Object-Oriented Programming: Doing More

Copy your first shapes Eclipse project and give it a new name. (Or, make a new project, and then copy any needed files from your first shapes project or from my "oop-basics-exercises" project). After you do the copying, you might want to close the first shapes project (R-click on the project in the Package Explorer on the left side and choose "Close project") so that you don't accidentally edit classes from the wrong project.

1. Change all your existing classes so that the fields are private and you have get*Xxx* and set*Xxx* methods to lookup and change the values of the fields. Run some test cases where you create a few shapes and then access their areas.

2. Have your Circle and Rectangle inherit from a common Shape class that has a getArea method. The best approach is to make Shape be an abstract class (or even an interface, but we haven't talked much about interfaces yet). However, if you are confused by abstract classes, just making Shape be a regular class is still pretty good. The really important idea is that you want to be able to make arrays that contain both Circle and Rectangle objects and access the area of each entry in the array. If you can do that, you have 95% of the problem solved.

3. Make a method that will take an array of `Shape` objects and sum their areas. Where is the best place to put this method? Make a test case consisting of an array of mixed shapes.

4. If you haven't already made a Square class, do so. Make your Square inherit from Rectangle, but still enforce the restriction that the width and the height are the same. Hint: override some method(s). You will find this problem to be a bit ugly, because you have two competing interests. On the one hand, you want squares to *be* rectangles because they are in real life. But on the other hand, the Rectangle class has separate width and height accessor methods that you can't totally get rid of in Square.

   Add some Square objects to your array of problem 3 and make sure it still works. You should not have to change the sumAreas method in *any* way.