E A Yoga Wilanda
1201222013

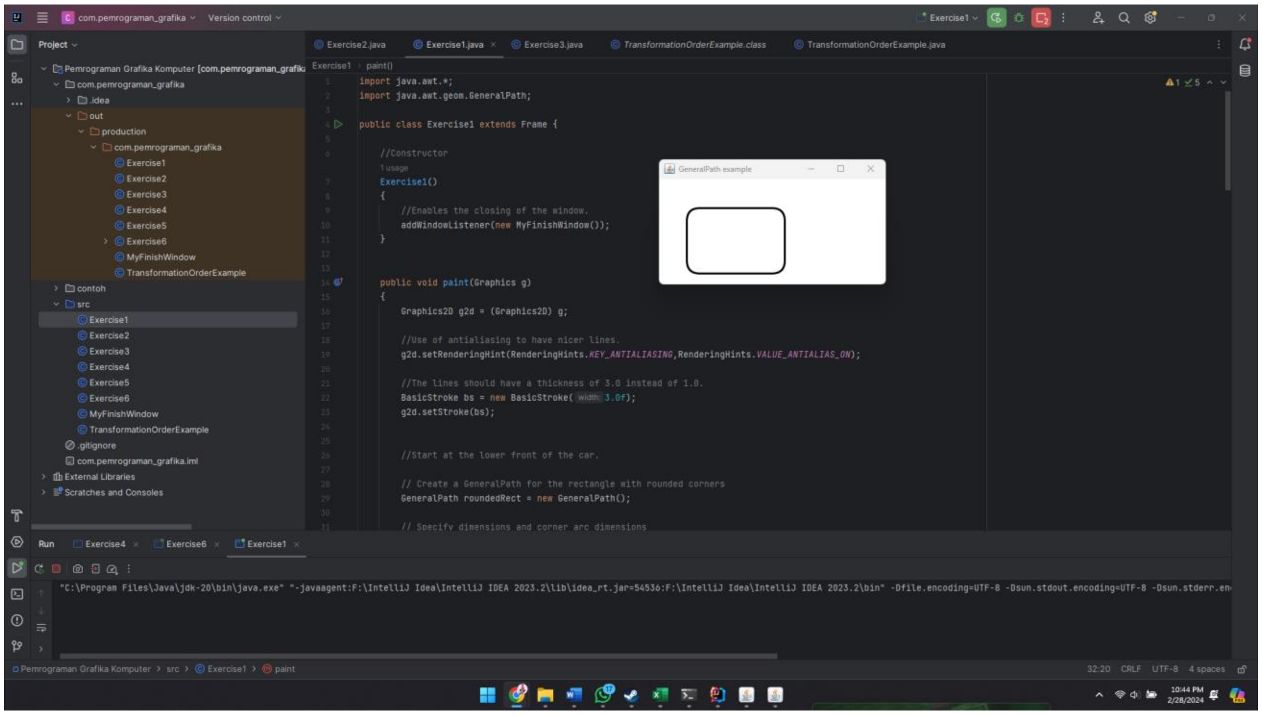
Rekayasa Perangkat Lunak / Semester IV

Pemrograman Grafika

Jawaban

Exercise 2.1

Hasil run:



```
import java.awt.*;
import java.awt.geom.GeneralPath;

public class Exercise1 extends Frame {

    //Constructor
    Exercise1()
    {
        //Enables the closing of the window.
        addWindowListener(new MyFinishWindow());
    }


    public void paint(Graphics g)
    {
        Graphics2D g2d = (Graphics2D) g;

        //Use of antialiasing to have nicer lines.
        g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,RenderingHints.VALUE_ANTIALIAS_ON);

        //The lines should have a thickness of 3.0 instead of 1.0.
        BasicStroke bs = new BasicStroke( width: 3.0f);
        g2d.setStroke(bs);


        //Start at the lower front of the car.

        // Create a GeneralPath for the rectangle with rounded corners
        GeneralPath roundedRect = new GeneralPath();

        // Specify dimensions and corner arc dimensions
```

```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:F:\IntelliJ Idea\IntelliJ IDEA 2023.2\lib\idea_rt.jar=54536:F:\IntelliJ Idea\IntelliJ IDEA 2023.2\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.en
```

