CSCI 136: Fall 2017

Lab 1

Due 17 September

Java, Unix, Silver Dollar Game

1 Prelab Preparation

This week, we will write the Silver Dollar Game at the end of Chapter 3 (page 67) of the textbook. BEFORE arriving to your scheduled lab section, please do the following:

1. Log in to the Macs. *You must use your CS-department username and password.* Please see Mary Bailey in TCL 312 if you do not have a CS-department Mac account or if you have forgotten your username/password.
2. Read through this lab handout and sketch out a design for your Silver Dollar Game program. You should use the sample Dice Design Doc as a guide. Each week your design doc will account for a small portion of your lab grade, so please bring it to lab and be prepared to discuss it.

2 Getting Started

The first goal of lab this week is to learn how to use the Java environment on the CS Lab Mac systems.

1. Go through the Unix tutorial on the course webpage (called Unix Cheat Sheet in the "links" section). This will teach you how to log in and out of the machines, use basic Unix commands, and edit files with emacs. Depending on your prior background, you may already be familiar with this environment. If that is the case, please refresh your memory and review the following commands.
2. Identify the function of and experiment with these Unix Commands:

   ```
   ls          cd          cp          mv          rm          mkdir       pwd
   man         history     cat         more        grep        head        tail
   ```

3. The Emacs text editor can be run within the terminal (just type `emacs`) or in a separate window (type `carbon-emacs`). If you choose to run it in a separate window, you may want the ability to use Command+C and Command+P for copying and pasting. To enable this functionality, open carbon-emacs, and go to Help→Mac-Style-Key-Bindings. Then choose Help→Save Options to make this your default environment. Identify the function of and experiment with these Emacs Commands:

   ```
   C-x C-s     C-x C-c     C-x C-f     C-x C-w     C-g         C-a         C-e
   C-d         C-_         C-v         M-v         C-s         C-r         M-%
   ```

   Learn these commands—you will use them often. For future reference, hints can always be found in the Unix and Emacs cheat sheet web pages on the course website.

4. To make sure we are all able to submit our Lab code when we are finished, please use `emacs` to make a file called `<your_unix>-test` (replace `<your_unix>` with your Williams Unix). The contents of the file can be anything you would like (the file will not be graded). Read the submission guide on the course webpage, and drag this file into your lab section's drop-off folder. *Complete this step before you leave the lab.*
5. Make a directory in your Unix account for CS 136 work (perhaps "`136`" or "`cs136`" might be reasonable). Make a subdirectory `<your_unix>-lab1` in this new directory for files related to this lab.
6. Write, compile, and run a Java program under Unix that prints the first ten odd numbers. Call it `Odd.java`. You will turn in your source code for `Odd.java` as part of your `<your_unix>-lab1` folder. Instructions for submitting are found in the **Submitting Your Program** section of this lab handout.

3 Lab Program

This week, we will write the Silver Dollar Game at the end of Chapter 3 (page 67). As you think about the design of your game, you should first decide on an internal representation of the coin strip. Make sure your representation supports all of the operations necessary, *i.e.* testing for a legal move, printing the board, testing for a win, moving pieces easily, etc. You should think about alternative designs and be able to justify your decisions. You may read ahead a little to Vectors if you like, but the lab can be implemented just as easily with arrays.

Once you have decided on a representation, write down the set of operations supported by your data structure. In other words, what are the public methods of `CoinStrip`, what parameters do they take, what do they return, and what do they do? We will briefly discuss this initial design with a partner.

The `main` method of your class should create a `CoinStrip` object and then prompt each of two players to make their moves. A move will be of the form:

```
cn ns
```

where `cn` is the coin to be moved and `ns` is the number of spaces it should be moved. The program should prompt the user for another input if the move is illegal. To read input from a terminal window, you should use the `Scanner` class, as we have seen in lecture. Consult the online documentation or sample code from class for details.

4 Submitting Your Program

Your main program should be named `CoinStrip.java`. **If you do not name your program `CoinStrip.java`, it may not be graded!**

When you are finished with the program, **answer the thought questions at the end of the lab**, and put your answers in a separate file called README.txt ('README' should be in upper case and the file should be a plain text file created using `emacs`—not a `Word`, `.pdf`, or `.rtf` document). Be sure to include your name in comments at the top of everything that you submit. Turn it in along with your other code in your `<unix>-lab1` folder:

```
<your_unix>-lab1/
            Odd.java
            CoinStrip.java
            README.txt
```

in your section's drop-off folder. Instructions for connecting to your drop-off folder can be found on the "Handouts" section of the course website.

As in all labs, you will be graded on design, documentation, style, and correctness. Be sure to document your program with appropriate comments, including a general description at the top of the file. Also use comments and descriptive variable names to clarify sections of the code which may not be clear to someone trying to understand it.

This program is due Sunday at 11pm.