Exercise 5– Inheritance and Polymorphism

Objective

The objective of this exercise is to look at object hierarchies and implement some classes which extend from one another and use interfaces to define behaviours.

Overview

This exercise is going to look at three classes, `Shape`, `Rectangle` and `Circle` . There is also a 'helper' class called `Point` which stores an x and y coordinate pair allowing us to return the current coordinates of the shape.

Inheritance

1. Create a new Java project
2. Create a package called com.qa
3. Create the `Point` class
    a. This has two fields, x and y
    b. Generate a constructor, getters, setters and a toString() method for the class
4. Inside the package create a `Shape` class
    a. A shape has four fields, a String name, a String colour and two doubles for the position called x and y
    b. Write a constructor for the class, the getters, setters and toString() methods
    c. Make the class abstract and add two abstract methods:
        i. getArea() that returns a double
        ii. getCenterPoint() that returns a Point object

Now we want to write the concrete implementations of the class

5. Create a class called `Rectangle`

    Rectangles are a type of shape that also store the height and width of the rectangle. They have a method "isSquare" which returns a Boolean value stating if the rectangle is a square (if the height and the width are the same).

a. Rectangles are a sub-class of Shape, so we need to extend the Shape class at the class declaration

b. As well as the standard shape classes we want our rectangle to store a height and a width field, declare these variables in the class.

c. Now write the constructor for the Rectangle, it should take in all the parameters required to setup the Shape object and also the parameters required for the rectangle.

   Remember to use super(…) to call Shape's constructor with the required fields.

d. It's always useful to have a toString method for testing the class, so either write one yourself or automatically generate one using eclipse's built in options.

e. Implement the method isSquare(), this should return a Boolean value based on whether the height and the width are the same value

f. Finally, we need to override the abstract methods in the shape class. Implement the getArea() method and the getCenterPoint() method for the rectangle class

g. Test your method by creating a Main class with a main method in it. In this create a few rectangle objects and print out the contents of the objects using the toString() method.

6. Next we want to create a `circle` class

   A circle is a shape that also stores the radius of the shape.

   a. Create the circle class, remember to extend Shape

   b. Implement the new fields, constructor, toString and abstract methods for the class

      i. The circle class does not have a isSquare() method as it would make no sense!

      ii. The area of a circle is (πr^2) the value of PI is available for use from the Math package, see if you can find it in the API and use that rather than writing the number

   c. Test the class by creating a few circle objects in your main class and printing out their contents

Glossary of key terms

Inheritance

The process of creating a new class from an existing class

Abstract class

This class allows for the parent in a relationship to describe what methods should be there without providing an implementation

Method Overriding

A subclass can override its super class methods without modifying the method prototype