Computer Graphics Program in Java

Bachelor in Information Management

Banepa NIST College
Computer Graphics
***Compiled By*: Dinesh Ghemosu**

July 29, 2023

Program 1: Displaying Different Text

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

public class DifferentText extends JFrame {
    private final int WIDTH = 800;
    private final int HEIGHT = 600;

    public DifferentText() {
        setSize(WIDTH, HEIGHT);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        displayText(g);
    }

    private void displayText(Graphics g) {
        // Text 1: Color RED, Size 12
        g.setColor(Color.RED);
        g.setFont(new Font("Arial", Font.PLAIN, 12));
        g.drawString("Text in RED, Size 12", 100, 100);

        // Text 2: Color BLUE, Size 18
        g.setColor(Color.BLUE);
        g.setFont(new Font("Arial", Font.BOLD, 18));
        g.drawString("Text in BLUE, Size 18", 100, 150);

        // Text 3: Color GREEN, Size 24
        g.setColor(Color.GREEN);
        g.setFont(new Font("Arial", Font.ITALIC, 24));
        g.drawString("Text in GREEN, Size 24", 100, 200);

        // Text 4: Color ORANGE, Size 30
        g.setColor(Color.ORANGE);
        g.setFont(new Font("Arial", Font.BOLD | Font.ITALIC, 30));
        g.drawString("Text in ORANGE, Size 30", 100, 250);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new DifferentText());
    }
}
```

Program 2: Displaying Different Lines

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

```

```
public class DifferentLines extends JFrame {
    private final int WIDTH = 800;
    private final int HEIGHT = 600;

    public DifferentLines() {
        setSize(WIDTH, HEIGHT);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        drawLines(g);
    }

    private void drawLines(Graphics g) {
        // Line 1: Color RED, Thickness 1
        g.setColor(Color.RED);
        g.drawLine(100, 100, 300, 100);

        // Line 2: Color BLUE, Thickness 2
        g.setColor(Color.BLUE);
        Graphics2D g2d = (Graphics2D) g;
        g2d.setStroke(new BasicStroke(2));
        g2d.drawLine(100, 150, 300, 150);

        // Line 3: Color GREEN, Thickness 3
        g.setColor(Color.GREEN);
        g2d.setStroke(new BasicStroke(3));
        g2d.drawLine(100, 200, 300, 200);

        // Line 4: Color ORANGE, Thickness 4
        g.setColor(Color.ORANGE);
        g2d.setStroke(new BasicStroke(4));
        g2d.drawLine(100, 250, 300, 250);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new DifferentLines());
    }
}
```

Program 3: Displaying Different Shapes

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

public class DifferentShapes extends JFrame {
    private final int WIDTH = 800;
    private final int HEIGHT = 600;

    public DifferentShapes() {
```

```
        setSize(WIDTH, HEIGHT);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        drawShapes(g);
    }

    private void drawShapes(Graphics g) {
        // Rectangle: Color RED
        g.setColor(Color.RED);
        g.fillRect(50, 50, 100, 80);

        // Circle: Color BLUE
        g.setColor(Color.BLUE);
        g.fillOval(200, 50, 100, 100);

        // Oval: Color GREEN
        g.setColor(Color.GREEN);
        g.fillOval(350, 50, 150, 80);

        // Polygon: Color ORANGE
        g.setColor(Color.ORANGE);
        int[] xPoints = {550, 600, 650};
        int[] yPoints = {50, 120, 50};
        g.fillPolygon(xPoints, yPoints, 3);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new DifferentShapes());
    }
}
```

Program 4: Drawing Arc

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

public class ArcDrawingExample extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        drawArcs(g);
    }

    private void drawArcs(Graphics g) {
        int width = getWidth();
        int height = getHeight();

```

```
        int xCenter = width / 2;
        int yCenter = height / 2;
        int arcRadius = Math.min(width, height) / 4; // Arc radius as 1/4th of the
            smaller dimension

        int startAngle = 45; // Starting angle for the arcs
        int arcAngle = 90; // Total angle of the arc (in degrees)

        // Draw a simple arc
        g.drawArc(xCenter - arcRadius, yCenter - arcRadius, arcRadius * 2,
            arcRadius * 2, startAngle, arcAngle);

        // Draw an arc with a larger angle
        g.drawArc(xCenter - arcRadius, yCenter - arcRadius, arcRadius * 2,
            arcRadius * 2, 120, 180);

        // Draw a filled arc
        g.fillArc(xCenter - arcRadius, yCenter - arcRadius, arcRadius * 2,
            arcRadius * 2, 270, 135);
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Arc Drawing Example");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 400);
        frame.add(new ArcDrawingExample());
        frame.setVisible(true);
    }
}
```

Program 5: Drawing Different Arc

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

public class DrawArcs extends JFrame {

    public DrawArcs() {
        setTitle("Drawing Arcs in Java");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);

        // Draw a full circle (360 degrees) starting from 0 degrees
        g.setColor(Color.BLUE);
        g.drawArc(50, 50, 300, 300, 0, 360);

