [Replace with your name]
// CMPSC 111 Spring 2016
// Lab #  [When you copy this file, fill in the lab number]
// Date: mmm dd yyyy  [When you copy this file, fill in the date]
//
// Purpose: ...  [When you copy this file, describe the program]
//************************************
import java.util.Date; // needed for printing today's date

public class Xxxxxx  [When you copy this file, replace with actual file name]
{
    //----------------------------
    // main method: program execution begins here
    //----------------------------
    public static void main(String[] args)
    {
       // Label output with name and date:
       System.out.println("Your Name\nLab #\n" + new Date() + "\n");

       // Variable dictionary:
        [Declare variables and use comments to explain their meanings]
    }
}
```

Save this file using the "File/Save" menu or by typing "`:w`" when you are not in insert mode. At this point, your `cs111S2016-<your user name>` directory should contain a file named "`Template.java`". You may now close the `gvim` editor. Make sure that you store this file in your Bitbucket repository! If you do not know how to transfer the files to Bitbucket, then please review the steps outlined in both the previous practical and laboratory assignments and the "Git Cheatsheet".

Create a New Directory

In your `cs111S2016-<your user name>` directory type the command "`mkdir lab2`" to create a new directory for the second laboratory. Type "`cd lab2`" to change into this new directory.

Write a Java Program to Perform a Simple Calculation

You are going to write a program that performs some kind of simple unit conversion. While I leave the specific problem up to you, I ask that it be in good taste and suitable for a classroom setting.

Creating the File: Type "`gvim Lab2.java`". Inside the `gvim` editor, make sure you are *not* in insert mode. That is, the word "`--INSERT--`" should *not* appear in the lower left corner. Then, type the following exactly as shown; it will appear in the bottom line of the `gvim` window:

```
:r../Template.java
```

This "reads" your `Template.java` file into your new `Lab2` program. You only need to do this once! If this step did not work correctly for you, then you can also try to copy and paste content from the `gvim` window containing the template to the one that has the source code for your `Lab2` program. Now you need to modify `Lab2.java` to include your name, the lab number, date, etc.

In your program for this laboratory assignment you will need to create variables and assign values to them. To create a variable you must declare it by specifying its type (e.g., `String`, `boolean`, or `int`) and the name of the variable that you chose; for instance, a full example would be "`int count;`"). To assign the value to a variable after you have declared it, you need to write an assignment statement using an assignment operator '=', as, for example, "`count = 0;`". You may combine the variable declaration and assignment into one statement as in "`int count = 0;`". You may print variables by incorporating them into a `print` or `println` statement by using '+' and the name of the variable as in `System.out.println("My first variable is " + count);`

Complete the laboratory assignment by writing the Java statements needed to perform several simple arithmetic calculations and print the results. See an example below. Obviously, you may not use this example—instead, you should creatively implement your own as part of this assignment.

```
/* Honor Code: The work I am submitting is a result of my own thinking and efforts.
   Gregory M. Kapfhammer
   Lab 2 for CMPSC 111 Spring 2016
   Date: January 27, 2016
   Purpose: to compute and print the number of yards between the Earth and the moon,
   then print lunar maximum and minimum temperatures in both Celsius and Fahrenheit. */

import java.util.Date; // needed for printing today's date

public class MoonDistance
{
    // main method: program execution begins here
    public static void main(String[] args)
    {
       // Label output with name and date:
       System.out.println("Your Name\nLab 2\n" + new Date() + "\n");
       // Variables:
       int milesToMoon = 238900;   // distance to moon in miles
       int ydsPerMile = 1760;      // number of yards in a mile
       int ydsToMoon;              // number of yards to the moon
       // Compute values:
       ydsToMoon = milesToMoon * ydsPerMile;

