CPSC 250L Lab 5
Inheritance

Fall 2016

1 Introduction

This lab will focus on abstract classes and inheritance. In particular, know how to declare an abstract class, extend a class, declare abstract methods, override methods, and call methods of the super class. Additionally, we will be using the `Point` class from the Java API, so be sure to read the JavaDoc for `java.awt.Point`.

2 Exercises

Fork and clone the `cpsc250l-lab05` repository from the `cpsc250-students` group on Gitlab.

2.1 Abstract Class: Shape

In the first exercise, you will create an abstract class that will serve as the basis for the remaining exercises in this lab. **Do not do any other exercise until this one passes all of its jUnit tests!**

Exercise 1

Create an abstract class called `Shape` with two private fields:

- `String name`
- `Point[] points` (you will need to import `java.awt.Point`).

Implement the following methods.

1. `protected Shape(String aName)`

   This constructor should set the `name` field to the `aName` parameter.

2. `public final String getName()`

   Returns the name of the `Shape`.

3. `protected final void setPoints(Point[] thePoints)`

   Sets the `points` field to the `thePoints` parameter.

4. `public final Point[] getPoints()`

   Returns the points of the `Shape`.

5. `public static double getDistance(Point p1, Point p2)`

   Returns the Euclidean distance between $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$. That is, it returns

   $$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

Declare the following abstract methods.

1. `public abstract double getPerimeter()`
2. `public abstract double getArea()`

Test your code against `ShapeTest.java`. **Do not continue until all jUnit tests pass!**

Exercise 1 Complete

Run:

```
git add .
git commit -m "Completed exercise 1"
git push origin master
```

2.2 Subclass: Triangle

Exercise 2

Create a new class called `Triangle` that extends `Shape` and implement the following constructor.

- `protected Triangle(Point[] aPoints)` This constructor calls the superclass's constructor and passes it the name `"Triangle"`. It then passes a copy of the first three entries in the `aPoints` array to the `setPoints` method.

Override the following methods.

1. `public double getPerimeter()`

   This method calculates the perimeter of the `Triangle`. The perimeter of a triangle is the sum of the lengths its sides (to get the length of a side, calculate the distance between the two of its vertices).

2. `public double getArea()`

This method calculates the area of the `Triangle`. If the vertices of a triangle are $A = (a_x, a_y)$, $B = (b_x, b_y)$, and $C = (c_x, c_y)$, then the area of the triangle is given by the expression

$$\left| \frac{a_x(b_y - c_y) + b_x(c_y - a_y) + c_x(a_y - b_y)}{2} \right|.$$

Test your code against `TriangleTest.java`.

Exercise 2 Complete

Run:

```
git add .
git commit -m "Completed exercise 2"
git push origin master
```

2.3 Subclass: Quadrilateral

Exercise 3

Create a new class called `Quadrilateral` that extends `Shape` and implement the following constructor.

- `protected Quadrilateral(Point[] aPoints)`

  This constructor calls the superclass's constructor and passes it the name `"Quadrilateral"`. It then passes a copy of the first four entries in the `aPoints` array to the `setPoints` method.

Override the following methods.

1. `public double getPerimeter()`

   This method calculates the perimeter of the `Quadrilateral`. The perimeter of a quadrilateral is the sum of the lengths its sides (to get the length of a side, calculate the distance between the two of its vertices).

2. `public double getArea()`

   This method calculates the area of the `Quadrilateral`. If the vertices of a quadrilateral are $A = (a_x, a_y)$, $B = (b_x, b_y)$, $C = (c_x, c_y)$, and $D = (d_x, d_y)$, then the area of the quadrilateral is given by the expression

$$\left| \frac{(a_x b_y - a_y b_x) + (b_x c_y - b_y c_x) + (c_x d_y - c_y d_x) + (d_x a_y - d_y a_x)}{2} \right|.$$

Test your code against `QuadrilateralTest.java`.

Exercise 3 Complete

Run:

```
git add .
git commit -m "Completed exercise 3"
git push origin master
```

2.4 Subclass: Circle

Exercise 4

Create a new class called `Circle` that extends `Shape` and implement the following methods.

1. `protected Circle(Point aCenter, double radius)`

   This constructor calls the superclass's constructor and passes it the name `"Circle"`. It then passes an array containing only the point `aCenter` to the `setPoints` method. Additionally, it stores `radius` in a private field. However, if `radius` is negative it will store `0.0` as the radius.

2. `public double getRadius()`

   This returns the radius of the `Circle`.

Override the following methods.

1. `public double getPerimeter()`

   This method calculates the perimeter of the `Circle`. The perimeter of a circle is given by the expression

   $$2\pi r$$

   where r is the radius of the circle.

2. `public double getArea()`

   This method calculates the area of the `Circle`. The area of a circle is given by the expression

   $$\pi r^2$$

   where r is the radius of the circle.

Test your code against `CircleTest.java`.

Exercise 4 Complete

Run:

```
git add .
git commit -m "Completed exercise 4"
git push origin master
```

3 Common Mistakes

The solutions to some common mistakes are as follows.

1. Make sure that `Shape` passes all of its tests before proceeding to the last three exercises. If it doesn't, it will adversely affect the remaining exercises.
2. Be sure to extend the `Shape` class! If you do not, you will fail the jUnit tests.
3. Before running the tests for any of the subclasses, make sure that `getPerimeter` and `getArea` are implemented (even if its just `return 0.0;`).