C212/A592 Spring 17 Lab8

Intro to Software Systems

Instructions:

- Review the requirements given below and complete your work. Please submit all files through Canvas.
- The grading scheme is provided on Canvas

Lab 8 Random Shape Generator

Part 1 – Introduction and abstract Shape class

- Lab 8 may be used as the first part of Assignment, writing an application that generates and draws different shapes of random colors, random sizes, and in random locations.

- I added a .jar file of the final application so you can get the idea of what the classes are for. Download the .jar file and double click to run the app.

    - Pressing the c, r, or s key draws a circle, rectangle, or shape

- The goals of this lab are:
    - inheritance
    - method overloading
    - method overriding
    - constructor chaining
    - polymorphism (will manifest more in the HW extension)
    - abstract classes
    - all very important OOP concepts that come up in job interview questions.

- You will create a Square, Rectangle, and Circle class that are all extended from an abstract Shape class

- The shape classes will contain state information for the object to be drawn

Note: Some of the methods will not be used now and we will come back to them to add more functionality. The way I designed this lab might feel somewhat contrived, which was intentional to emphasize the concepts.

- Below is the API for the shape class:
    - all methods that are not abstract are common to all shapes
    - the abstract methods are methods all shapes will have, but will be different for each shape

```
// we have not covered awt graphcis yet, but you do not need to know much for this assignment
// we will be covering next week and using them for the HW
import java.awt.Color;

abstract class Shape {

   private Color fillColor;
   private Color borderColor;
   private Boolean isFilled;
   private Point Location;

   // the three constructors initialize the instance fields
   public Shape(Color fillColor, borderColor, int x, int y) { }

   // set borderColor to Black since not provided
   public Shape(Color, fillColor, int x, int y) { }

   // set fillColor to white and border color to black
   public Shape(int x, int y { }

   public void setFillColor(Color c) { }

   public Color getFillColor() { }

   public void setBorderColor(Color c) { }

   public Color getBorderColor() { }

   public Point getLocation() { }

   // Note:  subclasses of Shape do not inherent private members so we need methods the subclasses
   // can use to get the x and y values from the private Point instance field
   public int getX() { }

   public int getY() { }

   // if fillColor is white returns true, else returns false
   public boolean isFilled() { }

   // moves location by dx and dy
   private void moveLoc(int dx, dy) { }

   abstract double getArea();

   abstract double getPerimeter();

}
```

Part 2 – Extending the Shape class

- Extend the shape class to make a Circle class and Rectangle class
- Extend the Rectangle class to make a Square class

- All extended Shape classes should also implement a toString method – it is up to you how you want to represent the Shape as a String

- The Circle, Square, and Rectangle should have the same number of constructors with the same arguments, **in addition to** any other fields that are needed to create those shapes

    - To initialize the private instance fields of the Shape class you will need to make calls to the super class constructor
        - private fields and methods of a super class are not directly accessible in the sub classes as if you were in the same class
        - they still belong to the object but cannot be referenced directly
    - Then initialize fields specific to the particular Shape class you are implementing

- If done correctly the Square class will have the least amount of code because it can reuse most of the Rectangle class methods
    - Hint: use super class constructors correctly and all this class should have is constructors and a toString method

    - **NOTE:** getting the previous bullet correct and the calls to super class constructors is part of the grading criteria