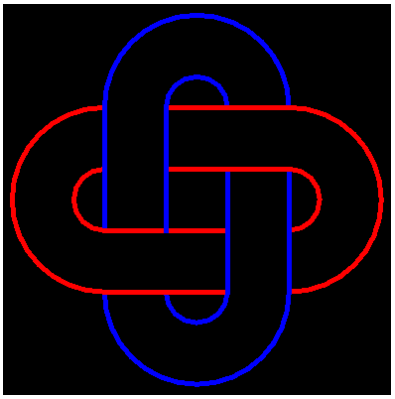
Project 11: Java Art

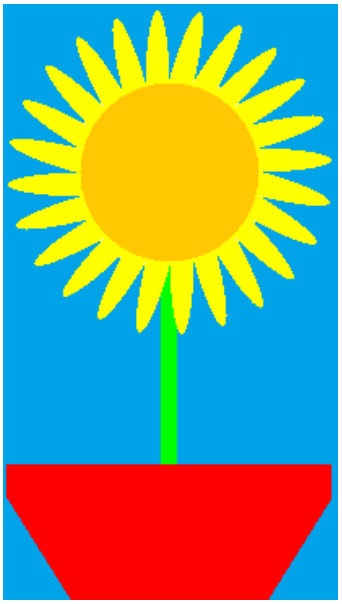
Due: Monday, Apr. 13 at 11:59 PM

Description: One of the benefits of object-oriented programming is that we can use classes written by other programmers without understanding all the underlying details. To use a class, we only need to know its method signatures.

Most classes include methods to create, change, and access data in objects of the class. These kinds of methods are called "constructors," "mutators," and "accessors" respectively. For Java API classes, the signatures of these methods can be found in the documentation.



Interlocking Chains by James Dizikes



Daisy by Deborah Trytten

The Java API includes some wonderful classes for creating graphics. In this project, we will use several of these classes to make some art! The images shown here are examples from previous semesters.

The purpose of this project is to give you practice working with classes that you have never seen before. In order to use these classes, you will need to read about them in the API documentation. Remember that the documentation contains details that can be confusing to new programmers, so skip the parts that don't make sense. Focus on the signatures of the methods that construct and change objects of the classes.

Objectives: Write a program that creates an image and satisfies the requirements below.

1. (30 points) Construct objects from at least three of the following classes: Polygon, Rectangle2D.Double, RoundRectangle2D.Double, Ellipse2D.Double, and Arc2D.Double.
2. (30 points) Call at least one method on each of the objects constructed for objective 1. (Note that the constructor does not count.) For instance, if you construct a Polygon object, you can call addPoint to add points to the shape.
3. (15 points) Use at least three colors from the Color class. These can be constants such as Color.RED, or you can create custom colors by constructing Color objects.
4. (15 points) Call at least three of the following methods from the Graphics2D class: draw, fill, setColor, setStroke, drawString, rotate, scale, and translate.
5. (10 points) Use meaningful variable names, consistent indentation, and whitespace to make your code readable. Add comments to explain the overall steps of your program.

Coordinate System: For historical reasons, computer graphics libraries often use a coordinate system with an inverted y-axis.[1] This coordinate system is shown in the image below.

When writing code to construct shape objects, keep in mind that increasing the y-coordinate of a point moves it lower on the screen. For example, the point (0, 100) is located 100 pixels *below* the origin.