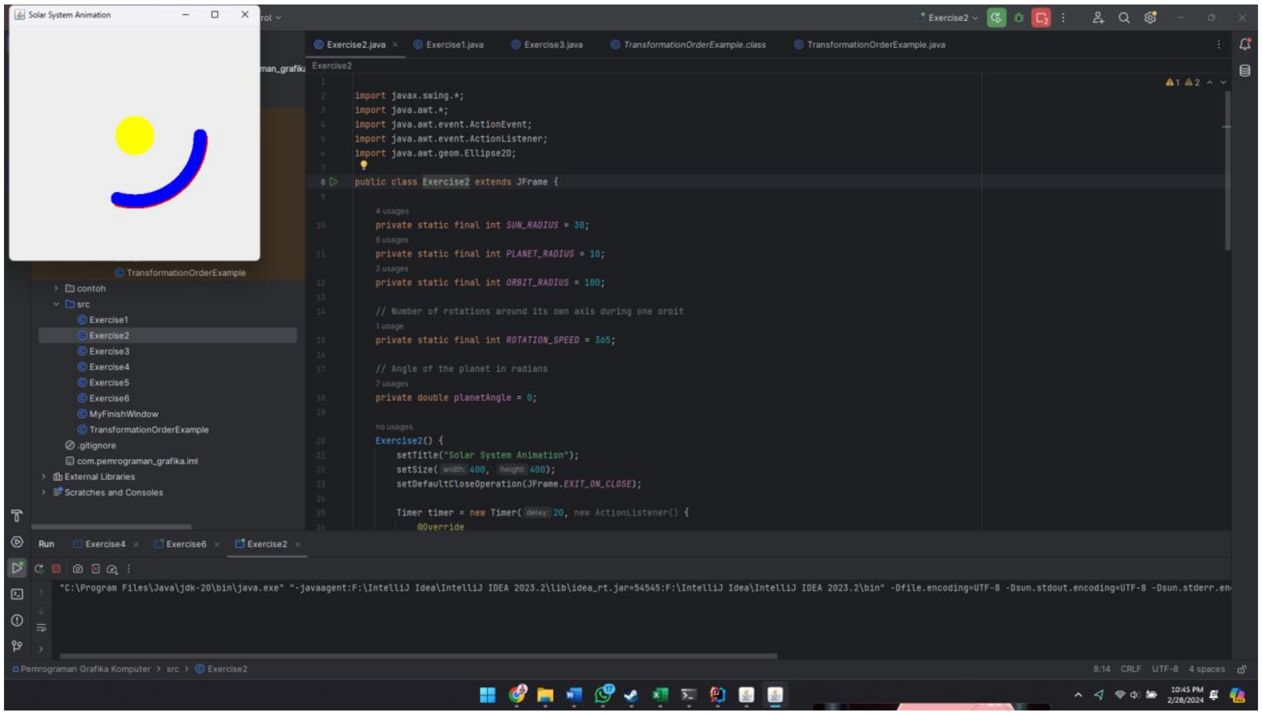
Kode:

```
Exercise 1.
import java.awt.*;
import java.awt.geom.GeneralPath;

public class Exercise1 extends Frame {

    //Constructor
    Exercise1()
    {
        //Enables the closing of the window.
        addWindowListener(new MyFinishWindow());
    }
    public void paint(Graphics g)
    {
        Graphics2D g2d = (Graphics2D) g;

        //Use of antialiasing to have nicer lines.

g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,RenderingHints.VALUE_ANTI
ALIAS_ON);

        //The lines should have a thickness of 3.0 instead of 1.0.
        BasicStroke bs = new BasicStroke(3.0f);
        g2d.setStroke(bs);


        //Start at the lower front of the car.

        // Create a GeneralPath for the rectangle with rounded corners
        GeneralPath roundedRect = new GeneralPath();

        // Specify dimensions and corner arc dimensions
        int x = 50;
        int y = 75;
        int width = 150;
        int height = 100;
        int arcWidth = 20;
        int arcHeight = 20;

        // Construct the rounded rectangle path
        // Begin at top-left corner
        roundedRect.moveTo(x + arcWidth, y);

        // Top edge
        roundedRect.lineTo(x + width - arcWidth, y);

        // Top-right arc
        roundedRect.quadTo(x + width, y, x + width, y + arcHeight);

        // Right edge
        roundedRect.lineTo(x + width, y + height - arcHeight);


        // Bottom-right arc
        roundedRect.quadTo(x + width, y + height, x + width - arcWidth, y +
height);


        // Bottom edge
        roundedRect.lineTo(x + arcWidth, y + height);


        // Bottom-left arc
        roundedRect.quadTo(x, y + height, x, y + height - arcHeight);

        // Left edge
        roundedRect.lineTo(x, y + arcHeight);

        // Top-left arc