        // Draw a 90-degree arc starting from 45 degrees
```

```
        g.setColor(Color.RED);
        g.drawArc(75, 75, 250, 250, 45, 90);

        // Draw a 180-degree arc starting from 180 degrees
        g.setColor(Color.GREEN);
        g.drawArc(100, 100, 200, 200, 180, 180);

        // Draw a 270-degree arc starting from 270 degrees
        g.setColor(Color.ORANGE);
        g.drawArc(125, 125, 150, 150, 270, 270);
    }

    public static void main(String[] args) {
        new DrawArcs();
    }
}
```

Program 5: Filling Different Arc

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

public class DrawFilledArcs extends JFrame {

    public DrawFilledArcs() {
        setTitle("Drawing Filled Arcs in Java");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);

        // Draw a full circle (360 degrees) starting from 0 degrees
        g.setColor(Color.BLUE);
        g.fillArc(50, 50, 300, 300, 0, 360);

        // Draw a 90-degree arc starting from 45 degrees
        g.setColor(Color.RED);
        g.fillArc(75, 75, 250, 250, 45, 90);

        // Draw a 180-degree arc starting from 180 degrees
        g.setColor(Color.GREEN);
        g.fillArc(100, 100, 200, 200, 180, 180);

        // Draw a 270-degree arc starting from 270 degrees
        g.setColor(Color.ORANGE);
        g.fillArc(125, 125, 150, 150, 270, 270);
    }

    public static void main(String[] args) {
```

```
        new DrawFilledArcs();
    }
}
```

Program 6: Drawing Different Polygons

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

public class DrawPolygons extends JFrame {

    public DrawPolygons() {
        setTitle("Drawing Polygons in Java");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);

        // Draw a triangle
        int[] xPointsTriangle = {100, 200, 150};
        int[] yPointsTriangle = {200, 200, 100};
        g.setColor(Color.BLUE);
        g.drawPolygon(xPointsTriangle, yPointsTriangle, 3);

        // Draw a rectangle
        int[] xPointsRectangle = {100, 300, 300, 100};
        int[] yPointsRectangle = {250, 250, 350, 350};
        g.setColor(Color.RED);
        g.drawPolygon(xPointsRectangle, yPointsRectangle, 4);

        // Draw a pentagon
        int[] xPointsPentagon = {200, 250, 300, 250, 200};
        int[] yPointsPentagon = {100, 50, 100, 150, 150};
        g.setColor(Color.GREEN);
        g.drawPolygon(xPointsPentagon, yPointsPentagon, 5);

        // Draw a hexagon
        int[] xPointsHexagon = {150, 200, 250, 250, 200, 150};
        int[] yPointsHexagon = {300, 300, 250, 200, 200, 250};
        g.setColor(Color.ORANGE);
        g.drawPolygon(xPointsHexagon, yPointsHexagon, 6);
    }

    public static void main(String[] args) {
        new DrawPolygons();
    }
}
```

Program 7: DDA Alogrithm

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

public class DDAAlgorithm extends JFrame {
    private final int WIDTH = 800;
    private final int HEIGHT = 600;

    public DDAAlgorithm() {
    setTitle("DDA Algorithm in Java");
        setSize(WIDTH, HEIGHT);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        ddaAlgorithm(g, 0, 0, 500, 400); // Replace these coordinates with your
            desired endpoints
    }

    private void ddaAlgorithm(Graphics g, int x1, int y1, int x2, int y2) {
        int dx = x2 - x1;
        int dy = y2 - y1;
        int steps = Math.max(Math.abs(dx), Math.abs(dy));

        float xIncrement = (float) dx / steps;
        float yIncrement = (float) dy / steps;

        float x = x1;
        float y = y1;

        for (int i = 0; i <= steps; i++) {
            int roundX = Math.round(x);
            int roundY = Math.round(y);
            g.drawLine(roundX, roundY, roundX, roundY);
            x += xIncrement;
            y += yIncrement;
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new DDAAlgorithm());
    }
}
```

Program 8: Bresenham Line Drawing Algorithm

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

public class BresenhamAlgorithm extends JFrame {
```

```
    private final int WIDTH = 800;
    private final int HEIGHT = 600;

    public BresenhamAlgorithm() {
    setTitle("Bresenham Line Drawing Algorithm in Java");
        setSize(WIDTH, HEIGHT);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        bresenhamAlgorithm(g, 100, 100, 500, 400); // Replace these coordinates
            with your desired endpoints
    }

    private void bresenhamAlgorithm(Graphics g, int x1, int y1, int x2, int y2) {
        int dx = Math.abs(x2 - x1);
        int dy = Math.abs(y2 - y1);
        int signX = x1 < x2 ? 1 : -1;
        int signY = y1 < y2 ? 1 : -1;

        int x = x1;
        int y = y1;

        boolean interchange = false;
        if (dy > dx) {
            // Swap dx and dy to ensure we are always iterating over the major axis
            int temp = dx;
            dx = dy;
            dy = temp;
            interchange = true;
        }

