SPRAWOZDANIE
Zajęcia: Grafika komputerowa
Prowadzący: prof. dr. hab. Vasyl Martsenyuk

Laboratorium 1
25 II 2021 r.
Temat: „Przekształcenia 2D
w bibliotece Java 2D”
Wariant:
Liczba kątów:5
Figura:1

Robert Laszczak
Informatyka I stopień
Stacjonarne, 4 semestr
Grupa 2B

1. Polecenie

a) Narysować zamiast obrazu wielokąt według wariantu (liczba n) w panelu wyświetlania. Dodać kod do metody paintComponent ().
b) Narysować figurę określoną wariantem, taką jak na rysunku

2. Wprowadzam dane:

Liczba kątów: 5

3. Wykorzystane komendy:

a) Kod źródłowy:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.image.BufferedImage;
import java.io.IOException;
public class Main {
    public static void main(String[] args) throws IOException {
        JFrame window = new JFrame("2D Transforms");
        window.setContentPane(new Transforms2D());
        window.pack();
        window.setResizable(false);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Dimension screen = Toolkit.getDefaultToolkit().getScreenSize();
        window.setLocation((screen.width - window.getWidth()) / 2,
(screen.height - window.getHeight()) / 2);
        window.setVisible(true);
    }
}

class Transforms2D extends JPanel {

    private class Display extends JPanel {
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            Graphics2D g2 = (Graphics2D) g;
            g2.translate(300, 300);  // Moves (0,0) to the center of the
display.
            int whichTransform = transformSelect.getSelectedIndex();

            // TODO Apply transforms here, depending on the value of
whichTransform!

            int[] x = new int[5];
            int[] y = new int[5];

            g2.setStroke(new BasicStroke(20));
            for (int i = 0; i < 5; i++) {
                x[i] = (int) (150 * Math.cos((2 * Math.PI * i) / 5));
                y[i] = (int) (150 * Math.sin((2 * Math.PI * i) / 5));
                System.out.println(x[i]);
            }
            Polygon polygon = new Polygon(x, y, 5);
            // TODO Apply transforms here, depending on the value of
whichTransform!
```

```
            switch (whichTransform) {
                case 0:
                    g2.drawPolygon(polygon);
                    break;
                case 1:
                    g2.scale(0.5,0.5);
                    g2.drawPolygon(polygon);
                    break;
                case 2:
                    g2.rotate(Math.toRadians(45));
                    g2.drawPolygon(polygon);
                    break;
                case 3:
                    g2.rotate(Math.PI);
                    g2.drawPolygon(polygon);
                    break;
                case 4:
                    g2.shear(0.5, 0);
                    g2.drawPolygon(polygon);
                    break;
                case 5:
                    g2.translate(0, -150 * 1.2);
                    g2.scale(1.5, 0.8);
                    g2.drawPolygon(polygon);
                    break;
                case 6:
                    g2.rotate(Math.toRadians(90));
                    g2.shear(0.5,0);
                    g2.drawPolygon(polygon);
                    break;
                case 7:
                    g2.scale(1, 1.5);
                    g2.rotate(Math.PI);
                    g2.drawPolygon(polygon);
                    break;
                case 8:
                    g2.translate(-100, 100);
                    g2.rotate(Math.PI/3);
                    g2.drawPolygon(polygon);
                    break;
                case 9:
                    g2.rotate(Math.PI);
                    g2.translate(-110,0);
                    g2.shear(0.5,0);
                    g2.drawPolygon(polygon);
                    break;
            }
            g2.drawImage(pic, -200, -150, null); // Draw image with center 
at (0,0).
        }
    }

    private Display display;
    private BufferedImage pic;
    private JComboBox<String> transformSelect;