Starter Code: The program below creates a window that is 640 pixels wide and 480 pixels tall. Two lines are drawn across the middle of the window: a horizontal blue line from (0, 240) to (640, 240) and a vertical red line from (320, 0) to (320, 480).

```
import javax.swing.*;
import java.awt.*;
import java.awt.geom.*;

public class JavaArt extends JPanel
{
    // Change these constants to adjust the size and title of the window.
    private static final int WIDTH = 640;
    private static final int HEIGHT = 480;
    private static final String WINDOW_TITLE = "My Image Title!";

    // Do not change the main method.
    public static void main(String[] args)
    {
        JPanel panel = new JavaArt();
        panel.setPreferredSize(new Dimension(WIDTH, HEIGHT));

        JFrame frame = new JFrame(WINDOW_TITLE);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
        frame.add(panel);
        frame.pack();
    }
```

[1] Before flat screen displays were common, computer monitors and televisions used cathode ray tubes to produce images. This technology works by scanning a beam of electrons across a phosphorescent screen from left to right and top to bottom. The inverted y-axis points in the vertical direction that the beam is scanned.

```
    // Add your drawing code to this method.
    public void paintComponent(Graphics g)
    {
        // Do not change these two lines.
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g;

        // Replace the example below with your code.

        // Draw a horizontal blue line in the middle of the screen.
        Line2D.Double line = new Line2D.Double(0, HEIGHT/2, WIDTH, HEIGHT/2);
        g2d.setColor(Color.BLUE);
        g2d.draw(line);

        // Draw a vertical red line in the middle of the screen.
        line.setLine(WIDTH/2, 0, WIDTH/2, HEIGHT);
        g2d.setColor(Color.RED);
        g2d.draw(line);
    }
}
```

Note the following things about this code:

1. The import statements include the character '*'. This is a "wildcard character" that represents the name of any class. For example, the statement "import javax.swing.*" imports all the classes in the javax.swing package.
2. The static constants WIDTH, HEIGHT, and WINDOW_TITLE can be accessed in any method. We will discuss static variables later in the semester. For now, just know that these variables can be used in a method without passing them as an argument.
3. The constants WIDTH and HEIGHT specify the size of the window. Adjust these as necessary to make the window large enough to display your entire image.
4. The constant WINDOW_TITLE stores the string that appears in the window title bar. Use this variable to give your artwork a fancy title!
5. The class declaration includes the keyword "extends," which we are unfamiliar with. This keyword is used to define a class that "inherits" properties from another class. You will learn more about this if you take CS 2334. For now, don't worry about this code.
6. The code that creates your image should be added to the method paintComponent after the first two lines of code. You do not need to change the main method.

Implementation Suggestions: Copy the starter code into a source file named "JavaArt.java" in a new Eclipse project. Fix any indentation and spacing issues and then run the program. You should see a window with a horizontal blue line and a vertical red line. Replace the code that draws these lines with code to make an image of your choosing. You can create any image you like, just make sure the program satisfies the objectives given on the first page.

Study the example code to see how to draw shapes in the window. First construct a shape object with the desired size and position. Then draw the object by calling a method of the Graphics2D object.

Methods such as setColor, setStroke, and rotate cause a lasting change in the Graphics2D object. For example, if the color is set to blue, all objects will be drawn in blue until a different color is set.

When designing an image, sketch your idea on paper before writing code. Think carefully about the order to draw the shapes. Choosing an appropriate order can make drawing an image easier. For example, the petals of the daisy shown on the first page are made by drawing an ellipse multiple times in different orientations (using the rotate method). After this is done, an orange circle is drawn on top of the ellipses. Drawing the circle first and then the petals would require a bunch of difficult math to determine where the petals should start and stop.

Upload Code to Zybooks: After you complete each project, you need to upload your source code to Zybooks so it can be graded. If you created the folder structure described earlier, your code is in the folder "Intro to Programming\Projects\Project 11\src". The code is contained in the file Project_11.java. Note that .java files are simply text files that contain Java code. They can be opened with any text editor (e.g., Notepad or TextEdit).

Upload your .java file to Zybooks on the Project 11 assignment page. This requires a few steps. Click the "Submit Assignment" button in the top-right corner. This will reveal a button near the bottom of the page with the text "Choose File." Click this button and browse to the location of your .java file. Finally, click the "Submit for grading" button below the "Choose on hard drive" button.

If Zybooks won't accept the file, make sure you're submitting a .java file, not a .class file. Make sure to see if the output you obtained and the output which I have mentioned match to obtain complete grade.