**COSC 436: Object-Oriented Design and Programming**

**In-class Exercise: Adapter Design Pattern**

**Problem:**

Below two classes, **Line** and **Rectangle**, are two problematic implementations. If the client wants to “draw” any of the two, he/she needs to check what type it is, which is very tedious.

**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** **AdapterDemo\_BadExample** {

**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) { // Need to check what type it is, NOT GOOD!

**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);

}

}

}

}

**Your task:**

We need to create adapters for Line and Rectangle so that they can be wrapped and the client can simply use the following code to “draw” them, without checking their types.

**public** **class** **AdapterDemo\_GoodExample** {

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

Shape[] shapes = {**new** RectangleAdapter(**new** Rectangle()), **new** LineAdapter(**new** Line())};

**int** x1 = 10, y1 = 20; **int** x2 = 30, y2 = 60;

**for** (Shape shape : shapes) { // No need to check what type it is, GOOD!

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

}

}

}

**Steps:**

1. From Blackboard, download the old **Line** and **Rectangle** classes to your IDE. Do not change anything in them.
2. From Blackboard, download the interface **Shape**. It specifies the method draw(int x1, int y1, int x2, int y2)
3. Create an adapter class **LineAdapter** and make it implement **Shape** interface.
4. In **LineAdapter**, create an instance variable for **Line**, name it **adaptee**.
5. The constructor of **LineAdapter** should accept a parameter of **Line** class, and assign it to the instance variable **adaptee**.
6. Implement the draw( … ) function by calling the original draw function in **Line** class.
7. Type the above new client code in your IDE and whether your **LineAdapter** works.
8. Do the similar thing for the old **Rectangle** class. (create **RectangleAdapter** for **Rectangle** class) HINT: You will have to add some calculation code for it.

Upload your code to the Blackboard when you are done. (Your program will NOT be graded.)