

Object-Oriented Programming: Basics

The first three (page one) are the really important problems. The other problems are for those with more extensive previous OOP or Java experience.

1. Make a new Eclipse project called “shapes1”. Create a Circle class that contains a radius field. Give it a constructor where you pass in the radius. Have your test routine create a few circles, assign a value to the radius, then print out some information about the circles. Put your Circle class and your test routine in two separate classes, like this:

- **Circle.java**

```
public class Circle {
    public double radius;
    ...
}
```

- **CircleTest.java**

```
public class CircleTest {
    public static void main(String[] args) {
        Circle c = new Circle(...);
        ...
    }
}
```

Notes

- The Circle class does not have a main, so you cannot execute it directly. You only directly run the CircleTest class (R-click, Run As, Java Application).
 - This problem requires several different capabilities. Unless you have previous Java experience, I strongly recommend you build up to the solution in a piecemeal fashion. First make a Circle class with a radius field only, and test it out from the main in CircleTest. Then add in a constructor. Then test it out again. And so on. Use the four Person examples from the end of the lecture as a model for both your code and the iterative development/testing process.
2. Give your Circle a getArea method that calculates its area, and a printInfo method that prints out the radius and area. Make a test case that tries these capabilities out.
 3. Make a program that creates an array of 100 circles, each with a random radius. Print out the sum of the areas of the circles. Also print the biggest and smallest areas. Hint: remember that in the two-step array allocation process, the following line only makes *space* for 100 circles (or, more technically, it allocates an array of 100 null Circle pointers), it does not *create* any circles:

```
Circle[] circles = new Circle[100];
```

To actually create the circles, you have to do a loop:

```
for(int i=0; i<circles.length; i++) {
    circles[i] = new Circle(...);
}
```

Object-Oriented Programming: Basics (Continued)

4. Create a Rectangle class that contains width and height fields. Also give it a `getArea` method. Again, make a few test cases.
5. Create a Square class with width and `getArea`. Then, give both Square and Circle `setArea` methods that let you specify a desired area. Make a few test cases.
6. Questions to ponder:
 - Suppose you create a method that takes a Rectangle as an argument. Now suppose you want to pass a Square to it (after all, squares are rectangles, aren't they?). Why won't it work? From what we know so far, how could you fix this problem?
 - Since there is no particular relationship among Circle, Square, and Rectangle, what would you do if you wanted to make an array of mixed shapes, then loop down the array and sum up the areas?
 - Suppose that, for efficiency reasons in the Circle class, you wanted to make an area instance variable, instead of a `getArea` method. So, instead of

```
public double getArea() { return(Math.PI * radius * radius); }
```

you instead made the Circle constructor compute and store the area like this:

```
public Circle(double radius) { this.radius = radius; area = Math.PI * radius * radius; },
```

then you had `getArea` simply do this:

```
public double getArea() { return(area); }
```

Why will this strategy *fail* (at least with what we know so far about OOP in Java)?