```

```java
        roundedRect.quadTo(x, y, x + arcWidth, y);

        // Close the path
        roundedRect.closePath();

        // Draw the rounded rectangle using Graphics2D
        g2d.draw(roundedRect);


    }




    /**
     * Draws a coordinate system (according to the window coordinates).
     *
     * @param xmax     x-coordinate to which the x-axis should extend.
     * @param ymax     y-coordinate to which the y-axis should extend.
     * @param g2d      Graphics2D object for drawing.
     */
    public static void drawSimpleCoordinateSystem(int xmax, int ymax,
                                                  Graphics2D g2d)
    {
        int xOffset = 30;
        int yOffset = 50;
        int step = 20;
        String s;
        //Remember the actual font.
        Font fo = g2d.getFont();
        //Use a small font.
        g2d.setFont(new Font("sansserif",Font.PLAIN,9));
        //x-axis.
        g2d.drawLine(xOffset,yOffset,xmax,yOffset);
        //Marks and labels for the x-axis.
        for (int i=xOffset+step; i<=xmax; i=i+step)
        {
            g2d.drawLine(i,yOffset-2,i,yOffset+2);
            g2d.drawString(String.valueOf(i),i-7,yOffset-7);
        }

        //y-axis.
        g2d.drawLine(xOffset,yOffset,xOffset,ymax);

        //Marks and labels for the y-axis.
        s="  ";
        for (int i=yOffset+step; i<=ymax; i=i+step)
        {
            g2d.drawLine(xOffset-2,i,xOffset+2,i);
            if (i>99){s="";}
            g2d.drawString(s+String.valueOf(i),xOffset-25,i+5);
        }

        //Reset to the original font.
        g2d.setFont(fo);
    }
    public static void main(String[] argv)
    {
        Exercise1 f = new Exercise1();
        f.setTitle("GeneralPath example");
        f.setSize(250,200);
        f.setVisible(true);
    }
}
```

