Crafton Hills College

Lab Manual

Data Structures and Algorithms (Java)

Frank Madrid

January 17, 2020

Introduction

This document serves as lab manual for the course *CSCI 230 - Data Structures and Algorithms* at Crafton Hills College and contains supplementary reading/laboratory assignments which are assigned throughout the course. It is expected that students will read the assigned lab manual before attending each lab session.

Please send any corrections to fmadrid@craftonhills.edu or leave a voicemail on my office phone (909) 394-3555.

Contents

The Eclipse IDE

1.1 Introduction

The Eclipse IDE is a graphical user interface famous for its Java Integrated Development Environment (IDE) but has extended to include other programming languages, including C/C++, JavaScript, PHP, and more. In this course, we will be using the Eclipse IDE to read and write Java programs.

1.2 Installing the Eclipse IDE

To use Eclipse for Java programming, we need to first install a Java Development Kit (JDK) version 8 or higher, see Java SE Development Kit 13 Downloads (**Figure 1.1**). Afterwards, installing the Eclipse IDE is as simple as installing any other application on your operating system.

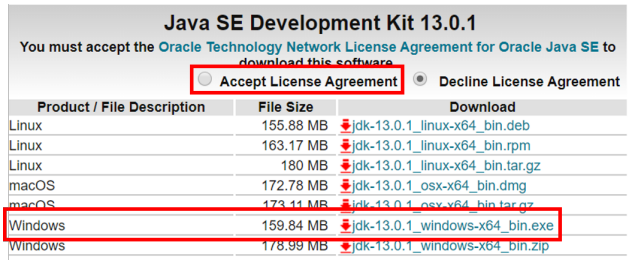

Figure 1.1: *For Windows installations, I recommend the* `*.exe` *file. Also, do not forget to select the "Accept License Agreement" button.*

1.3 Using the Eclipse IDE

The Eclipse IDE (like many other IDEs) includes many features to facilitate the programming process. This lab only contains basic instructions for the reading, writing, and executing Java source code. Students interested in exploring other features of the Eclipse IDE should watch this YouTube playlist.

1.3.1 Creating a Java Project

Similar to the Visual Studio IDE, the Eclipse IDE does not operate on single files. This means you will not simply be able to double-click on a `*.java` file and execute the source code within. You will first need to create a new java project for your `*.java` source code to reside in. The Eclipse IDE expedites this process using the **Create a Java Project** wizard.

1. Inside Eclipse select the menu item **File** → **New** → **Java Project** to open the **Create a Java Project** wizard (see **Figure 1.2**). In this wizard,
   a. Provide an appropriate **Project name**. I suggest *"CSCI 230 Lab 1"*.
   b. Ensure you are using an execution environment JRE of 1.8 or higher (e.g., JavaSE-1.8)
   c. and then click **Finish**.

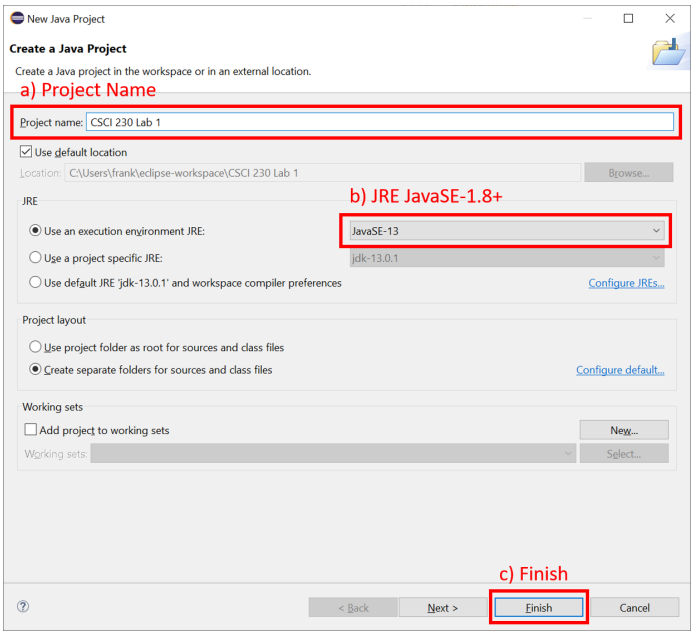

Figure 1.2: *Caption*

1.3.2 Class HelloWorld

In our new java project, we can now write a simple project which will output the text `Hello World!` to the console. Once again, the Eclipse IDE facilitates this process using the **Java Class** wizard.