        int decision = 2 * dy - dx;

        for (int i = 0; i <= dx; i++) {
            // Draw the pixel at (x, y)
            g.drawLine(x, y, x, y);

            while (decision >= 0) {
                if (interchange)
                    x += signX;
                else
                    y += signY;

                decision -= 2 * dx;
            }

            if (interchange)
                y += signY;
```

```
            else
                x += signX;

            decision += 2 * dy;
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new BresenhamAlgorithm());
    }
}
```

Program 8: Midpoint Circle Drawing Algorithm

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

public class MidpointCircleAlgorithm extends JFrame {
    private final int WIDTH = 800;
    private final int HEIGHT = 600;

    public MidpointCircleAlgorithm() {
    setTitle("Midpoint Circle Drawing Algorithm in Java");
        setSize(WIDTH, HEIGHT);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        int centerX = WIDTH / 2;   // Replace with desired center X-coordinate
        int centerY = HEIGHT / 2;  // Replace with desired center Y-coordinate

        int radius = 100; // Change the radius to draw a different circle


    g.setColor(Color.RED);
    g.drawLine(WIDTH/2, 0, WIDTH/2, HEIGHT);
    g.drawLine(0, HEIGHT/2, WIDTH, HEIGHT/2 );

        midpointCircleAlgorithm(g, centerX, centerY, radius);
    }

    private void plotCirclePixel(Graphics g, int x, int y, int centerX, int
        centerY) {
        // Draw the pixels at the corresponding positions
    g.setColor(Color.BLACK);
        g.drawLine(centerX + x, centerY + y, centerX + x, centerY + y);
        g.drawLine(centerX - x, centerY + y, centerX - x, centerY + y);
        g.drawLine(centerX + x, centerY - y, centerX + x, centerY - y);
        g.drawLine(centerX - x, centerY - y, centerX - x, centerY - y);
        g.drawLine(centerX + y, centerY + x, centerX + y, centerY + x);
```

```
        g.drawLine(centerX - y, centerY + x, centerX - y, centerY + x);
        g.drawLine(centerX + y, centerY - x, centerX + y, centerY - x);
        g.drawLine(centerX - y, centerY - x, centerX - y, centerY - x);
    }

    private void midpointCircleAlgorithm(Graphics g, int centerX, int centerY, int
        radius) {
        int x = 0;
        int y = radius;
        int decision = 1 - radius;

        plotCirclePixel(g, x, y, centerX, centerY);

        while (y > x) {
            if (decision < 0) {
                decision += 2 * x + 1;
            } else {
                decision += 2 * (x - y) + 1;
                y--;
            }
            x++;
            plotCirclePixel(g, x, y, centerX, centerY);
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new MidpointCircleAlgorithm());
    }
}
```

Program 9: 2D Translation Algorithm Example

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

public class Translation2D extends JPanel {
    private void drawTranslatedShape(Graphics2D g2d, int[] xPoints, int[] yPoints,
        int dx, int dy) {
        int nPoints = xPoints.length;

        // Translate the shape's coordinates by dx and dy
        int[] translatedXPoints = new int[nPoints];
        int[] translatedYPoints = new int[nPoints];
        for (int i = 0; i < nPoints; i++) {
            translatedXPoints[i] = xPoints[i] + dx;
            translatedYPoints[i] = yPoints[i] + dy;
        }

        // Draw the original shape
        g2d.setColor(Color.BLACK);
        g2d.drawPolygon(xPoints, yPoints, nPoints);

        // Draw the translated shape in red
        g2d.setColor(Color.RED);
```

```
        g2d.drawPolygon(translatedXPoints, translatedYPoints, nPoints);
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        Graphics2D g2d = (Graphics2D) g;

        int[] xPoints = {100, 150, 200};
        int[] yPoints = {200, 100, 200};

        // Call drawTranslatedShape to draw the translated shape with dx and dy
        drawTranslatedShape(g2d, xPoints, yPoints, 50, 50);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            JFrame frame = new JFrame("2D Translation");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setSize(400, 400);
            frame.setContentPane(new Translation2D());
            frame.setVisible(true);
        });
    }
}
```

Program 10: 2D Translation Using Inbuilt Java Affine Transformation Example

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

public class TranslationExample extends JFrame {
    private final int WIDTH = 800;
    private final int HEIGHT = 600;

    public TranslationExample() {
    setTitle("Affine Transformation in Java")
        setSize(WIDTH, HEIGHT);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        Graphics2D g2d = (Graphics2D) g;

        int x = 100;
        int y = 100;
        int dx = 50;
        int dy = 100;
```

```
26
27          // Original rectangle
28          g2d.setColor(Color.RED);
29          g2d.fillRect(x, y, 100, 80);
30
31          // Translated rectangle
32          AffineTransform translation = AffineTransform.getTranslateInstance(dx, dy);
33          g2d.setColor(Color.BLUE);
34          g2d.fill(translation.createTransformedShape(new Rectangle(x, y, 100, 80)));
35      }
36
37      public static void main(String[] args) {
38          SwingUtilities.invokeLater(() -> new TranslationExample());
39      }
40  }
```

