COS126 Programming Exam 2 (Spring 2015)

Instructions. This exam is like a mini programming assignment. You will have 90 minutes to create several programs. Debug your code as needed. You may use your book, your notes, your code from programming assignments, the code on the COS 126 course website, the booksite, and you may read Piazza. No form of communication is permitted (e.g., talking, email, IM, texting, cell phones) during the exam. When you are done, submit your program via the course website using the submit link on the "Assignments" page.

Grading. Your program will be graded on correctness, clarity (including comments), design, and efficiency. You will lose a substantial number of points if your program does not compile or does not have the proper API, or if it crashes on typical inputs.

This paper. In addition to submitting your code electronically, you must turn in this paper. Print your name, login, and precept on this page. Copy and sign the Honor Code pledge.

Discussing this exam. As you know, discussing the contents of this exam before solutions have been posted is a serious violation of the Honor Code.

NAME: _____

LOGIN: _____

PRECEPT: _____

"I pledge my honor that I have not violated the Honor Code during this examination."

_____

_____

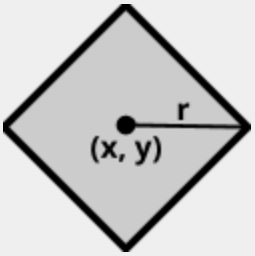
SIGNATURE: _____

Part 1. Diamond.java (10 points)

Create a class that represents a two-dimensional diamond using the following API.

```
public class Diamond
-----------------------------------------------------------
Diamond(double r, double x,  // creates a Diamond object of
        double y, Color c)   // radius r, centered at (x, y)
                             // color c but does not draw it

void drawFilled()            // sets the pen color & draws
                             // the diamond using StdDraw

double area()                // returns area of the diamond
                             // where area = 2 * r^2

void changeColor(Color c)    // changes the color, but
                             // does not draw the diamond

void move(double deltaX,     // changes (x, y) by deltaX and
          double deltaY)     // deltaY respectively, but
                             // does not draw the diamond
```

Diamonds



The coordinates of the points of a diamond are **r** units from the center point (**x**, **y**).

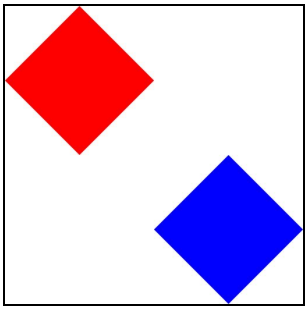
Remember to put **`import java.awt.Color;`** at the beginning of your program.

Use this **`main`**`()` method to test your code. Further testing is *strongly recommended* but not required.

```
public static void main(String[] args) {
    StdDraw.setXscale(-1, 1); // set scale
    StdDraw.setYscale(-1, 1); // in main()
    Color red = StdDraw.RED;
    Diamond d = new Diamond(.5, -.5, .5, red);
    d.drawFilled();
    System.out.println(d.area());
    d.changeColor(StdDraw.BLUE);
    d.move(1, -1);
    d.drawFilled();
    System.out.println(d.area());
}
```

Run your program in DrJava. You should get the following output:

```
> java Diamond
0.5
0.5
```



You can download this **`main()`** code at the following URL:

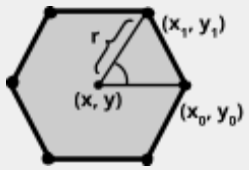
http://www.cs.princeton.edu/~cos126/docs/data/Ngon

Part 2. Ngon.java (15 points)

Create a class that represents a two-dimensional N-sided regular polygon using the following API.

```
public class Ngon
-----------------------------------------------------------
Ngon(int n, double r,      // n-sided polygon, radius r,
     double x, double y,   // center (x, y), color c
     Color c)

Ngon(int n, double r)      // as above, except centered at
                           // (0, 0) & color StdDraw.BLACK

void drawFilled()          // sets pen color & draws Ngon

double area()              // area of Ngon = (1/2) * (r^2)
                           // * (# of sides) * sin(angle)

Ngon merge(Ngon that)      // merges w/ an Ngon (see below)
```

Regular Polygons



Interior angle, in radians:
angle = *2 * pi / # of sides*

Coordinates of the points:
$x_i = x + r * \cos(i * angle)$
$y_i = y + r * \sin(i * angle)$
where i is 0, 1, ..., N-1

The **merge**() method creates a new (bigger) Ngon such that:

- The radius of the new Ngon is sum of the two radii.
- The center point coordinates (x, y) are the averages of the x and y values respectively.
- The color RGB values are the averages of the R, G, and B values respectively.

Note, **merge()** must return **null** if the two Ngons have different numbers of sides.

We provide an example below, in which a **red** hexagon is merged with a **blue** hexagon.