Exercise, 2.2

Hasil run:



```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.geom.Ellipse2D;

public class Exercise2 extends JFrame {

    private static final int SUN_RADIUS = 30;
    private static final int PLANET_RADIUS = 10;
    private static final int ORBIT_RADIUS = 100;

    // Number of rotations around its own axis during one orbit
    private static final int ROTATION_SPEED = 365;

    // Angle of the planet in radians
    private double planetAngle = 0;

    Exercise2() {
        setTitle("Solar System Animation");
        setSize( width: 400,  height: 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Timer timer = new Timer( delay: 20, new ActionListener() {
            @Override
```

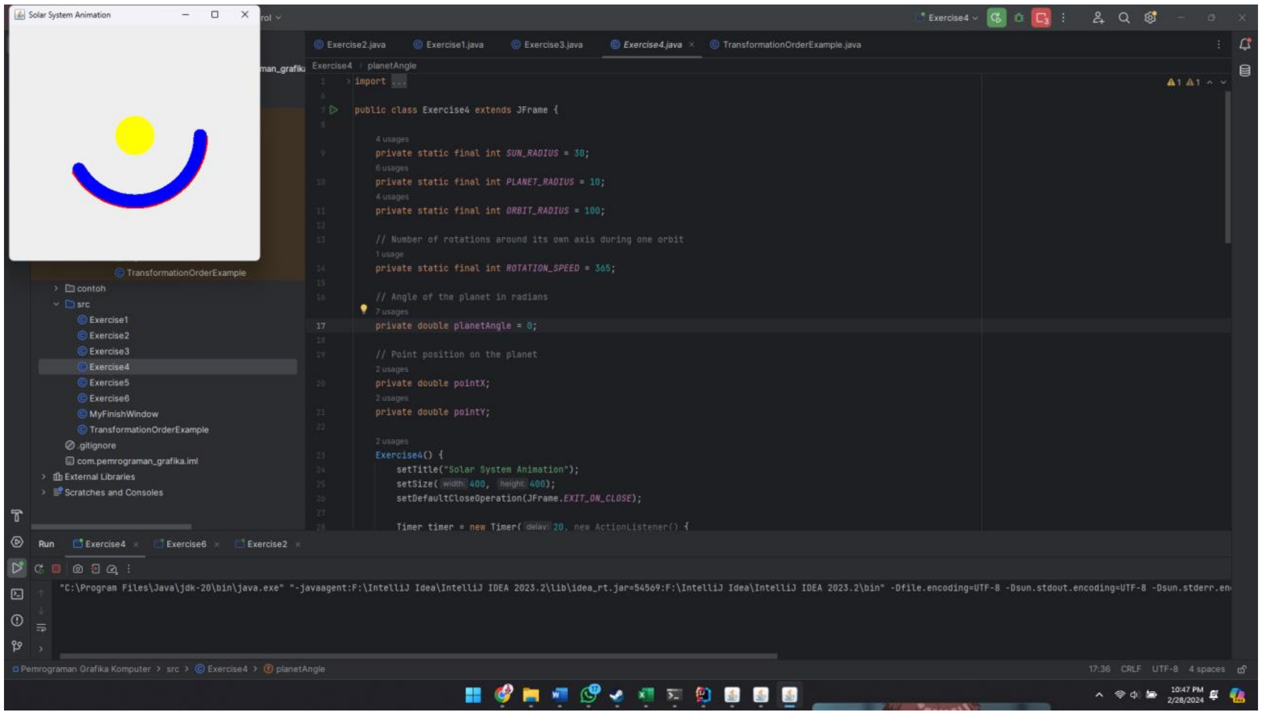
```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:F:\IntelliJ Idea\IntelliJ IDEA 2023.2\lib\idea_rt.jar=54545:F:\IntelliJ Idea\IntelliJ IDEA 2023.2\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.en
```

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.geom.Ellipse2D;

public class Exercise2 extends JFrame {

    private static final int SUN_RADIUS = 30;
    private static final int PLANET_RADIUS = 10;
    private static final int ORBIT_RADIUS = 100;

    // Number of rotations around its own axis during one orbit
    private static final int ROTATION_SPEED = 365;

    // Angle of the planet in radians
    private double planetAngle = 0;
    Exercise2() {
        setTitle("Solar System Animation");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Timer timer = new Timer(20, new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Update the planet angle for animation (3.14)
                planetAngle += 2 * Math.PI / ROTATION_SPEED;

                // Repaint the frame
                repaint();
            }
        });

        timer.start();
    }

    public void paint(Graphics g) {
        Graphics2D g2d = (Graphics2D) g;

        // Set the coordinate system to the center of the frame
        int centerX = getWidth() / 2;
        int centerY = getHeight() / 2;
        g2d.translate(centerX, centerY);

        // Draw the sun
        g2d.setColor(Color.YELLOW);
        g2d.fill(new Ellipse2D.Double(-SUN_RADIUS, -SUN_RADIUS, 2 *
SUN_RADIUS, 2 * SUN_RADIUS));

        // Calculate the position of the planet
        double planetX = ORBIT_RADIUS * Math.cos(planetAngle);
        double planetY = ORBIT_RADIUS * Math.sin(planetAngle);

        // Draw the planet
        g2d.setColor(Color.BLUE);
        g2d.fill(new Ellipse2D.Double(planetX - PLANET_RADIUS, planetY -
PLANET_RADIUS, 2 * PLANET_RADIUS, 2 * PLANET_RADIUS));