Program 10: 2D Rotation Example

```
1   import javax.swing.*;
2   import java.awt.*;
3
4   public class Rotation2DUsingFormula extends JPanel {
5       private void drawRotatedShape(Graphics2D g2d, int[] xPoints, int[] yPoints,
            double angleDegrees, int centerX, int centerY) {
6           int nPoints = xPoints.length;
7
8           // Convert the angle to radians
9           double angleRadians = Math.toRadians(angleDegrees);
10
11          // Apply the rotation formula to each point
12          int[] rotatedXPoints = new int[nPoints];
13          int[] rotatedYPoints = new int[nPoints];
14          for (int i = 0; i < nPoints; i++) {
15              int x = xPoints[i];
16              int y = yPoints[i];
17              rotatedXPoints[i] = (int) (centerX + (x - centerX) *
                    Math.cos(angleRadians) - (y - centerY) * Math.sin(angleRadians));
18              rotatedYPoints[i] = (int) (centerY + (x - centerX) *
                    Math.sin(angleRadians) + (y - centerY) * Math.cos(angleRadians));
19          }
20
21          // Draw the original shape
22          g2d.setColor(Color.BLACK);
23          g2d.drawPolygon(xPoints, yPoints, nPoints);
24
25          // Draw the rotated shape in red
26          g2d.setColor(Color.RED);
27          g2d.drawPolygon(rotatedXPoints, rotatedYPoints, nPoints);
28      }
29
30      @Override
31      protected void paintComponent(Graphics g) {
32          super.paintComponent(g);
33
```

```
        Graphics2D g2d = (Graphics2D) g;

        int[] xPoints = {100, 150, 200};
        int[] yPoints = {200, 100, 200};
        double angleDegrees = 45; // Rotation angle in degrees
        int centerX = 150; // Center of rotation x-coordinate
        int centerY = 150; // Center of rotation y-coordinate

        // Call drawRotatedShape to draw the rotated shape
        drawRotatedShape(g2d, xPoints, yPoints, angleDegrees, centerX, centerY);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            JFrame frame = new JFrame("2D Rotation using Formula");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setSize(400, 400);
            frame.setContentPane(new Rotation2DUsingFormula());
            frame.setVisible(true);
        });
    }
}
```

Program 11: 2D Scaling About an Origin

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

public class Scaling2DUsingFormula extends JPanel {
    private void drawScaledShape(Graphics2D g2d, int[] xPoints, int[] yPoints,
        double Sx, double Sy) {
        int nPoints = xPoints.length;



        // Apply the rotation formula to each point
        int[] scaledXPoints = new int[nPoints];
        int[] scaledYPoints = new int[nPoints];
        for (int i = 0; i < nPoints; i++) {
            int x = xPoints[i];
            int y = yPoints[i];
            scaledXPoints[i] = (int) (x * Sx);
            scaledYPoints[i] = (int) (y * Sy);
        }

        // Draw the original shape
        g2d.setColor(Color.BLACK);
        g2d.drawPolygon(xPoints, yPoints, nPoints);

        // Draw the rotated shape in red
        g2d.setColor(Color.RED);
        g2d.drawPolygon(scaledXPoints, scaledYPoints, nPoints);
    }

```

```
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        Graphics2D g2d = (Graphics2D) g;

        int[] xPoints = {100, 150, 200};
        int[] yPoints = {200, 100, 200};

        double Sx = 1.5; // Scaling factor in x direction
        double Sy = 1.5; // Scaling factor in y direction

        // Call drawRotatedShape to draw the rotated shape
        drawScaledShape(g2d, xPoints, yPoints, Sx, Sy);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            JFrame frame = new JFrame("2D Scaling about an origin using Formula");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setSize(800, 600);
            frame.setContentPane(new Scaling2DUsingFormula());
            frame.setVisible(true);
        });
    }
}
```

Program 12: 2D Scaling About an a fixed point

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

public class Scaling2DFixedPoint extends JPanel {
    private void drawScaledShape(Graphics2D g2d, int[] xPoints, int[] yPoints,
        double Sx, double Sy, int xf, int yf) {
        int nPoints = xPoints.length;




        // Apply the rotation formula to each point
        int[] scaledXPoints = new int[nPoints];
        int[] scaledYPoints = new int[nPoints];
        for (int i = 0; i < nPoints; i++) {
            int x = xPoints[i];
            int y = yPoints[i];
            scaledXPoints[i] = (int) (xf + (x - xf) * Sx);
            scaledYPoints[i] = (int) (yf + (y - yf) * Sy);
        }

        // Draw the original shape
        g2d.setColor(Color.BLACK);
        g2d.drawPolygon(xPoints, yPoints, nPoints);

        // Draw the rotated shape in red
```

```
        g2d.setColor(Color.RED);
        g2d.drawPolygon(scaledXPoints, scaledYPoints, nPoints);
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        Graphics2D g2d = (Graphics2D) g;

        int[] xPoints = {100, 150, 200};
        int[] yPoints = {200, 100, 200};

        double Sx = 1.5; // Scaling factor in x direction
        double Sy = 1.5; // Scaling factor in y direction

    int xf = 150;  //fixed point reference
    int yf = 150; //fixed point reference

        // Call drawRotatedShape to draw the rotated shape
        drawScaledShape(g2d, xPoints, yPoints, Sx, Sy, xf, yf);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            JFrame frame = new JFrame("2D Scaling about an origin using Formula");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setSize(800, 600);
            frame.setContentPane(new Scaling2DFixedPoint());
            frame.setVisible(true);
        });
    }
}
```

