EECS 2030 Winter 2020: Lab 1

Lassonde School of Engineering, York University

INTRODUCTION TO LAB-1

Students in the same lab section are allowed to work in groups of 2 or 3 students.

The purpose of this lab is to review the following basic Java concepts that should have been covered in your previous courses:

- style
- using int and double values and variables
- arithmetic and the methods provided in java.lang.Math
- boolean expressions
- using objects
- if statements
- using Strings
- using Lists
- for loops

This lab also introduces code testing using JUnit. This lab will be graded for style and correctness.

STYLE RULES FOR CODING

The style rules are not overly restrictive in EECS2030.

1) Your programs should use the normal Java conventions (class names begin with an uppercase letter, variable names begin with a lowercase letter, public static final constants should be in all caps, etc.).
2) In general, use short but descriptive variable names. There are exceptions to this rule; for example, traditional loop variables are often called i, j, k, etc. Avoid very long names; they are hard to read, take up too much screen space, and are easy to mistype.
3) Use a consistent indentation size. Beware of the TAB vs SPACE problem: Tabs have no fixed size; one editor might interpret a tab to be 4 spaces and another might use 8 spaces. If you mix tabs and spaces, you will have indenting errors when your code is viewed in different editors.
4) Use a consistent and aligned brace style.
5) Insert a space around operators (except the period ".")
6) Avoid using "magic numbers". A magic number is a number that appears in a program in place of a named constant.
7) A good IDE (integrated development environment) such as eclipse will correct many style errors for you. In eclipse, you can select the code that you want to format, right

click to bring up a context menu, and choose Source → Format to automatically format your code.

GETTING STARTED

Download the provided project package **EECS2030_LAB1W20_student.zip** and import it into Eclipse. If you do this lab in prism lab, you can import directly from directory **/eecs/dept/www/course/2030/labs/lab1**.

Once the project has been imported, you can see a folder **doc** under the project, which contains the provided API. You can view the API by clicking file **index.html** and viewing it inside Eclipse, or, go to the project directory and open in your Browser.

LAB EXCERCISES

1) **Exercise 1:** Complete the methods in the **Vector2.java**. Run the JUnit test cases in **Vector2Test.java** after you complete each method to check your work. Read the API for more details.
2) **Exercise 2:** Complete the methods in the **Point2.java**. Run the JUnit test cases in **PointTester.java** after you complete each method to check your work. Read the API for more details.
3) **Exercise 3:** In package **lab1package2**, create a public class **Rectangle.java** with integer variables height and width. Write down the default and the custom constructor. Then write down the getters methods for the height, width, and area of the rectangle. Then, write down the setters for the height and width. Finally, write a **toString()** method to print the height, width, and area of the rectangle. Consult the test cases provided in API and **RectTester.java** to figure out the correct names of the getter and setter methods.
4) **Exercise 4:** In package **lab1package**, create a public class **Circle.java** with variables radius and color. Write down the default constructor for radius 1.0 and color ”blue” of the circle. Write down the custom constructor that initializes the circle to the given radius and given color. An exception **IllegalArgumentException** is thrown if the radius is less than 0.0. Consult the test cases provided in **Circletest** to figure out the correct names of the getter and setter methods.
5) **Exercise 5:** In package **lab1package**, create a public class **student** which models a student having courses and grades. The student class has two variables **name** and **address**. The courses taken and grades for the courses are kept in 2 parallel arrays. The maximum number of courses taken by student is 30.
   - Create a constructor **public Student(String name, String address)**. You can also initialize number of courses and the grades array.

- Create the public getters and setters **public String getName(), public String getAddress(), public void setAddress(String address)**. No setter for name as it is not designed to be changed.
- Print the name and address of student
- Write a method to add a course and grade for the student **public void addCourseGrade(String course, int grade)**
- Print all courses taken and their grades by writing a method **public void printGrades()**
- Write a method to compute the average grade for the student **public double getAverageGrade()**
- Check out the class **tester** to verify your methods.

The console output will look like this.
Jim(1 Happy Ave)
Jim(8 maple, vaughan)
Jim
8 maple, vaughan
Jim IM101:89 IM102:57 IM103:96
The average grade is 80.67

6) **Exercise 6:** In package **lab1package3**, create classes **Attraction.java** and **TourismBook.java**.
   - For the **Attraction** class, we have two variables **name** and **price**. Create a defalult constructor and custom constructor for the Attraction class. Write down the getters and setters for the name and price variables. Finally, write a method to print the attraction name and the price.
   - For the **TourismBook** class, we have an array of attractions and a variable to store the number of attractions. Write down the defalult constructor. Write down a getter method **getAttractions()** for the attractions. Write down two different methods **addAttraction(Attraction c)** and **addAttraction(String n, double p)** to add attractions. Write down a method **nameBestAttraction()** which selects the attraction having minimum price.

   Read the provided API and the two testers (**Tester2.java** and **tester.java**) for hints on how the constructors and methods work, and use the two different testers classes to verify your classes.

SUBMIT INSTRUCTIONS FOR STUDENTS NOT WORKING IN A GROUP

Submit your solutions using the submit command. Refer to lab0 (manual) webpage for instructions. **Submit one file at a time**.

Remember that you first need to find your workspace directory, then you need to find your project directory. In your project directory, your files will be located under dif-

ferent package directories. For example, your **Vector2.java** should be in the directory **EECS2030_LAB1W20_student/src/lab1package2**.

Once you are in the directory, issue **submit 2030 lab1 Vector2.java**

Repeat the above steps for each of the other program files.

At any time and in any directory, you can view the files that you have submitted by issuing **submit -l 2030 lab1**

Once your file reaches the submit directory, it will be pre-processed. First we check if the file name is correct. If it is not, then you will see warning message **"ERROR We expected the files: .... java but you submitted the ...."**. If the file name is correct, it will be compiled, and if passed the compilation it will be tested (using the test cases similar to those provided to you). If the file does not compile or does not pass all the test cases, you will see relevant error messages. In both cases, the file is accepted but you are expected to fix the problem and submit again. (You can always submit a file again, and the later versions will overwrite the old ones.) If your file passed all the test cases, you will see message **"Your submission passed the unit tests."**

SUBMIT INSTRUCTIONS FOR STUDENTS WORKING IN A GROUP

If you are working in a group, in addition to submitting program files as shown above, you also need to submit a text file containing EECS login names of your group members.

Create a plain text file named **group.txt**. You can do this in eclipse using the menu File $\rightarrow$ New $\rightarrow$ File, or, use any other text editors. Type your login names into the file with each login name on its own line. For example, if the students with EECS login names rey, finn, and dameronp, worked in a group the contents of **group.txt** would be:

rey
finn
dameronp

Submit your group information using submit command. In the directory where the text file is saved, issue **submit 2030 lab1 group.txt**

SUBMIT INSTRUCTIONS FOR STUDENTS OUTSIDE PRISM LAB

It is possible to submit work from outside the Prism lab, but the process is not trivial; do not attempt to do so at the last minute if the process is new to you. The process for submitting from outside of the Prism lab involves the following steps:

1) transfer the files from your computer to the undergraduate EECS server red.eecs.yorku.ca
2) remotely log in to your EECS account
3) submit your newly transferred files in your remote login session using the instructions for submitting from within the lab
4) repeat Steps 1 and 3 as required

Windows users will likely need to install additional software first. Mac users have all of the required software as part of MacOS.