CSCI 1933 Lab 1

Setting up Java and Programming Warm-up

Procedures for Lab

Labs will be run synchronously in person at the normally scheduled time. Lab attendance is a part of the requirements for this course. Unless there are special circumstances, such as illness, please plan to attend your scheduled lab each week. You are strongly encouraged to work with a partner within your lab section. The TAs will assist you in finding one, if needed. Have a TA evaluate your progress on each milestone before you move onto the next. The labs are designed to be mostly completed by the end of lab. If you are unable to complete all of the milestones by the end of lab, you will have up to **2:30 P.M. on the following Friday during office hours** to have any remaining lab steps graded by a TA. The office hours schedule, including locations, is located on Canvas. We suggest you get your milestones checked off as soon as you complete them since Friday office hours tend to become extremely crowded. You will only receive credit for the milestones you have checked off by a TA. Regrades of a milestone are available but only if you get them checked-off before the last office hours on the following Friday.

Introduction

Welcome to the first lab of CSCI 1933! In this first lab, you will be installing Java and running a simple program. After this quick introduction to Java, you will be solving warm-up questions in any programming language of your choice to get you ready for the coming semester. These warm-up questions should re-familiarize you with programming concepts from a background course, such as CSCI 1133.

Milestone 0:
Milestone 0 credit for each lab is earned by attendance at the lab meeting. When Milestone 1 is completed during lab, Milestone 0 credit will also be awarded. Credit will NOT be given during office hour checkoffs.

1 Hello Java!

1.1 The Java SDK

- If you are planning to use your own machine to work on assignments, a guide to installing Java on your home machine can be found on the Installing Java Canvas page. Note that we expect you to use Java 17 or higher for this course.
- Although Java is extremely portable, we recommend you to port your code over to the CSE lab machines before submission and make sure that it compiles and runs on the machines. For labs, you do not have to worry about this since labs are a "checkoff process". For projects,

however, you should verify that your code runs on the machines. If you find that your code runs on your machine but doesn't on the CSE machines and cannot figure out why, ask a TA for assistance. If you have not used the CSE machines remotely before this semester, we recommend using VOLE.

1.2 Simple Java Program

Java requires code to be in some kind of class in order for it to be run. If you don't remember what a class is, ask a TA. For the purposes of this course, every Java class we write will be in its own file. First, we need to set up your file structure. It's up to you how you want it to look, but we recommend the following file structure:

1. Create a `csci1933` directory (folder)
2. Create a labs directory in `csci1933`
3. For each lab, create a lab directory in labs, in today's case, `lab1`

Next, we need to make your first java class. To do so:

1. Open your favorite text editor — gedit, vim, emacs, notepad++, sublime, Atom, etc. — and make a new file.
2. Save your file as `Application.java`, save the file in your `lab1` directory.

Every Java application you write in this course will need a main method. You can treat a main method as the scripting portion of the java file, in other words where the program begins executing from. Type out the following into `Application.java`:

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

To run your first Java application:

1. Open your terminal/command line and navigate to the directory where your java file is.
2. Next, compile your program by running `javac Application.java`
3. Finally, to run your program, run `java Application`

Now modify the program to print a list of your classes this semester. Include a horizontal line between each of the listed classes. An example output for a student taking 3 computer science courses would look as follows:

```
Taylor's Classes:
CSCI 1933
---------
CSCI 2011
---------
CSCI 2033
```

Milestone 1:
To get checked off for this portion of the lab, show the modified `Application` class and the text being printed on the screen by running your class.

Warm-up Problems

Please complete the following problems in whatever language you feel comfortable. We recommend using Python if you completed CSCI 1133 or are familiar with Python. If these questions give you trouble, please reach out to a TA.

2 Most common character

Write a function that takes an input string and returns the most common character from the string, and the number of times it appeared. For example, the string "data" could return ("a", 2).

Note: For some words, there may be multiple "most common" letters. Your function does not need to return all of them. As long as your function returns one of them it will be considered correct. For example, in the word "noon," either ("n", 2) or ("o", 2) would be correct.

Milestone 2:
To get checked off for this portion of the lab, show correct output from the function using words your TA gives you. For example, if your program contains this code:

```
System.out.println(mostCommonChar("bejeweled"));
```

Then your output should be:

```
("e", 4)
```

3 Palindrome

A palindrome is a sequence that reads the same forwards and backwards. For example, the word car is not a palindrome because car read forward (c-a-r) is different from car read backwards (r-a-c). On the otherhand, the word racecar is a palindrome (r-a-c-e-c-a-r, forwards, is the same as r-a-c-e-c-a-r, backwards). Write a function that takes in a string as an argument and **uses recursion** to check if the string is a palindrome.

Milestone 3:
To get checked off for this portion of the lab, show correct output from the function using strings your TA gives you.
For example, if your program contains this code:

```
System.out.println(isPalindrome("folklore"));
System.out.println(isPalindrome("radar"));
```

Then your output should be:

```
false
true
```

4 Circle Class

Please use an object-oriented programming language (like Python) for this problem. For this next exercise, write a Circle class. As you may recall, objects usually have attributes that store useful information. A BankAccount has a balance attribute, a Car has the number of miles driven, a TV has a screen size, etc. For our Circle class, we will only have a radius. In addition to the radius attribute, the Circle class should have the following methods/constructors.

- A constructor that takes in the radius.
- A method getRadius, which returns the radius of the circle.
- A method setRadius, which sets the radius of the circle.
- A method getArea, which returns the area of the circle.
- A method getDiameter which returns the diameter of the circle.
- A method getCircumference which returns the circumference of a circle.
- If it is possible in your language, override the equality operator (==). In Python it's the __eq__ method. You should return True if the radii are equal and False otherwise.

Milestone 4:

To get checked off for this portion of the lab, create two circles with some radius. Display their areas, diameters and circumferences and also test the equality of the two circles.
For example, if your program contains this code:

```
Circle c = new Circle(22);
Circle b = new Circle(22);
System.out.println(c.getRadius());
System.out.println(c.equals(b));
c.setRadius(13);
System.out.println(c.getRadius());
System.out.println(c.equals(b));
System.out.println(c.getArea());
System.out.println(c.getDiameter());
System.out.println(c.getCircumference());
```

Then your output should be:

```
22
true
13
false
530.92871
26.0
81.68133999999999
```