Program 13: X-direction Shear

```
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.*;
import javax.swing.*;

public class XShearingRectangle extends JPanel {

  private final int WIDTH = 800;
    private final int HEIGHT = 600;

    private int[] originalX = {50, 250, 250, 50}; // x-coordinates of the original
        rectangle
    private int[] originalY = {50, 50, 150, 150}; // y-coordinates of the original
        rectangle
    private int shearingFactorX = 2; // Shearing factor along X-axis

    @Override
```

```
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

    g.setColor(Color.RED);
    g.drawLine(WIDTH/2, 0, WIDTH/2, HEIGHT);
    g.drawLine(0, HEIGHT/2, WIDTH, HEIGHT/2 );

        // Draw the original rectangle
    g.setColor(Color.BLACK);
        g.drawPolygon(originalX, originalY, 4);

        // Shearing the rectangle along X-axis
        int[] shearedX = new int[4];
        int[] shearedY = new int[4];
        for (int i = 0; i < 4; i++) {
            shearedX[i] = originalX[i] + shearingFactorX * originalY[i];
            shearedY[i] = originalY[i];
        }

        // Draw the sheared rectangle
    g.setColor(Color.BLUE);
        g.drawPolygon(shearedX, shearedY, 4);
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Shearing Rectangle");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 600);
        frame.add(new XShearingRectangle());
        frame.setVisible(true);
    }
}
```

Program 14: Y-direction Shear

```
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.*;
import javax.swing.*;

public class YShearingRectangle extends JPanel {

  private final int WIDTH = 800;
    private final int HEIGHT = 600;

    private int[] originalX = {50, 250, 250, 50}; // x-coordinates of the original
        rectangle
    private int[] originalY = {50, 50, 150, 150}; // y-coordinates of the original
        rectangle
    private int shearingFactorY = 2; // Shearing factor along Y-axis

    @Override
    protected void paintComponent(Graphics g) {
```

```
        super.paintComponent(g);

    g.setColor(Color.RED);
    g.drawLine(WIDTH/2, 0, WIDTH/2, HEIGHT);
    g.drawLine(0, HEIGHT/2, WIDTH, HEIGHT/2 );

        // Draw the original rectangle
    g.setColor(Color.BLACK);
        g.drawPolygon(originalX, originalY, 4);

        // Shearing the rectangle along X-axis
        int[] shearedX = new int[4];
        int[] shearedY = new int[4];
        for (int i = 0; i < 4; i++) {
            shearedX[i] = originalX[i] ;
            shearedY[i] = originalX[i] * shearingFactorY + originalY[i];
        }

        // Draw the sheared rectangle
    g.setColor(Color.BLUE);
        g.drawPolygon(shearedX, shearedY, 4);
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Y-direction Shearing");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 600);
        frame.add(new YShearingRectangle());
        frame.setVisible(true);
    }
}
```

Program 15: Reflection about x-axis

```
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.*;

public class ReflectTriangleXAxis extends JPanel {

  private final int WIDTH = 800;
    private final int HEIGHT = 600;

    private int[] originalX = {100, 200, 150}; // x-coordinates of the original
        triangle
    private int[] originalY = {100, 100, 50}; // y-coordinates of the original
        triangle

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
    g.setColor(Color.RED);
    g.drawLine(WIDTH/2, 0, WIDTH/2, HEIGHT);
```

```
    g.drawLine(0, HEIGHT/2, WIDTH, HEIGHT/2 );

        // Draw the original triangle
    g.setColor(Color.BLACK);
        g.drawPolygon(originalX, originalY, 3);

        // Reflecting the triangle along X-axis
        int[] reflectedX = new int[3];
        int[] reflectedY = new int[3];
        for (int i = 0; i < 3; i++) {
            reflectedX[i] = originalX[i];
            reflectedY[i] = getHeight() - originalY[i];
        }

        // Draw the reflected triangle
    g.setColor(Color.BLUE);
        g.drawPolygon(reflectedX, reflectedY, 3);
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Reflect Triangle about X-axis");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 600);
        frame.add(new ReflectTriangleXAxis());
        frame.setVisible(true);
    }
}
```

