CECS 277 LAB ADAPTER PATTERN

OBJECTIVE: Get some first-hand experience building an adapter.

INTRODUCTION: Please remember the coding standards here.

There are all sorts of ways to define a rectangle in two-dimensional space. For this exercise, we will make the simplifying assumption that our rectangles all parallel the X and the Y axis, so that their sides are either "vertical" or "horizontal". The more general case is certainly more interesting, but I will leave that as an exercise for the student.

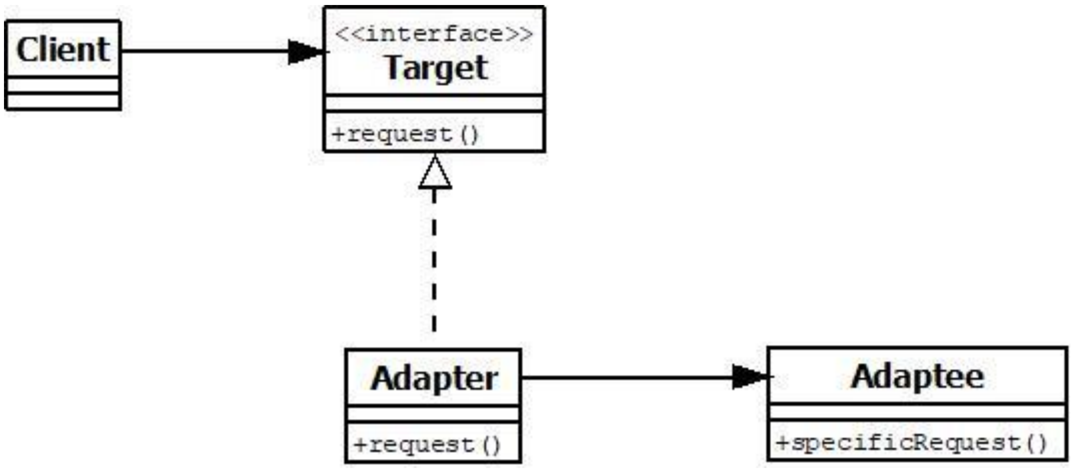
The company has been using an old set of classes for quite some time. To define a rectangle in that library, you gave the constructor four points: one for each of the four corners of the rectangle. Very flexible, but also caused loads of problems when people got the order of the corners wrong, or the sides of the rectangle were not perpendicular.

Just recently, we bought a new library that just requires the upper left hand corner of the rectangle (in the form of a Point object), the width, and the length of the rectangle. Presume that the width measures the rectangle parallel to the X axis and the length measures it parallel to the Y axis.

The old Rectangle class provided us with

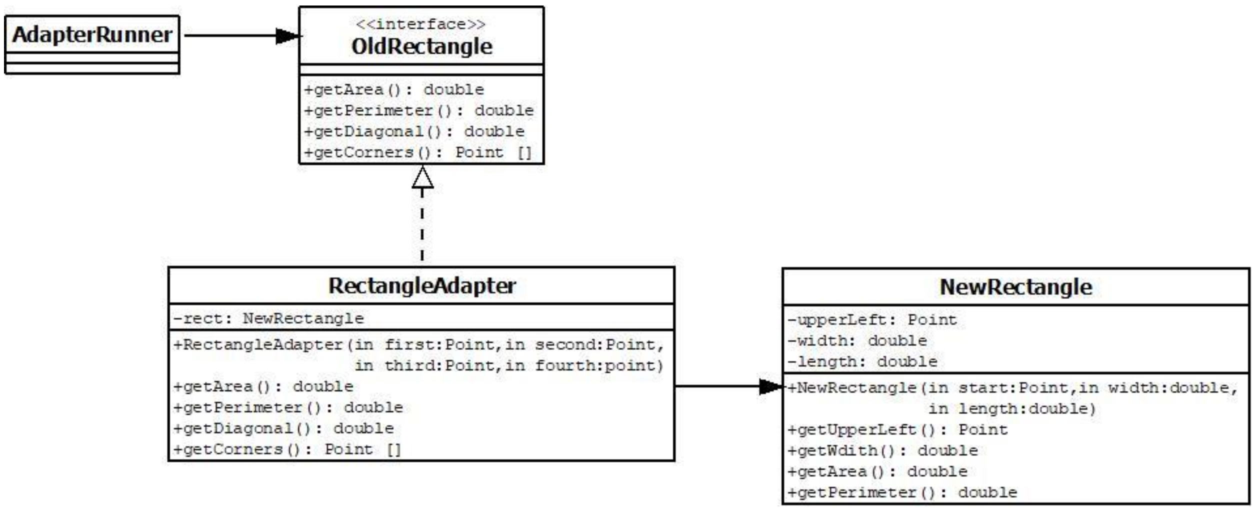
- `getArea()` – square area of the rectangle
- `getPerimeter ()` – the distance around the perimeter of the rectangle
- `getDiagonal()` – the distance between opposite corners of the rectangle
- `getCorners()` – An array comprising the corners of the rectangle

The generic Adapter pattern look like this in UML:



CECS 277 LAB ADAPTER PATTERN

For this lab, you need to implement an **instance** of this design pattern that looks like:



The code that you need for the Point class is here. The code for NewRectangle is here.

PROCEDURE:

1. Build the `OldRectangle` interface as shown in the UML above.
2. Build the `RectangleAdapter` class. Notice that `RectangleAdapter` has an instance of `NewRectangle` inside of it. This association is called a composition. The only reason that particular instance of `NewRectangle` exists is to serve as the storage and method implementation for key methods that `RectangleAdapter` needs to implement the `OldRectangle` interface.
3. Build `AdapterRunner` such that it declares a variable of type `OldRectangle` and populates it with an instance of `RectangleAdapter`. Then print out the output from all of the getter methods that the `OldRectangle` interface gives you.

WHAT TO TURN IN:

- OldRectangle.java
- RectangleAdapter.java
- AdapterRunner.java
- Your sample console output, in file: console.txt