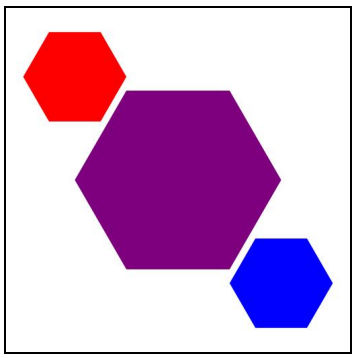
Use this **main**() method to test your code. Further testing is *strongly recommended* but not required.

```
public static void main(String[] args) {
    StdDraw.setXscale(-1, 1); // set scale
    StdDraw.setYscale(-1, 1); // in main()
    Color red = StdDraw.RED;
    Color blue = StdDraw.BLUE;
    Ngon h1 = new Ngon(6, .3, -.6, .6, red);
    Ngon h2 = new Ngon(6, .3, .6, -.6, blue);
    Ngon h3 = h1.merge(h2);
    h1.drawFilled();
    h2.drawFilled();
    h3.drawFilled();
    System.out.println(h3.area());
}
```

You can download this code at the URL provided in Part 1.

Run your program in DrJava. You should get the following output:

```
> java Ngon
0.9353074360871938
```



Part 3. NgonArt.java (5 points)

Create a client for Ngon that reads the parameters of some Ngons from StdIn and plots them in descending order of area (i.e., plots the Ngon with the largest area first).

You may write your own private methods to help you accomplish this goal, or you may write your entire program in `main()`. To simplify your task, assume that no two Ngons will have exactly the same area.

As you did in the previous parts, set the scale to (-1, 1) for both axes.

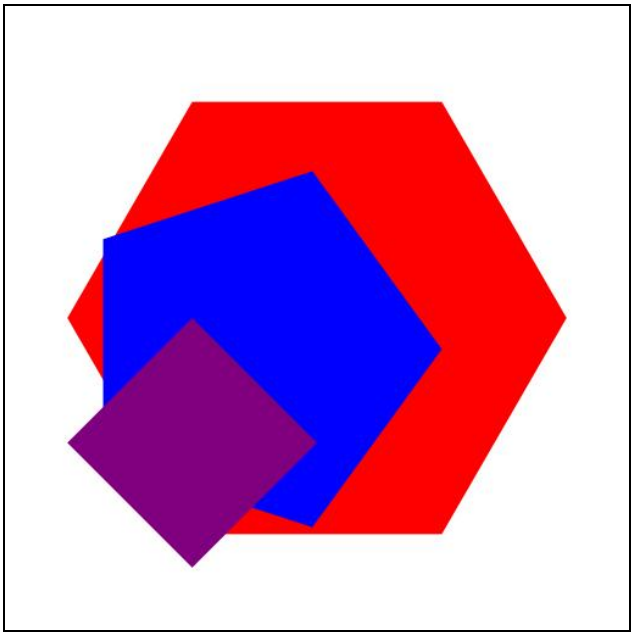
You can download the following sample input files at the URL provided in Part 1.

The file **3gon.txt** contains 3 Ngons:

```
3
6 0.8  0.0  0.0 255 0 0
4 0.4 -0.4 -0.4 127 0 127
5 0.6 -0.2 -0.1 0   0 255
```

Run your program from the command line. You should get the following output:

```
> java-introcs NgonArt < 3gon.txt
```

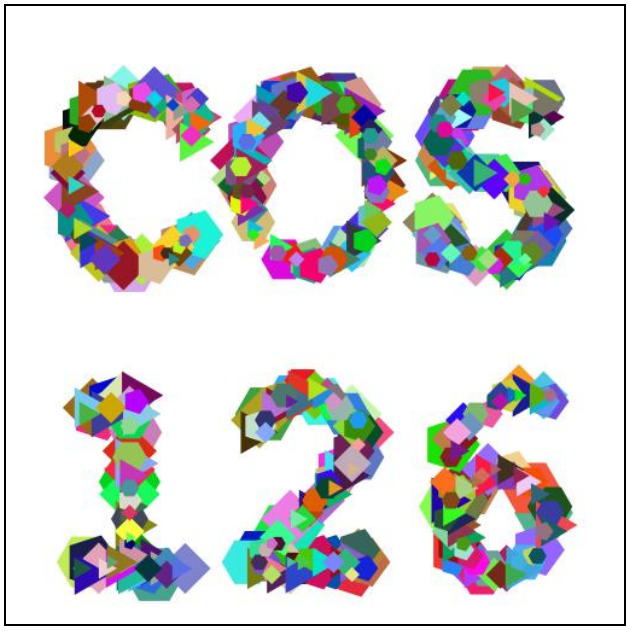


The file **cos126gon.txt** contains 478 Ngons:

```
478
3 0.07617048 -0.74 -0.84 47 10 180
3 0.07334798 -0.60 -0.84 32 187 158
        ... and many more ...
```

Run your program from the command line. You should get the following output:

```
> java-introcs NgonArt < cos126gon.txt
```



Note: You may not use java.util.*