Program 16: Reflection about y-axis

```
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.*;

public class ReflectTriangleYAxis extends JPanel {

  private final int WIDTH = 800;
    private final int HEIGHT = 600;

    private int[] originalX = {100, 200, 150}; // x-coordinates of the original
        triangle
    private int[] originalY = {100, 100, 50}; // y-coordinates of the original
        triangle

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
    g.setColor(Color.RED);
    g.drawLine(WIDTH/2, 0, WIDTH/2, HEIGHT);
    g.drawLine(0, HEIGHT/2, WIDTH, HEIGHT/2 );

        // Draw the original triangle
    g.setColor(Color.BLACK);
```

```
        g.drawPolygon(originalX, originalY, 3);

        // Reflecting the triangle along X-axis
        int[] reflectedX = new int[3];
        int[] reflectedY = new int[3];
        for (int i = 0; i < 3; i++) {
            reflectedX[i] = getWidth() - originalX[i];
            reflectedY[i] = originalY[i];
        }

        // Draw the reflected triangle
    g.setColor(Color.BLUE);
        g.drawPolygon(reflectedX, reflectedY, 3);
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Reflect Triangle about Y-axis");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 600);
        frame.add(new ReflectTriangleYAxis());
        frame.setVisible(true);
    }
}
```

Program 17: Drawing a Triangle with center as origin

```
import java.awt.Dimension;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.*;

public class TriangleWithCenterOrigin extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        // Get the dimensions of the display
        Dimension screenSize = getSize();
        int centerX = screenSize.width / 2;
        int centerY = screenSize.height / 2;

    int WIDTH = screenSize.width;
    int HEIGHT = screenSize.height;

    g.setColor(Color.RED);
    g.drawLine(WIDTH/2, 0, WIDTH/2, HEIGHT);
    g.drawLine(0, HEIGHT/2, WIDTH, HEIGHT/2 );

        // Define the vertices of the triangle with respect to the center
        int[] verticesX = {centerX, centerX + 100, centerX - 100};
        int[] verticesY = {centerY - 100, centerY + 50, centerY + 50};

```

```
        // Draw the triangle
    g.setColor(Color.BLACK);
        g.drawPolygon(verticesX, verticesY, 3);
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Triangle with Center Origin");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 400);
        frame.add(new TriangleWithCenterOrigin());
        frame.setVisible(true);
    }
}
```

Program 18: Reflection of a Triangle about x-axis(y=0)

```
import java.awt.Dimension;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.*;

public class ReflectionTriangleXaxis extends JPanel {
  private int[] originalX = {50, 150, 100}; // x-coordinates of the original
      triangle
    private int[] originalY = {100, 100, 50}; // y-coordinates of the original
        triangle

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        // Get the dimensions of the display
        Dimension screenSize = getSize();
        int centerX = screenSize.width / 2;
        int centerY = screenSize.height / 2;

    int WIDTH = screenSize.width;
    int HEIGHT = screenSize.height;

    g.setColor(Color.RED);
    g.drawLine(WIDTH/2, 0, WIDTH/2, HEIGHT);
    g.drawLine(0, HEIGHT/2, WIDTH, HEIGHT/2 );

    // Draw the original triangle with respect to the center
    g.setColor(Color.BLACK);
        drawTriangle(g, originalX, originalY, centerX, centerY);



    // Reflecting the triangle along X-axis
        int[] reflectedX = new int[3];
        int[] reflectedY = new int[3];
        for (int i = 0; i < 3; i++) {
```

```
            reflectedX[i] = originalX[i];
            reflectedY[i] = - originalY[i];
        }

        // Draw the reflected triangle
    g.setColor(Color.BLUE);
    drawTriangle(g, reflectedX, reflectedY, centerX, centerY);

    }

  private void drawTriangle(Graphics g, int[] x, int[] y, int centerX, int centerY)
      {
        // Shift the triangle to draw it with respect to the center
        int[] shiftedX = new int[3];
        int[] shiftedY = new int[3];
        for (int i = 0; i < 3; i++) {
            shiftedX[i] = centerX + x[i];
            shiftedY[i] = centerY - y[i];
        }

        // Draw the triangle
        g.drawPolygon(shiftedX, shiftedY, 3);
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Reflection of a triangle about x-axis");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 400);
        frame.add(new ReflectionTriangleXaxis());
        frame.setVisible(true);
    }
}
```