        // Describe the position of the point on the planet after one third of
its orbit
        if (planetAngle >= 2 * Math.PI / 3 && planetAngle <= 4 * Math.PI / 3)
{
            double pointX = planetX + PLANET_RADIUS * Math.cos(planetAngle);
            double pointY = planetY + PLANET_RADIUS * Math.sin(planetAngle);

            g2d.setColor(Color.RED);
            g2d.fill(new Ellipse2D.Double(pointX - 2, pointY - 2, 4, 4));
        }
    }

    public static void main(String[] argv) {
        SwingUtilities.invokeLater(() -> {
            Exercise4 ex = new Exercise4();
```

```
                ex.setVisible(true);
            });
        }
}
```

Exercise 2.24

Hasil run:



```
import ...

public class Exercise4 extends JFrame {

    private static final int SUN_RADIUS = 30;
    private static final int PLANET_RADIUS = 10;
    private static final int ORBIT_RADIUS = 100;

    // Number of rotations around its own axis during one orbit
    private static final int ROTATION_SPEED = 365;

    // Angle of the planet in radians
    private double planetAngle = 0;

    // Point position on the planet
    private double pointX;
    private double pointY;

    Exercise4() {
        setTitle("Solar System Animation");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Timer timer = new Timer(20, new ActionListener() {
```

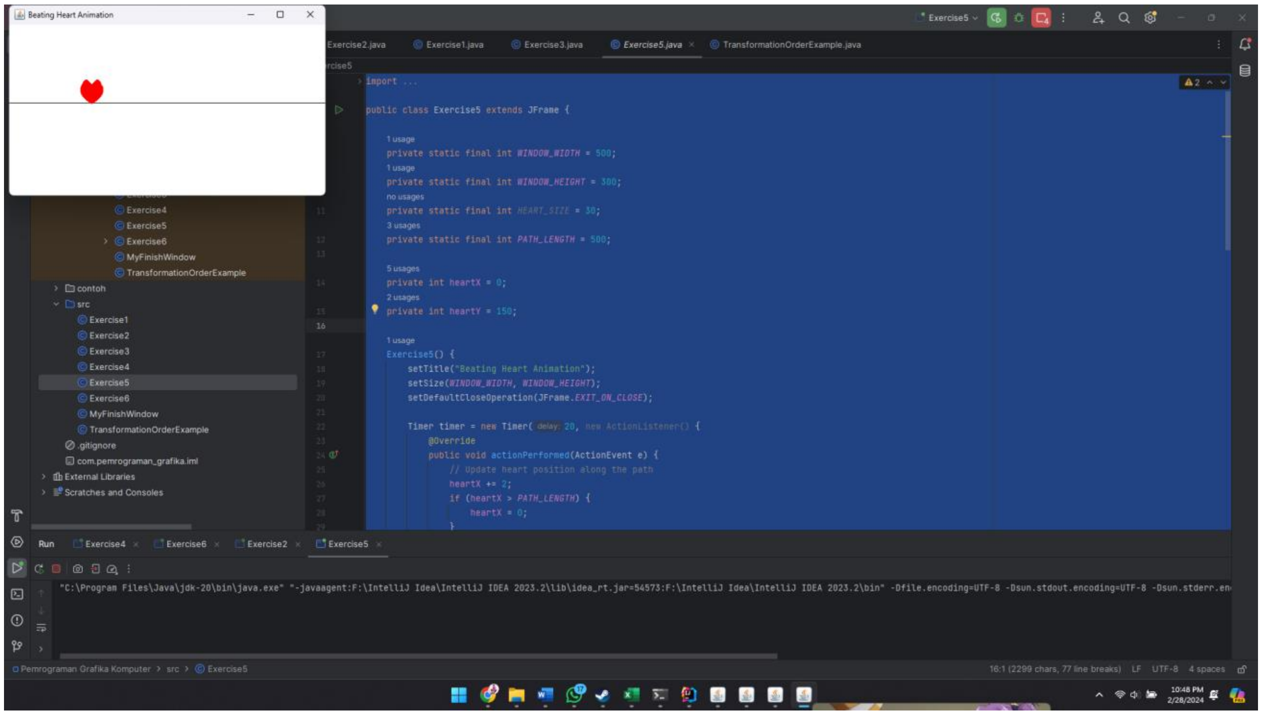
```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:F:\IntelliJ Idea\IntelliJ IDEA 2023.2\lib\idea_rt.jar=54569:F:\IntelliJ Idea\IntelliJ IDEA 2023.2\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.en
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.geom.Ellipse2D;

public class Exercise4 extends JFrame {

    private static final int SUN_RADIUS = 30;
    private static final int PLANET_RADIUS = 10;
    private static final int ORBIT_RADIUS = 100;

