CSCI 1933 Lab 1

Setting up Java and Programming Warm-up

Lab Rules

You may work individually or with *one* partner in lab. We suggest that you have a TA evaluate your progress on each milestone before you move onto the next. The labs are designed to be completable by the end of lab. If you are unable to complete all of the milestones by the end of lab, you have **until the last office hours on Monday** to get those checked off by **any** of the course TAs. We suggest you get your milestones checked off as soon as you complete them since Monday office hours tend to become crowded. If you worked with a partner, both of you must be present when getting checked off to receive credit. You will only receive credit for the milestones you have checked off by a TA. There is nothing to submit to Canvas for this lab.

Attendance

Per the syllabus, students with 4 unexcused lab absences over the course of the semester will automatically fail the course. Meaning if you have missed 3 labs without excuses you are OK, but if you miss a fourth lab (without a formal excuse) you would fail the course. The TAs will take attendance during lab; it is expected that students will be actively working on the lab material. *Your physical presence in lab is not sufficient to be marked as present for the purposes of attendance.*

Introduction

Welcome to the first lab of CSCI 1933! You will be using the CSE Labs machines to work on your assignments, unless you bring your own computer, which you are free to do. A CSE Labs account is required if you plan on using the CSE Labs machine, so if you do not have one, you can submit an application online or go to the operator office, pick up a form, and have one of your TAs authorize your account.

1 Hello Java!

1.1 The Java SDK

- If you are planning to use your own machine to work on assignments, make sure you have Java SDK 1.8 installed on your machine. You can download the SDK 1.8 from here. If you've installed the SDK and you still cannot run files, try updating your path — described here
- If you plan on using both your machine and the CSE machines (i.e. transferring files via USB or Google Drive), you must use SDK 1.8 since the CSE machines have SDK 1.8 installed.
- Although Java is extremely portable, we recommend you to port your code over to the CSE 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.

1.2 Simple Java Program

Java requires code to be in some kind of class in order for 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, lab0

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

1. Open your favorite text editor (gedit, vim, emacs, notepad++, sublime, etc.) and make a new file
2. Save your file as Application.java, save the file in your lab0 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. Run `javac Application.java`
3. Now run `java Application`

Milestone 1:
To get checked off for this portion of the lab, show `Hello world!` being printed on the screen by running your class.

Warmup Problems

Complete the following problems in whatever language you feel comfortable. We recommend using Python if you completed CSCI 1133. Also if these questions give you trouble please reach out to a TA.

2 Most common character

Write a function that takes a string as a parameter and returns the most common character from the string along with the number of times it appeared. For example, the string "data" should return ("a", 2).

Note: The mode is not always unique. E.g. in the word "noon", the characters 'n' and 'o' each show up twice. Your function does not need to return all the modes for a string (i.e. you can choose to return one arbitrarily).

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

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 other hand, 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 accepts a string as an argument and recursively checks 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.

4 Circle Class

Use an object-oriented programming language (like Python) for this problem. 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 this next exercise, write a `Circle` class which holds a radius attribute. In addition, 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 and only if the radii are equal.

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.

5 Honors Section — Sum of its parts

Note: This section and milestone are only required for students in the honors section. *Students in other sections do not need to do this, but are still encouraged to work on it if interested and time permits.*

Write a function that accepts two lists — `l1` and `l2` — and an integer, `n` as parameters. Your function's goal is to find one element from each list such that their sum is `n`.

For example: given `l1` = $\begin{array}{|c|c|c|c|} \multicolumn{1}{c}{0} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} \\ \hline 7 & 4 & 5 & 2 \\ \hline \end{array}$, `l2` = $\begin{array}{|c|c|c|c|} \multicolumn{1}{c}{0} & \multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} \\ \hline 6 & 1 & 7 & 0 \\ \hline \end{array}$, and `n` = 9 — the result would be $(2, 7)$

Milestone 5:
To get checked off for this portion of the lab, create a series of test cases for your function and demonstrate their correctness to a TA.