Program 19: Reflection of a Triangle about y-axis(x=0)

```
import java.awt.Dimension;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.*;

public class ReflectionTriangleYaxis extends JPanel {
  private int[] originalX = {50, 150, 100}; // x-coordinates of the original
      triangle
    private int[] originalY = {100, 100, 50}; // y-coordinates of the original
        triangle

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        // Get the dimensions of the display
        Dimension screenSize = getSize();
        int centerX = screenSize.width / 2;
```

```
        int centerY = screenSize.height / 2;

    int WIDTH = screenSize.width;
    int HEIGHT = screenSize.height;

    g.setColor(Color.RED);
    g.drawLine(WIDTH/2, 0, WIDTH/2, HEIGHT);
    g.drawLine(0, HEIGHT/2, WIDTH, HEIGHT/2 );

    // Draw the original triangle with respect to the center
    g.setColor(Color.BLACK);
        drawTriangle(g, originalX, originalY, centerX, centerY);



    // Reflecting the triangle along Y-axis
        int[] reflectedX = new int[3];
        int[] reflectedY = new int[3];
        for (int i = 0; i < 3; i++) {
            reflectedX[i] = - originalX[i];
            reflectedY[i] = originalY[i];
        }

        // Draw the reflected triangle
    g.setColor(Color.BLUE);
    drawTriangle(g, reflectedX, reflectedY, centerX, centerY);

    }

  private void drawTriangle(Graphics g, int[] x, int[] y, int centerX, int centerY)
      {
        // Shift the triangle to draw it with respect to the center
        int[] shiftedX = new int[3];
        int[] shiftedY = new int[3];
        for (int i = 0; i < 3; i++) {
            shiftedX[i] = centerX + x[i];
            shiftedY[i] = centerY - y[i];
        }

        // Draw the triangle
        g.drawPolygon(shiftedX, shiftedY, 3);
  }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Reflection of a triangle about y-axis");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 600);
        frame.add(new ReflectionTriangleYaxis());
        frame.setVisible(true);
    }
}
```

Program 20: Reflection of a Triangle about an origin

```java
import java.awt.Dimension;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.*;

public class ReflectionTriangleAboutOrigin extends JPanel {
  private int[] originalX = {50, 150, 100}; // x-coordinates of the original triangle
    private int[] originalY = {100, 100, 50}; // y-coordinates of the original triangle

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        // Get the dimensions of the display
        Dimension screenSize = getSize();
        int centerX = screenSize.width / 2;
        int centerY = screenSize.height / 2;

    int WIDTH = screenSize.width;
    int HEIGHT = screenSize.height;

    g.setColor(Color.RED);
    g.drawLine(WIDTH/2, 0, WIDTH/2, HEIGHT);
    g.drawLine(0, HEIGHT/2, WIDTH, HEIGHT/2 );

    // Draw the original triangle with respect to the center
    g.setColor(Color.BLACK);
        drawTriangle(g, originalX, originalY, centerX, centerY);

    // Reflecting the triangle about an origin
        int[] reflectedX = new int[3];
        int[] reflectedY = new int[3];
        for (int i = 0; i < 3; i++) {
            reflectedX[i] = - originalX[i];
            reflectedY[i] = - originalY[i];
        }

        // Draw the reflected triangle
    g.setColor(Color.BLUE);
    drawTriangle(g, reflectedX, reflectedY, centerX, centerY);
    }

  private void drawTriangle(Graphics g, int[] x, int[] y, int centerX, int centerY) {
      // Shift the triangle to draw it with respect to the center
      int[] shiftedX = new int[3];
      int[] shiftedY = new int[3];
      for (int i = 0; i < 3; i++) {
          shiftedX[i] = centerX + x[i];
          shiftedY[i] = centerY - y[i];
```

```
        }

        // Draw the triangle
        g.drawPolygon(shiftedX, shiftedY, 3);
    }

       public static void main(String[] args) {
        JFrame frame = new JFrame("Reflection of a triangle about about an origin");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 400);
        frame.add(new ReflectionTriangleAboutOrigin());
        frame.setVisible(true);
    }
}
```

Program 21: Reflection of a Triangle with respect to y=x

```
import java.awt.Dimension;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.*;

public class ReflectionTriangleYequalX extends JPanel {
  private int[] originalX = {50, 120, 50}; // x-coordinates of the original triangle
    private int[] originalY = {150, 150, 200}; // y-coordinates of the original
        triangle

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        // Get the dimensions of the display
        Dimension screenSize = getSize();
        int centerX = screenSize.width / 2;
        int centerY = screenSize.height / 2;

    int WIDTH = screenSize.width;
    int HEIGHT = screenSize.height;

    g.setColor(Color.RED);
    g.drawLine(WIDTH/2, 0, WIDTH/2, HEIGHT);
    g.drawLine(0, HEIGHT/2, WIDTH, HEIGHT/2 );

    // Lie Color GREEN, Thickness 1
        g.setColor(Color.GREEN);
    Graphics2D g2d = (Graphics2D) g;
        g2d.setStroke(new BasicStroke(1));
        g2d.drawLine(WIDTH, 0, 0, HEIGHT);
    g.drawString("y = x", 650, 50);

    //draw an original polygon with screen coordinates
    g.setColor(Color.BLACK);
    g.drawPolygon(originalX, originalY,3);
```

```
    g.drawString("An original Rectangle", 50, 140);

    // Draw the original triangle with respect to the center
    g.setColor(Color.RED);
        drawTriangle(g, originalX, originalY, centerX, centerY);

    // Reflecting the triangle with respect to y = x
        int[] reflectedX = new int[3];
        int[] reflectedY = new int[3];
        for (int i = 0; i < 3; i++) {
            reflectedX[i] = originalY[i];
            reflectedY[i] = originalX[i];
        }

    // Draw the reflected triangle with respect to the center
    g.setColor(Color.BLUE);
        drawTriangle(g, reflectedX, reflectedY, centerX, centerY);

  }

    private void drawTriangle(Graphics g, int[] x, int[] y, int centerX, int
        centerY) {
        // Shift the triangle to draw it with respect to the center
        int[] shiftedX = new int[3];
        int[] shiftedY = new int[3];
        for (int i = 0; i < 3; i++) {
            shiftedX[i] = centerX + x[i];
            shiftedY[i] = centerY - y[i];
        }

        // Draw the triangle considering origin at center of a screen

        g.drawPolygon(shiftedX, shiftedY, 3);
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Reflection of a triangle w.r.t to y = x");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 600);
        frame.add(new ReflectionTriangleYequalX());
        frame.setVisible(true);
    }
}
```

