Lab 1

The purpose of this lab is to ensure that you can write, compile, and run simple Java programs.

You have two exercises to complete:

1) exercise on page 2 of these instructions (which is the same as Exercise 1.1.1 from Lecture 1 notes posted on Canvas), and

2) exercise on page 4 of these instructions (which is similar to the example discussed in class on August 26th, so your class notes will be helpful here).

Ideally, you will complete the first exercise by the end of your first lab on Tuesday, August 25. However, because this is your first lab, the deadline for this lab is Tuesday, September 1, 11:59 pm.

Both exercises must be demoed either to your TA or me. You are also asked to upload your solutions to Canvas.

Exercise 1 (5 points)

Consider the following `Duck.java` file (which you can also ***download from here***):

```
1  public class Duck {
2     // instance variables
3     private int age = 2;
4     private String name;
5
6     // one-argument constructor
7     public Duck(String name) {
8        this.name = name;
9     }
10
11    // other methods
12    public void makeSound() {
13       System.out.println("Quack!");
14    }
15
16    public void sayHello() {
17       System.out.println("I'm " + name);
18    }
19
20    public int getAge() {
21       return age;
22    }
23
24    // main() tests the Duck data type
25    public static void main(String[] args) {
26       Duck duck = new Duck("Daffy");
27
28       duck.sayHello();  // prints "I'm Daffy"
29       duck.makeSound(); // prints "Quack!"
30    }
31 }
```

1. Add a `public` method `getName()` to the `Duck` class. The method should receive no parameters and return a `String` (it should return the value of the `name` field).

2. Line 26 in `Duck` creates a `Duck` object and sets its `name` field to `Daffy` and assigns it to the reference variable `daffy`. Add a line inside the `main()` method that prints `daffy`'s name using the getter method you wrote in part 1.

3. Line 8 in the constructor reads `this.name = name;` Change this line to `name = name;` (that is, delete `this.`) Recompile and run your new `Duck`. What does it print? Why? Explain the result in one or two sentences.

4. Now change line 17 to `System.out.println(this.name)` (that is, add `this.`). What difference did this change make? Do you see any changes in the output? Explain in a sentence or two the effect of adding `this.` before `name` on line 17.

5. Do you think that `Duck`'s author made a wise choice by using the same variable name, `name`, for both the `name` field (line 4) and for the parameter name (line 8)? Which of the following three versions of the constructor do you prefer?

```
// field:
private String name;

// constructor:
public Duck(String name)
{
        this.name = name;
}
```

```
// field:
private String name;

// constructor:
public Duck(String newName)
{
        this.name = newName;
}
```

```
// field:
private String name;

// constructor:
public Duck(String newName)
{
        name = newName;
}
```

How to get full credit for Exercise 1?

1) Show your solution to your TA during the lab.

2) Create a text file called `Exercise1solution.txt` containing your answers to parts 3, 4, and 5 of this exercise, and upload your answers to Canvas before 11:59 pm on Tuesday, September 1. (You do not need to upload your answers to parts 1 and 2.)

Exercise 2 (15 points)

Write a program that

1) reads a `String` from the keyboard,

2) constructs a URL for `http://www.X.com/`, replacing `X` with the String read in,

3) prints the first five lines of the Web page at that URL in *reverse order* (i.e., the fifth, fourth, third, second, and first lines).

Starter Code

We've already created a starter file for you in the file `OpenWebPage.java`. You just need to fill it in. Use the `println()` method to print each of the five lines, so that there's a carriage return at the end of each line.

What to do?

Download ***the starter file*** and insert your solution below the line that says "`YOUR CODE HERE.`"

How to get full credit for this assignment?

1) Your solution must be in a file called `OpenWebPage.java` and uploaded to Canvas before the due date (11:59 pm, Tuesday, September 1.).

2) Do not edit any of the lines before the line that says "`YOUR CODE HERE.`"

3) Your program must print only five lines from the given home page, and must print them in reverse order. Do not add any extraneous `println()` statements.

4) Demo your solution either to your TA or me before the due date.

Optional: Hints & Suggestions Regarding Exercise 2

- One hint is offered by the first line of the starter code, which says "`import java.net.*;`". This suggests to look for a class that might be useful in the `java.net` package.
- The exercise asks to construct a URL from a String. As luck would have it, there is a class called `URL` in the `java.net` package. An bject of this class can be instantiated by passing a `String` as an argument to the `URL` constructor.
- As you explore the `URL` documentation, you'll notice that `URL` has a method that returns an `InputStream` object. Use that method.
- Starting with an `InputStream` object, construct a 'pipeline' (as we did in class) that can read a line of text from the URL. You will likely call the `readLine()` method on one of the objects in your pipeline.
- Once you are able to read the lines from a webpage, create an array of five `String`s and store the first five the lines of the web page.
- Finally, print that array backwards (starting from the fifth line and printing the first line last).

A sample run

```
% javac OpenWebPage.java
% java OpenWebPage
Please enter the name of a company (without spaces): oracle
<head>
        <!-- start : ocom/common/global/components/framework/head -->
<!-- start : Framework/HomePage -->
<html lang="en-US" class="no-js">
<!DOCTYPE html>
%
```