**CMPSC 111**
**Introduction to Computer Science I**
**Fall 2014**

**Lab 2**
**11 September 2014**
**Due Thursday, 18 September by 2:30 pm**

## Objective

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.

## 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.

## Reading

Please review the handout on "Tips on Using Linux and Gvim" (available in the shared repository under 'handouts' directory). Also review lecture slides and sections 2.2–2.6 in your textbook.

## 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 to accommodate new features. You will store this template in your directory/repository, named `cs111F2014-yourname`, that you created during practical 1, and make a copy of it for each laboratory assignment.

Go to `cs111F2014-yourname` directory and type the command "`gvim Template.java`". Create a Java program template that you can fill in for each laboratory assignment. See a program below 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, then you can 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 Fall 2014
// 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. At this point, your `cs111F2014-yourname` directory should contain a file named "`Template.java`." You may now close `gvim` editor.

### Create a New Directory

In your `cs111F2014-yourname` directory type the command "`mkdir lab2`" to create a new directory for lab 2. Type "`cd lab2`" to move to the 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. I leave the specific problem up to you; I just ask that it be "G-rated" and in good taste.

**Creating the File:** Type "`gvim Lab2.java`". Inside the `gvim` editor, make sure you are *not* in insert mode. (The word "`--INSERT--`" should *not* appear in the lower left corner.) 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! Now you need to edit `Lab2.java` program with 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 (String, integer, etc.) and the name of the variable that you chose (for example, `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: `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: `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!

```
/* Honor Code: The work I am submitting is a result of my own thinking and efforts.
   Janyl Jumadinova
   CMPSC 111 Fall 2014
   Lab 2
   Date: September 11, 2014

   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 this 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 printed since they are comments.
- Print your name, the lab number, and the current date and time (using "`new Date()`"—imitate the lab 1 program).
- Declare and use at least three variables (try using variables of different types).
- Some of your variables should be initialized with constant values; others should be assigned the results of calculations (see example below).
- Use at least three of the five arithmetic operators `+`, `-`, `*`, `/`, or `%`
- Once their values are known, print the values of all variables, labeled appropriately.

In addition, for full credit you must:

- Make sure the output printed by your program has a pleasing appearance—for instance, you should have spaces between words; lines should not be longer than the width of the screen, forcing them to "wrap" to the next line. (You do not need to worry about the way fractional values are displayed.)
- Make sure your program is properly indented.
- 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 /* ... */).
- Use variable names that "make sense"

## The Compile/Execute Cycle

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 the errors, type:

```
java Lab2
```

Did you get the desired result? If not, repair the program and go back to the `javac` command to re-compile it.

The cycle goes on like that: `javac` is used to re-compile the program every time you make a change to the file. Use `java` to execute the program once `javac` finds no more errors.

## Required Deliverables

This assignment asks you to submit electronic versions of the following deliverables through a bitbucket repository that you created during practical 1:

- A completed, properly commented and formatted `Lab2.java` program.
- 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 "paste" command to paste your program output into the file. Save this file.

Share your program and the output file through your repository with me by using 'add', 'commit' and 'push' commands correctly. When you are done, please check through bitbucket website that your 'lab2' directory in your repository contains two files: `Lab2.java` and `output`.