Program 22: Reflection of a Triangle with respect to y = -x

```
import java.awt.Dimension;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.*;

public class ReflectionTriangleYEqualsNegativeX extends JPanel {
  private int[] originalX = {50, 150, 100}; // x-coordinates of the original
```

```
    triangle
    private int[] originalY = {100, 100, 50}; // y-coordinates of the original
        triangle

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        // Get the dimensions of the display
        Dimension screenSize = getSize();
        int centerX = screenSize.width / 2;
        int centerY = screenSize.height / 2;

    int WIDTH = screenSize.width;
    int HEIGHT = screenSize.height;

    g.setColor(Color.RED);
    g.drawLine(WIDTH/2, 0, WIDTH/2, HEIGHT);
    g.drawLine(0, HEIGHT/2, WIDTH, HEIGHT/2 );

    // Lie Color GREEN, Thickness 1
        g.setColor(Color.GREEN);
    Graphics2D g2d = (Graphics2D) g;
        g2d.setStroke(new BasicStroke(1));
        g2d.drawLine(0, 0, WIDTH, HEIGHT);
    g.drawString("y = -x", 180, 160);

    // Draw the original triangle with respect to the center
    g.setColor(Color.BLACK);
        drawTriangle(g, originalX, originalY, centerX, centerY);

    // Reflecting the triangle about an origin
        int[] reflectedX = new int[3];
        int[] reflectedY = new int[3];
        for (int i = 0; i < 3; i++) {
            reflectedX[i] = - originalY[i];
            reflectedY[i] = - originalX[i];
        }

        // Draw the reflected triangle
    g.setColor(Color.BLUE);
    drawTriangle(g, reflectedX, reflectedY, centerX, centerY);
    }

private void drawTriangle(Graphics g, int[] x, int[] y, int centerX, int centerY)
    {
        // Shift the triangle to draw it with respect to the center
        int[] shiftedX = new int[3];
        int[] shiftedY = new int[3];
        for (int i = 0; i < 3; i++) {
            shiftedX[i] = centerX + x[i];
            shiftedY[i] = centerY - y[i];
        }
```

```
59
60          // Draw the triangle
61          g.drawPolygon(shiftedX, shiftedY, 3);
62      }
63      public static void main(String[] args) {
64          JFrame frame = new JFrame("Reflection of a triangle w.r.t to y = - x");
65          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
66          frame.setSize(800, 600);
67          frame.add(new ReflectionTriangleYEqualsNegativeX());
68          frame.setVisible(true);
69      }
70  }
```

Program 23: Rotate a text

```
1  import java.awt.Font;
2  import java.awt.Color;
3  import java.awt.FontMetrics;
4  import java.awt.Graphics;
5  import java.awt.Graphics2D;
6  import javax.swing.JFrame;
7  import javax.swing.JPanel;
8
9  public class RotateTextInJava extends JPanel {
10
11     @Override
12     protected void paintComponent(Graphics g) {
13         super.paintComponent(g);
14
15         // Set the X and Y coordinates where you want to print the text
16         int xCoordinate = 100;
17         int yCoordinate = 150;
18
19         // Set the text to be printed
20         String text = "Hello, Java!";
21
22
23
24         // Set the rotation angle in degrees
25         double rotationAngleDegrees = 45;
26
27     // Create a new Font with the desired font family, style, and size
28         Font font = new Font("Arial", Font.BOLD, 26);
29
30         // Cast Graphics object to Graphics2D for additional functionality
31         Graphics2D g2d = (Graphics2D) g;
32
33     // Set the new Font
34         g2d.setFont(font);
35
36     //set color of text
37
38     g2d.setColor(Color.RED);
39
```

```
    // Set the rotation angle
        double rotationAngleRadians = Math.toRadians(rotationAngleDegrees);
        g2d.rotate(rotationAngleRadians, xCoordinate, yCoordinate);

        // Draw the text at the specified rotated coordinates
        g2d.drawString(text, xCoordinate, yCoordinate);
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame("Rotate Text in Java");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 400);
        frame.add(new RotateTextInJava());
        frame.setVisible(true);
    }
}
```