    // Number of rotations around its own axis during one orbit
    private static final int ROTATION_SPEED = 365;

    // Angle of the planet in radians
    private double planetAngle = 0;
```

```java
    // Point position on the planet
    private double pointX;
    private double pointY;

    Exercise4() {
        setTitle("Solar System Animation");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Timer timer = new Timer(20, new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Update the planet angle for animation (3.14)
                planetAngle += 2 * Math.PI / ROTATION_SPEED;

                // Calculate the position of the planet
                double planetX = ORBIT_RADIUS * Math.cos(planetAngle);
                double planetY = ORBIT_RADIUS * Math.sin(planetAngle);

                // Update point position on the planet
                pointX = planetX + PLANET_RADIUS * Math.cos(planetAngle);
                pointY = planetY + PLANET_RADIUS * Math.sin(planetAngle);

                // Repaint the frame
                repaint();
            }
        });

        timer.start();
    }

    public void paint(Graphics g) {
        Graphics2D g2d = (Graphics2D) g;

        // Set the coordinate system to the center of the frame
        int centerX = getWidth() / 2;
        int centerY = getHeight() / 2;
        g2d.translate(centerX, centerY);

        // Draw the sun
        g2d.setColor(Color.YELLOW);
        g2d.fill(new Ellipse2D.Double(-SUN_RADIUS, -SUN_RADIUS, 2 *
SUN_RADIUS, 2 * SUN_RADIUS));

        // Calculate the position of the planet
        double planetX = ORBIT_RADIUS * Math.cos(planetAngle);
        double planetY = ORBIT_RADIUS * Math.sin(planetAngle);

        // Draw the planet
        g2d.setColor(Color.BLUE);
        g2d.fill(new Ellipse2D.Double(planetX - PLANET_RADIUS, planetY -
PLANET_RADIUS, 2 * PLANET_RADIUS, 2 * PLANET_RADIUS));

        // Draw the moving point on the planet
        g2d.setColor(Color.RED);
        g2d.fill(new Ellipse2D.Double(pointX - 2, pointY - 2, 4, 4));
    }
    public static void main(String[] argv) {
        SwingUtilities.invokeLater(() -> {
            Exercise4 ex = new Exercise4();
            ex.setVisible(true);
        });
    }
}
```

Exercise 2.25

Hasil Run:



```java
import ...

public class Exercise5 extends JFrame {

    1 usage
    private static final int WINDOW_WIDTH = 500;
    1 usage
    private static final int WINDOW_HEIGHT = 300;
    no usages
    private static final int HEART_SIZE = 30;
    3 usages
    private static final int PATH_LENGTH = 500;

    5 usages
    private int heartX = 0;
    2 usages
    private int heartY = 150;