1. Inside Eclipse select the menu item **File** → **New** → **Class** to open the **Java Class** wizard (see **Figure 1.3**). In this wizard,
   a. Provide a valid programmer-identifier for **Class name** (i.e., cannot begin with a number nor contain special characters) using a camel case format. I suggest *"HelloWorld"*.
   b. Select the checkbox for `public static void main(String[] args)` to have Eclipse automatically include this method skeleton in our class file.
   c. and then click **Finish**.
2. Copy the code found in **Code 1.1**.
3. Execute the program using menu item **Run** → **Run** or keyboard shortcut `Ctrl + F11`. Alternatively, if you have multiple classes with a function `main`, you can execute a specific file by right-clicking the file and selecting **Run As** → **Java Application**. This will launch the selected class as a local Java application. The Run As context menu item is also available in other places, such as the Outline view.
4. Program output can be viewed in the console window (**Window** → **Show view** → **Console**).

Code 1.1: `HelloWorld.java`

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

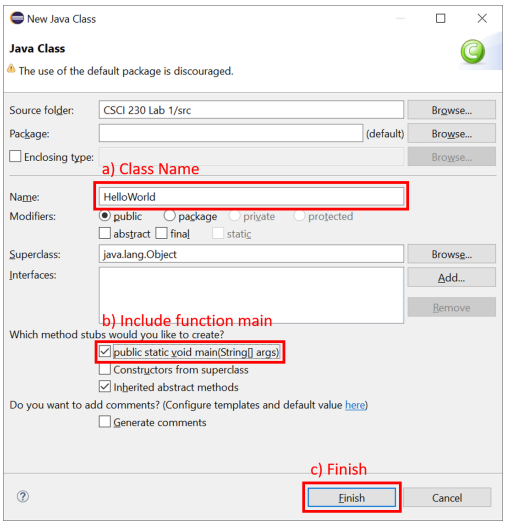

Figure 1.3: *Caption*

1.4 Lab - Basic Java

Instructions

In groups of two-to-three, complete *at least one* of the following problems (more than if you have extra time in the lab). When you are ready to get checked off for this lab, notify the instructor. The instructor will ask you questions about the basics of the Eclipse IDE and will ask you to explain and demonstrate your complete Java program.

Problem 1.4.1 (Subtract the Product and Sum of Digits of an Integer)**.** Given an integer number n, return the difference between the product of its digits and the sum of its digits. Write both an iterative and a recursive solution.

Problem 1.4.2 (Split a String in Balanced Strings)**.** A *balanced string* is a string which has equal quantities of the characters `L` and `R`. Let s be a balanced string, find all substrings s' of s which are also balanced.

Problem 1.4.3 (Array Partition)**.** Given an array of $2n$ integers, your task is to group these integers into n pairs of integers, say $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$ which makes the sum of $\min(a_i, , b_i)$ for all i from 1 to n as large as possible.

Example

```
Input: [1,4,3,2]
Output: 4
Explanation: n is 2, and the maximum sum of pairs is 4 = min(1,2) + min(3,4)
```

1.5 Lab - Object Oriented Programming

Instructions

In groups of two-to-three, complete *at least one* of the following problems (more than if you have extra time in the lab). When you are ready to get checked off for this lab, notify the instructor. The instructor will ask you questions about the basics of object-oriented programming and will ask you to explain and demonstrate your complete Java program.

Problem 1.5.1 (Shape Inheritance)**.** Define a `Polygon` interface that has methods `area()` and `perimeter()`. Then implement classes for `Triangle`, `Quadrilateral`, `Pentagon`, `Hexagon`, and `Octagon`, which implement this interface, with the obvious meanings for the `area()` and `perimeter()` methods. Also implement classes,

`IsoscelesTriangle`, `EquilateralTriangle`, `Rectangle`, and `Square`, which have the appropriate inheritance relationships. Finally, write a simple user interface, which allows users to create polygons of the various types, input their geometric dimensions, and then output their area and perimeter.

Problem 1.5.2. Design and implement class `Polynomial<T,n>` which can be used to represent an n-degree polynomial $p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$. This class should facilitate the addition, subtraction, multiplication, and derivative operations and contain the appropriate constructors, getters, and setters.