       System.out.println("Distance to the moon in miles: " + milesToMoon);
       System.out.println("The number of yards per mile: " +ydsPerMile);
       System.out.println("The number of yards from the earth");
       System.out.println("to the moon is " + ydsToMoon);
    }
}
```

Note the previous example shows only one calculation using the multiplication operator '*'. Your program needs to use at least three different arithmetic operators.

Program requirements

Your program must:

- Have a comments header section, containing the Honor Code, your name, lab number, and the purpose statement for the lab. These items will not be output since they are comments.
- Print your name, the lab number, and the current date and time, using "`new Date()`" as in the example source code from the first laboratory assignment.
- Declare and use at least three variables (if you can, try using variables of different types).
- Contain at least some variables that are initialized with constant values; others should be assigned the results of calculations, as in the previously discussed examples.
- Use at least three of the five arithmetic operators `+`, `-`, `*`, `/`, or `%`.
- Print the values of all variables, once their values are known, with appropriate labels.

In addition, for full credit you must:

- Make sure that the output printed by your program has an aesthetically pleasing appearance—for instance, you should have spaces between words and lines should not be longer than the width of the screen, forcing them to "wrap" to the next line. (For this laboratory assignment, you do not need to worry about the way fractional — that is, decimal — values are displayed.)
- Make sure your program is properly indented and makes careful use of whitespace.
- Make sure you have inserted comments describing the program's purpose and comments describing what each of the variables represents. Use both comment styles you have seen in class and are discussed in Chapter 1 of the textbook (i.e., `//` for a single-line comment and `/* ... */` for a comment that can span multiple lines).
- Use sensible variable names that are easy to remember and understand.

The Compile/Execute Cycle for the Java Programming Language

In your terminal window, still in the `lab2` directory, type:

```
javac Lab2.java
```

If there are errors, try to figure them out and correct them. Ask for help if you don't understand the error messages. Be sure to watch out for uppercase/lowercase errors, missing semicolons, etc.

When you have corrected all of the errors, then you can type:

```
java Lab2
```

Did you get the desired result? If not, repair the program and re-run the `javac` command to re-compile it. The cycle goes on like this: `javac` is used to re-compile the program every time you make a change to the file and then you use `java` to execute the program once `javac` finds no more errors. You may review a graphical summary of these steps by looking at the diagrams in Figures 1.20 and 1.21 of your textbook. Please see the instructor if you have questions about these steps.

Carefully Review the Honor Code

The Academic Honor Program that governs the entire academic program at Allegheny College is described in the Allegheny Academic Bulletin. The Honor Program applies to all work that is submitted for academic credit or to meet non-credit requirements for graduation at Allegheny College. This includes all work assigned for this class (e.g., examinations, laboratory assignments, and the final project). All students who have enrolled in the College will work under the Honor Program. Each student who has matriculated at the College has acknowledged the following pledge:

> I hereby recognize and pledge to fulfill my responsibilities, as defined in the Honor Code, and to maintain the integrity of both myself and the College community as a whole.

It is understood that an important part of the learning process in any course, and particularly one in computer science, derives from thoughtful discussions with teachers and fellow students. Such dialogue is encouraged. However, it is necessary to distinguish carefully between the student who discusses the principles underlying a problem with others and the student who produces assignments that are identical to, or merely variations on, someone else's work. While it is acceptable for students in this class to discuss their programs, data sets, and reports with their classmates, deliverables that are nearly identical to the work of others will be taken as evidence of violating the Honor Code.

Summary of the Required Deliverables

This assignment invites you to submit electronic versions of the following deliverables through the Bitbucket repository that you created during the first practical assignment. As you complete this step, you should make sure that you created a `lab2/` directory within the Git repository. Then, you can save all of the required deliverables in the `lab2/` directory — please see the course instructor or a teaching assistant if you are not able to create your directory properly. Additionally, students should submit printed and signed versions of all the required deliverables.

1. A completed, properly commented and formatted `Lab2.java` program.
2. The output from running `Lab2` in the terminal window. You may use `gvim` to save your output as follows: using the mouse, select everything from the "`java Lab2`" command to the end of your output. Right-click on the selected text and copy it. Type "`gvim output`"—note that this *not* a Java program!—and use the "Edit/Paste" menu item to paste your program's output into the file. Now, use "`:w`" or the "File/Save" menu item to save this file.

Share your program and the output file with me through your Git repository by correctly using "`git add`", "`git commit`", and "`git push`" commands. When you are done, please ensure that the Bitbucket Web site has a `lab2/` directory in your repository with the two files called `Lab2.java` and `output`. You should see the instructor if you have questions about assignment submission.