```java
// Rachel Burke
// Shearing.java: This program is Chapter 3 Homework's Problem 2

import java.awt.*;
import java.awt.event.*;
import java.awt.geom.AffineTransform;
import java.awt.geom.Ellipse2D;
import java.awt.geom.Rectangle2D;
import javax.swing.*;
import java.lang.Math;

public class Shearing {

    private JFrame mainFrame;
    private JPanel controlPanel;
    private CvShearing cvs;

    private int x = 200;
    private int y = 200;

    public Shearing() {
        prepareGUI();
    }

    public static void main(String[] args) {
        Shearing shearing = new Shearing();
        shearing.showContent();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Chapter 3: Homework 2 - Problem 2");
        mainFrame.setSize(475, 465);
        mainFrame.setResizable(false);
        mainFrame.setLayout(new FlowLayout());
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(controlPanel);
        mainFrame.setVisible(true);
    }

    private void showContent() {
        cvs = new CvShearing(x, y);
        controlPanel.add(cvs);
        mainFrame.setVisible(true);
    }
}

class CvShearing extends Canvas {
    private static final long serialVersionUID = 1L;
    int x, y;

    public CvShearing(int x, int y) {
        setBackground(Color.LIGHT_GRAY);
        setSize(401, 401);
        this.x = x;
        this.y = y;
    }

    int Transform(int p, int shift)
    {
        return p + shift;
    }

    int Shear(int p, int d, double a) {
        return p + (int) (a * d);
    }

    void DrawCircle(int x, int y, int r, int shiftto, int shiftback, double a, Graphics g) {
```

```
        for (double angle = 0; angle < 360; angle += 0.1) {
            double x1 = r * Math.cos(angle * Math.PI / 180);
            double y1 = r * Math.sin(angle * Math.PI / 180);
            int pX = Transform(Shear(Transform((int)(x + x1), -shiftto), (int) (y + y1), a),
            shiftback);
            int pY = Transform(Shear(Transform((int)(y + y1), -shiftto), 0, a), shiftto);
            g.drawLine(pX, pY, pX, pY);
        }
    }

    void DrawSquare(int x, int y, int s, int shift, double a, Graphics g) {
        int x1 = Transform(Shear(Transform(x, (int) 0), y, a), shift);
        int x2 = Transform(Shear(Transform(x + s, (int) 0), y, a), shift);
        int x3 = Transform(Shear(Transform(x + s, (int) 0), y + s, a), shift);
        int x4 = Transform(Shear(Transform(x, (int) 0), y + s, a), shift);
        g.drawLine(x1, y, x2, y); // Top -
        g.drawLine(x1, y, x4, y + s); // Left |
        g.drawLine(x2, y, x3, y + s); // Right |
        g.drawLine(x4, y + s, x3, y + s); // Bottom -
    }

    public void paint(Graphics g) {
        // Draw the before square and approximated circle
        DrawCircle(x, y, 100, 0, 0, 0, g);
        double s = (Math.sqrt(Math.PI) * 100);
        DrawSquare((int) (x - (s/2)), (int) (y - (s/2)), (int) s, 0, 0, g);

        // Shearing about middle and redrawing shapes
        g.setColor(Color.RED);
        DrawCircle(x, y, 100, 100, (100 - (int) (x/2)), 0.5, g);
        DrawSquare((int) (x - (s/2)), (int) (y - (s/2)), (int) s, -100, 0.5, g);
    }
}
```