CSCI 1933 Project #1: Object-oriented geometry and fractals

Due date: Friday, 2/11/2022, before 7pm
Total Points: 100

Summary. This project will involve implementing several shape classes in Java and using a drawing class, that we have implemented and provided to you, to create some graphics that will include a fractal based on your shapes.

1. Defining and Implementing Shape Classes (50 points)

You should design classes for three different shapes (triangle, square, and circle) using the specifications listed below. These classes will be used in a main method (that you will implement in the next part of the project) and passed to the Canvas class for drawing. Thus, each class must use standard class names and method names to work correctly.

Circle class

Methods:

- Name: Circle (constructor); **input**: x position (double), y position (double), radius (double)
- 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)

Member variables:

Any data members needed to support these methods.

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

```
import java.awt.*;
```

Triangle class

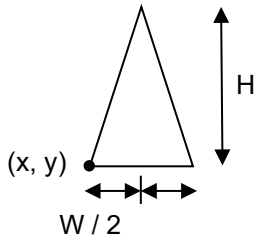
Methods:

- Name: Triangle (constructor); **input**: x position of bottom left corner (double), y position of bottom left corner (double), width (double), height (double)
- 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; i**nput**: 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)

Member variables:

Any data members needed to support these methods.

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



Rectangle class

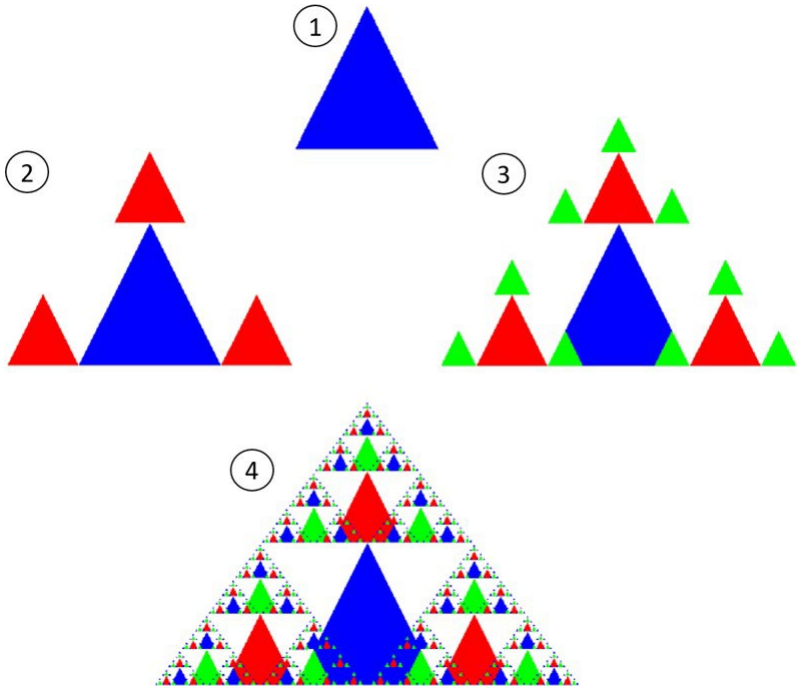
- Name: Rectangle (constructor); **input**: x position of bottom left corner (double), y position of bottom left corner (double), width (double), height (double)
- Name: calculatePerimeter; **input**: none; **output**: perimeter of the rectangle (type double)
- Name: calculateArea; **input**: none; **output**: area of the rectangle (double)
- Name: setColor; **input**: color of the shape (type Color); **output**: none
- Name: setPos; **input**: x position of bottom (double), y position of left (double) (i.e., bottom left corner); **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)

Member variables:

Any data members needed to support these methods.

2. Implementing a Fractal Drawer class (40 points)

For this part of the project, you will write a Java program that uses your shape classes 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., snowflakes, 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.



To help you with this drawing, we have already implemented a Canvas class that supports all the drawing capability you need. You should look at the code as you will need to know 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 is an example of how you might use the Canvas class to draw a blue circle:

```
// Creates a new Canvas object
Canvas drawing = new Canvas(800,800);
// Creates a new Circle object
Circle myCircle = new Circle(0,0,100);
```

```
// Set the color of the Circle object
myCircle.setColor(Color.BLUE);
// Draw the Circle object
drawing.drawShape(myCircle);
```

We have given you a skeleton class for a class called FractalDrawer. You will be responsible for implementing this class, which includes completing all to-dos marked in the provided .java file. 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 of your fractal
- draw a pattern that repeats on itself at least seven times and uses a new color in each repetition (feel free to be creative - you can cycle colors or choose colors at random, if overlapping shapes are distinct from one another)
- draw at least four shapes per layer for rectangles and circles and at least three per layer for triangles (e.g., draw the new shapes on all three points or sides of a triangle, or on four opposite sides of a circle - again, feel free to be creative)
- compute the total area of all shapes that form your fractal and print the result to the screen after your program has finished drawing.

Note: The main method should collect the user data, draw the fractal, and then print the area of the drawn fractal.

To receive full credit, your program will need to be able to draw a fractal for all three 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 a triangle fractal, and one that draws a rectangle fractal. If nothing appears on the canvas when you run your program, try resizing the canvas window.

3. Working with a partner:

As discussed in lecture, you may work with one partner to complete this project (max team size = 2). If you choose to work as a team, **please only turn in one copy of your assignment.** Be sure to include both team members’ information in the README file included with your assignment (see details below).

4. Submitting your finished assignment:

Be sure to include your name(s) and x500 in a comment in all files you submit.
Example: // Written by Chad Myers, chadm

Submit a .zip or .tar archive on Canvas containing all your .java files and a README.txt file (see details below). Your archive file should contain the following .java files:
Classes you created:

- Circle.java
- Rectangle.java

• Triangle.java

Classes you edited:

• FractalDrawer.java

Classes we gave to you (do not edit this file, but you should include with your submission):

• Canvas.java

You can change or modify your submission on Canvas, and we will grade your latest submission. Please verify that all your .java files are in your submissions. Failure to submit the correct files will result in a score of zero for all missing parts. Late submissions will be penalized in accordance with the syllabus.

Be sure to include a README.txt **(10 Points)** in your submission that contains the following information:

- Group members' names and x500s
- Contributions of each partner (if working with a partner)
- How to compile and run your program
- Any assumptions
- Additional features that you implemented (if applicable)
- Any known bugs or defects in the program
- Any outside sources (aside from course resources) consulted for ideas used in the project, in the format:
    - Idea1: source
    - Idea2: source
    - Idea3: source
- Include the statement: "**I certify that the information contained in this README file is complete and accurate. I have both read and followed the rules described in the "Course Policies" section of the course syllabus.**" Type your name(s) underneath this statement to indicate you agree with it.

You will lose all ten points for the README if it is missing or incomplete.

5. Grading information:

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
- Main program implemented that correctly receives input from user
- Correct instantiation/use of the Canvas class
- Working fractal drawing (pattern repeated at least seven times)
- Total shape area computed and printed
- Good programming style (spacing and comments)

B-level assignment:

- Properly defined Shape classes with correctly implemented methods
- 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
- Main program implemented that correctly receives input from user

6. Fractal examples:

Here are some examples of fractals for all three shapes if you want some inspiration. Your fractal patterns do not have to match these; feel free to make your own patterns, as long as they meet the basic requirements listed in Section 2!