Csci 1933 Project #1: Object-oriented geometry and graphics

Due date: 9/27/2017, before midnight

100 points.

Summary. Project #1 will involve implementing several shape classes in Java and using a drawing class we have implemented for you to create some graphics, including a fractal based on your shapes.

Part I. Defining and Implementing Shape Classes (70 points)

You should design classes for three different shapes, triangle, square, and circle, with the following specifications. These classes are to be used by the main program (the class with the "main" method), and passed to the Canvas class for drawing. Thus, they must use standard class names and method names in order to work correctly. The jUnit tests we will provide you with will also require standardized names, so if your tests pass, then your names are satisfactory.

Circle class

Methods:

- Name: Circle (constructor); input: x position (double), y position (double), radius (double); output : object of type Circle
- Name: calculatePerimeter; input: none; output: perimeter of the circle (type double)
- Name: calculateArea; input: none; output: area of the circle (double)
- Name: setColor; input: color of the shape (type Color); output: none
- Name: setPos; input: x, y position of the center (both doubles) output: none
- Name: setRadius; input: radius (double) output: none
- Name: getColor; input: none; output: color of the shape (type Color);
- Name: getXPos; input: none; output: x position of the center (double)
- Name: getYPos; input: none; output: y position of the center (double)
- Name: getRadius; input: none; output: radius (double)

Any data members needed to support these methods.

Note: the Color class is implemented in the java.awt library, which you can use by including the following statement at the top of your class definition file: `import java.awt.*;` See the Java API documentation for more details on this class (note the static data fields, e.g. `Color.BLUE`).

Triangle class

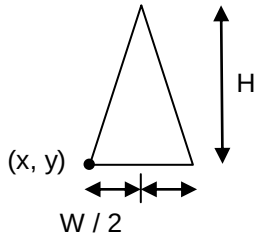
Methods:

- Name: Triangle (constructor); input: x position of bottom left corner (double), y position of bottom left corner (double), width (double), height (double); output: object of type Triangle
- Name: calculatePerimeter; input: none; output: perimeter of the triangle (type double)

- Name: calculateArea; input: none; output: area of the triangle (double)
- Name: setColor; input: color of the shape (type Color); output: none
- Name: setPos; input: x, y position of bottom left corner (both doubles) output: none
- Name: setHeight; input: height (double) output: none
- Name: setWidth; input: width (double) output: none
- Name: getColor; input: none; output: color of the shape (type Color);
- Name: getXPos; input: none; output: x position of the bottom left corner (double)
- Name: getYPos; input: none; output: y position of the bottom left corner (double)
- Name: getHeight; input: none; output: height (double)
- Name: getWidth; input: none; output:width (double)

Any data members needed to support these methods.

HINT: You can assume the triangles we want to draw are isosceles triangles, i.e.:



Rectangle class

- Name: Rectangle (constructor); input: x position of bottom left corner (double), y position of bottom left corner (double), width (double), height (double); output: object of type Rectangle
- Name: calculatePerimeter; input: none; output: perimeter of the rectangle (type double)
- Name: calculateArea; input: none; output: area of the rectangle (double)
- Name: setColor; input: color of the shape (type Color); output: none
- Name: setPos; input: x, y position of bottom left corner (both doubles) output: none
- Name: setHeight; input: height (double) output: none
- Name: setWidth; input: width (double) output: none
- Name: getColor; input: none; output: color of the shape (type Color);
- Name: getXPos; input: none; output: x position of the bottom left corner (double)
- Name: getYPos; input: none; output: y position of the bottom left corner (double)
- Name: getHeight; input: none; output: height (double)
- Name: getWidth; input: none; output: width (double)

Any data members needed to support these methods.

To verify your classes are implemented correctly, you should run the jUnit test classes, which we have provided for you.

