**COSC 419 – Summer II 2019**

**Assignment3: Code\_Refactoring plus Adapter and Façade design patterns (Maxi 50 points)**

Number of People: Individual. You are welcome to discuss the problem among your classmates but not sharing the solution.

Due: (Tues) July 23, 2019 by 11:59pm or 7 days from the posted date.

Submission: Zip all your Java source files (\*.java) plus your written/non-programing description (\*.docx) into a single file and upload it to the proper course folder in Canvas. Please make sure you have successfully tested your program without crashing. Be sure to submit the .java (source codes) not the .class files.

Coding Style: Use consistent indentation (use the Format tool in Eclipse (Ctrl+Shift+ F) . Use standard Java naming conventions for **variableAndMethodNames**, **ClassNames**, **CONSTANT\_NAMES**. Must have comments for all your codes. Comments should focus on what and why and NOT how.

**TA: Mariam Mayeesha ( mayeesha.mariam@alumni.ubc.ca)**

Please discuss with your TA/GA first regarding your marking discrepancy.

The purpose of this assignment is to perform adapter and Façade design pattern type exercises

**Question 1**: (25 points) This question deals with Adapter design pattern that provides an interface between two incompatible objects. The following program NoAdapterProvide( ) uses the two incompatible interfaces to draw their corresponding lines based on the co-ordinates provides. Since the interface between Line and Rectangle objects is incompatible, the client has to get (if-else) the type of each shape and manually punch in the proper arguments. (The source codes (NoAdapterProvided.java) is included in the assignment3 zip file). You are to write an Adapter for the two interfaces (Line, Rectangle). The Rectangle width and height dimensions can be calculated based on the (x1, y1), (x2, y2) pair provides by the client.

**/\***

**\* Because the interface between Line and Rectangle objects is incompatible, the user has to recover the type of each shape and**

**\* manually supply the correct arguments.**

**\***

**\*/**

**class** Line **{**

**public** **void** draw**(int** x1**,** **int** y1**,** **int** x2**,** **int** y2**)** **{**

System**.*out*.**println**("Line from point A("** **+** x1 **+** **";"** **+** y1 **+** **"), to point B("** **+** x2 **+** **";"** **+** y2 **+** **")");**

**}**

**}**

**class** Rectangle **{**

**public** **void** draw**(int** x**,** **int** y**,** **int** width**,** **int** height**)** **{**

System**.*out*.**println**("Rectangle with coordinate left-down point ("** **+** x **+** **";"** **+** y **+** **"), width: "** **+** width

**+** **", height: "** **+** height**);**

**}**

**}**

**public** **class** NoAdapterProvided **{**

**public** **static** **void** main**(**String**[]** args**)** **{**

Object**[]** shapes **=** **{new** Line**(),** **new** Rectangle**()};**

**int** x1 **=** 10**,** y1 **=** 20**;**

**int** x2 **=** 30**,** y2 **=** 60**;**

**int** width **=** 40**,** height **=** 40**;**

**for** **(**Object shape **:** shapes**)** **{**

**if** **(**shape**.**getClass**().**getSimpleName**().**equals**("Line"))** **{**

**((**Line**)**shape**).**draw**(**x1**,** y1**,** x2**,** y2**);**

**}** **else** **if** **(**shape**.**getClass**().**getSimpleName**().**equals**("Rectangle"))** **{**

**((**Rectangle**)**shape**).**draw**(**x2**,** y2**,** width**,** height**);**

**}**

**}**

**}**

**}**

(3) (25 points) This question deals with the Facade Design Pattern as we discussed in class. You are to write a simple facade interface that will function as a wrapper that will “draw” the shape of Circle, Rectangle, and Square. Without this facade interface, you will have to call each of the shapes individually to do the drawing. You should have a class/java file for each of the shapes (Circle, Rectangle, and Square), a Shape interface, Shape Maker that serves as t Façade design pattern, and test program to run it. Be sure to provide an UML class diagram for your program.

Your final result should show:

Circle.draw() with a radius of 1.5

Rectangle.draw() ( 2.1,3.2)

Square.draw()( 2.5)