    1 usage
    Exercise5() {
        setTitle("Beating Heart Animation");
        setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Timer timer = new Timer( delay: 20, new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Update heart position along the path
                heartX += 2;
                if (heartX > PATH_LENGTH) {
                    heartX = 0;
                }
```

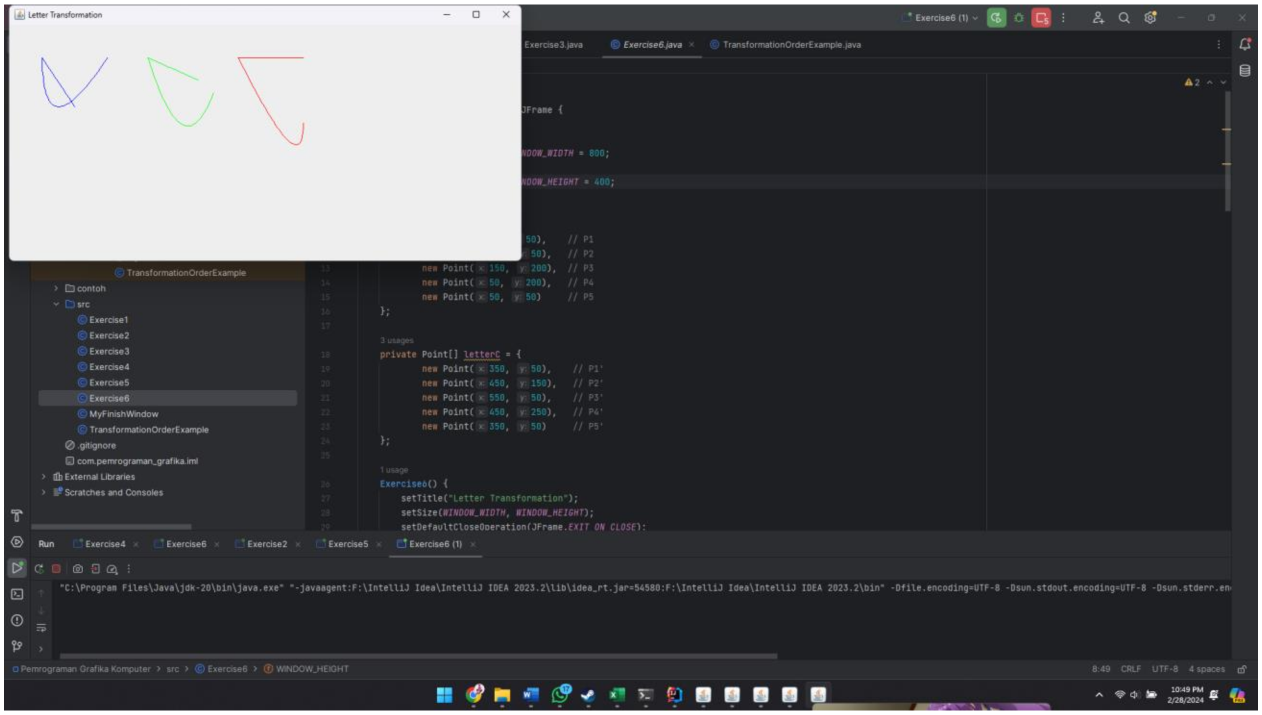
```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:F:\IntelliJ Idea\IntelliJ IDEA 2023.2\lib\idea_rt.jar=54573:F:\IntelliJ Idea\IntelliJ IDEA 2023.2\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.en
```

Kode:

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.geom.GeneralPath;

public class Exercise5 extends JFrame {

    private static final int WINDOW_WIDTH = 500;
    private static final int WINDOW_HEIGHT = 300;
    private static final int HEART_SIZE = 30;
    private static final int PATH_LENGTH = 500;

    private int heartX = 0;
    private int heartY = 150;
    Exercise5() {
        setTitle("Beating Heart Animation");
        setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Timer timer = new Timer(20, new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Update heart position along the path
                heartX += 2;
                if (heartX > PATH_LENGTH) {
                    heartX = 0;
                }

                // Update heart's vertical position with a cosine function
                heartY = 150 + (int) (60 * Math.cos(5 * Math.PI * heartX /
PATH_LENGTH));

                // Repaint the frame
                repaint();
            }
        });

        timer.start();
    }

    public void paint(Graphics g) {
        // Clear the previous drawings by filling the entire panel with a
background color
        g.setColor(Color.WHITE);
        g.fillRect(0, 0, getWidth(), getHeight());

        Graphics2D g2d = (Graphics2D) g;

        // Draw the path
        g2d.setColor(Color.BLACK);
        g2d.drawLine(0, 150, PATH_LENGTH, 150);

        // Draw the beating heart
        drawHeart(g2d, heartX, heartY);
    }

    private void drawHeart(Graphics2D g2d, int x, int y) {
        // Define the shape of a heart
        GeneralPath heart = new GeneralPath();
        heart.moveTo(x, y);
        heart.quadTo(x + 10, y - 15, x + 15, y);
        heart.quadTo(x + 25, y + 15, x, y + 30);
        heart.quadTo(x - 25, y + 15, x - 15, y);
        heart.quadTo(x - 10, y - 15, x, y);

        // Draw the heart
        g2d.setColor(Color.RED);
        g2d.fill(heart);
    }

    public static void main(String[] argv) {
        SwingUtilities.invokeLater(() -> {
            Exercise5 ex = new Exercise5();
            ex.setVisible(true);
        });
    }
}
```