Part II. Writing a program that draws a fractal based on your shape classes (30 points)

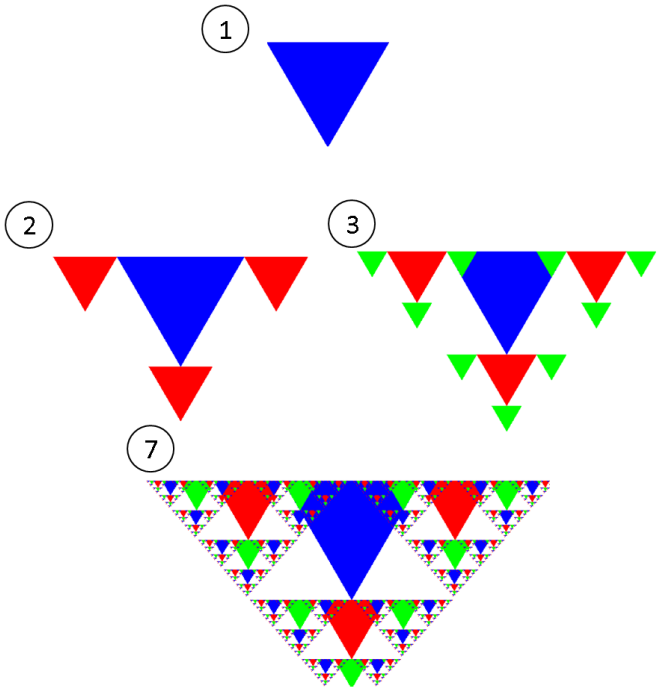
For this part of the project, we will write a Java program that uses your shape classes to draw a fractal. To help you out, we've already implemented a Canvas class that supports all of the drawing capability you need. You should look at the code if you're interested, but all you'll need to know is how you can interact with Canvas objects. Here are the method specifications of the Canvas class:

- Canvas() (default constructor): creates drawing of default size
- Canvas(int height, int width): creates drawing of specified width and height
- void drawShape(Circle circleObj)
- void drawShape(Rectangle rectangleObj)
- void drawShape(Triangle triangleObj)

Each of the drawing methods will draw the shape you pass it in the specified location and in the specified color. Here's an example of how you might use the Canvas class to draw a blue circle:

```
Canvas drawing = new Canvas(800,800);
Circle myCircle = new Circle(0,0,100);
myCircle.setColor(Color.BLUE);
drawing.drawShape(myCircle);
```

Your task is to write a Java program that uses your shapes and our Canvas class to draw a fractal. Fractals are geometric patterns that repeat on themselves at smaller and smaller scales. They have been studied for centuries because of their interesting mathematical properties and often appear in natural objects (e.g. snow flakes, plants). You can read more about fractals and their history here: http://en.wikipedia.org/wiki/Fractal. Consider the example below, which illustrates the process of constructing a fractal composed of triangles at several steps in the process. Notice that at each step, triangles of increasingly smaller sizes are drawn at the three points of each existing triangle.



Your goal is to write a Java program that uses your shape classes and our Canvas class to draw a fractal like this. For full credit, your program should have the following features:

- ask the user for input (choices: "circle", "triangle", or "rectangle") and use the corresponding shape as the base shape of your fractal
- draw a pattern that repeats on itself at least 7 times
- compute the total area of any shapes that form your fractal and print the result to the screen after your program is finished drawing

Note that to receive full credit, your program will need to be able to draw a fractal for all three possible inputs (circle, triangle, or rectangle). We suggest that the most convenient implementation is to simply implement three different methods, one that draws a circle fractal, one that draws and triangle fractal, and one that draws a rectangle fractal.

Submitting your finished assignment:
Once you've completed Project #1, create a zip file with all of your Java files (.java), including one that implements your main program, and submit it through Moodle.

Working with a partner:
As discussed in lecture, you may work with one partner to complete this assignment (max team size = 2). If you choose to work as a team, please only turn in one copy of your assignment. At the top of your class definition that implements your main program, include in the comments both of your names and student IDs. In doing so, you are attesting to the fact that both of you have contributed substantially to completion of the project and that both of you understand all code that has been implemented.

Some guidelines about how we will grade your project:
To give you a sense for the grade your assignment will receive, here are some examples of what will be required for each grade level:

A-level assignment:

- Properly defined Shape classes with correctly implemented methods
- JUnit test passes
- Main program implemented that correctly receives input from user
- Correct instantiation/use of the Canvas class
- Working fractal drawing (pattern repeated at least 7 times)
- Total shape area computed and printed
- Good programming style (spacing and comments)

B-level assignment:

- Properly defined Shape classes with correctly implemented methods
- JUnit test passes
- Main program implemented that correctly receives input from user
- Correct instantiation/use of the Canvas class
- Partially working fractal drawing

C-level assignment:

- Properly defined Shape classes with correctly implemented methods
- JUnit test passes