Java Programming Exercises - Shapes

Written by Ravi Dudhagra -- https://github.com/rdudhagra

Welcome! Today, we will explore Java classes and interfaces by creating shapes.

Contents

Class/method structure

Classes are the base of Java. Here is the basic structure to make them (examples to follow):

```
[modifiers] class [class name] [extends [class name] (optional)] [implements
[interface name(s)]] {
    // Instance variables
    [standard syntax for declaring variables]

    // Constructors
    [modifier] [class name](parameters of constructor) {
        [constructor body]
    }

    // Methods
    [modifier] [return type] [method name]([parameters]) {
        [method body]
        [return [value/expression/variable] (if return type not void)]
    }
}
```

Abstraction

Our shape representation makes use of Java's class structure to represent the relationships that different shapes have to each other.

We first have to define a shape. What's something that all shapes have (we're assuming closed shapes)? For now, let's assume that all shapes have an area and a perimeter. We will get the area and perimeter of a shape through a method.

Open `Shape.java`. This is the base for all shapes. Note that this class is abstract. An abstract class is a class that we cannot make objects from; however, we can still extend it and make subclasses from it. Why is shape abstract? Well, let's say I ask you to make a shape? You reply, what kind of shape? Exactly. `Shape` is too vague on it's own to have any practical semblence. That's why we make it abstract.

Now open `Circle.java`. Understand the structure of a class, variables, and methods by reading through the code and associated comments.

Task 1

Open `Main.java`. Print out the circumference of the circle (see the example of how to print the area and work from there).

Task 2

In `Main.java`, set the radius of `circ` to `3`, and then print out the area and circumference again.

Now let's talk about polygons. A polygon is a shape with a certain number of sides. Polygons don't have curves in them. Because of this, there are many properties about polygons that we can say that are not true about shapes in general. Therefore, let's define a new class to reperesent all polygons.

Take a look at `Polygon.java`. Since we cannot just ask someone to draw a 'polygon' without being more specific, we make it abstract as well. `Polygon` extends `Shape` as expected, but we don't see any reference to `area` and `perimeter`! This is ok, since `Polygon` is abstract. However, any subclass of `Polygon` *will* need to implement methods in `Shape`, as well as methods in `Polygon`. What methods would those be (Hint: there are three)?

Now open `Square.java`. Read through the code and understand why the constructor, `getSideLength()` and `setSideLength()` are written the way they are.

Task 3

In `Square.java`, implement the `area()` method. If you need to square a number, look back at `Circle.java` for an example on how to do that.

Task 4

In `Square.java`, implement the `perimeter()` method.

Task 5

In `Main.java`, print out the area and perimeter of a square with side length 8. You **must** use the `Square` class to calculate these values.

> Note: The code you wrote in tasks 1, 2, and 5 are often referred to as test cases. Test cases are extremely important in making sure that your code does what it is supposed to. Not writing test cases is a recipe for disaster.

Task 6

In `Main.java`, write more test cases for Square. That means that you should do the following:

1. Create a `Square` (or multiple `Square`s)
2. Write code that tests that *each* of the methods in `Square` does what they are supposed to do

Task 7

In `Triangle.java`, create a new class called `Triangle`. `Triangle` should meet the following requirements:

1. Extend `Polygon`
2. Take in a `base` and `height` in its constructor (of type `double`), and store these values in variables
3. Return `-1` for the perimeter (Calculating the perimeter of a triangle is too mathy for this exercise (also would depend on the type of triangle), so don't worry about it). Add a comment next to your implementation of this method explaining the above.

Task 8

In `Main.java`, write test cases for `Triangle`, similar to `Square`.