Exercise 2.26

Run :



```
                      JFrame {

                      NDOW_WIDTH = 800;

                      NDOW_HEIGHT = 400;

                       50),    // P1
                       50),    // P2
            new Point( x 150,  y 200),  // P3
            new Point( x 50,  y 200),   // P4
            new Point( x 50,  y 50)     // P5
    };

    3 usages
    private Point[] letterC = {
            new Point( x 350,  y 50),    // P1'
            new Point( x 450,  y 150),   // P2'
            new Point( x 550,  y 50),    // P3'
            new Point( x 450,  y 250),   // P4'
            new Point( x 350,  y 50)     // P5'
    };

    1 usage
    Exercise6() {
        setTitle("Letter Transformation");
        setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:F:\IntelliJ Idea\IntelliJ IDEA 2023.2\lib\idea_rt.jar=54580:F:\IntelliJ Idea\IntelliJ IDEA 2023.2\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.en
```

```
Kode :
import javax.swing.*;
import java.awt.*;
import java.awt.geom.QuadCurve2D;

public class Exercise6 extends JFrame {

    private static final int WINDOW_WIDTH = 800;
    private static final int WINDOW_HEIGHT = 400;

    private Point[] letterD = {
            new Point(50, 50),     // P1
            new Point(150, 50),    // P2
            new Point(150, 200),   // P3
            new Point(50, 200),    // P4
            new Point(50, 50)      // P5
    };
```

```java
    private Point[] letterC = {
            new Point(350, 50),     // P1'
            new Point(450, 150),    // P2'
            new Point(550, 50),     // P3'
            new Point(450, 250),    // P4'
            new Point(350, 50)      // P5'
    };

    Exercise6() {
        setTitle("Letter Transformation");
        setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Set the content pane to a custom JPanel
        setContentPane(new LetterTransformationPanel());

        // Start the animation
        Timer timer = new Timer(100, e -> repaint());
        timer.start();
    }

    private class LetterTransformationPanel extends JPanel {

        private double alpha = 0.0;

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

            Graphics2D g2d = (Graphics2D) g;

            // Draw the letter C
            drawLetter(g2d, letterC, Color.RED);

            // Draw the letter D
            drawLetter(g2d, letterD, Color.BLUE);

            // Draw the intermediate transformation
            Point[] intermediatePoints = calculateIntermediatePoints(alpha);
            drawLetter(g2d, intermediatePoints, Color.GREEN);

            // Update alpha for the next frame
            alpha += 0.02;
            if (alpha > 1.0) {
                alpha = 0.0;
            }
        }

        private void drawLetter(Graphics2D g2d, Point[] points, Color color) {
            g2d.setColor(color);

            QuadCurve2D curve1 = new QuadCurve2D.Double(
                    points[0].getX(), points[0].getY(),
                    points[2].getX(), points[2].getY(),
                    points[4].getX(), points[4].getY()
            );
            QuadCurve2D curve2 = new QuadCurve2D.Double(
                    points[1].getX(), points[1].getY(),
                    points[3].getX(), points[3].getY(),
                    points[4].getX(), points[4].getY()
            );

            g2d.draw(curve1);
            g2d.draw(curve2);
        }

        private Point[] calculateIntermediatePoints(double alpha) {
            Point[] intermediatePoints = new Point[5];
            for (int i = 0; i < 5; i++) {
                double x = (1 - alpha) * letterD[i].getX() + alpha *
letterC[i].getX();
                double y = (1 - alpha) * letterD[i].getY() + alpha *
letterC[i].getY();
                intermediatePoints[i] = new Point((int) x, (int) y);
            }
            return intermediatePoints;
        }
    }

    public static void main(String[] argv) {
        SwingUtilities.invokeLater(() -> {
            Exercise6 ex = new Exercise6();
            ex.setVisible(true);
        });
    }
}
```