    public Transforms2D() throws IOException {
        // pic = 
ImageIO.read(getClass().getClassLoader().getResource("shuttle.jpg"));
```

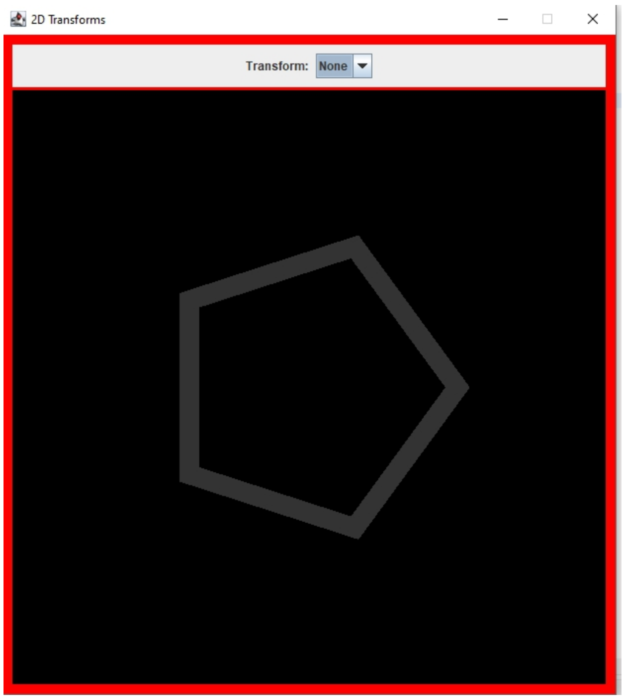
```
        display = new Display();
        display.setBackground(Color.BLACK);
        display.setPreferredSize(new Dimension(600, 600));
        transformSelect = new JComboBox<String>();
        transformSelect.addItem("None");
        for (int i = 1; i < 10; i++) {
            transformSelect.addItem("No. " + i);
        }
        transformSelect.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                display.repaint();
            }
        });
        setLayout(new BorderLayout(3, 3));
        setBackground(Color.RED);
        setBorder(BorderFactory.createLineBorder(Color.RED, 10));
        JPanel top = new JPanel();
        top.setLayout(new FlowLayout(FlowLayout.CENTER));
        top.setBorder(BorderFactory.createEmptyBorder(4, 4, 4, 4));
        top.add(new JLabel("Transform: "));
        top.add(transformSelect);
        add(display, BorderLayout.CENTER);
        add(top, BorderLayout.NORTH);
    }
}
```

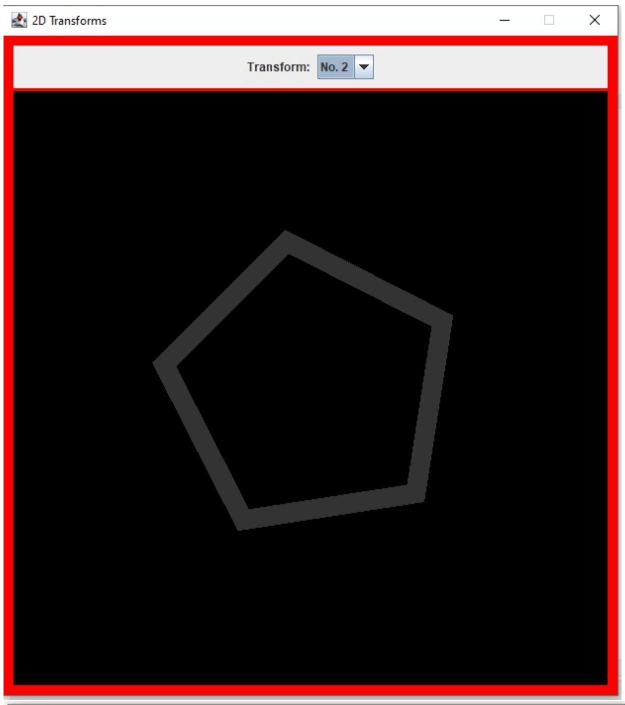
b) Kod źródłowy:

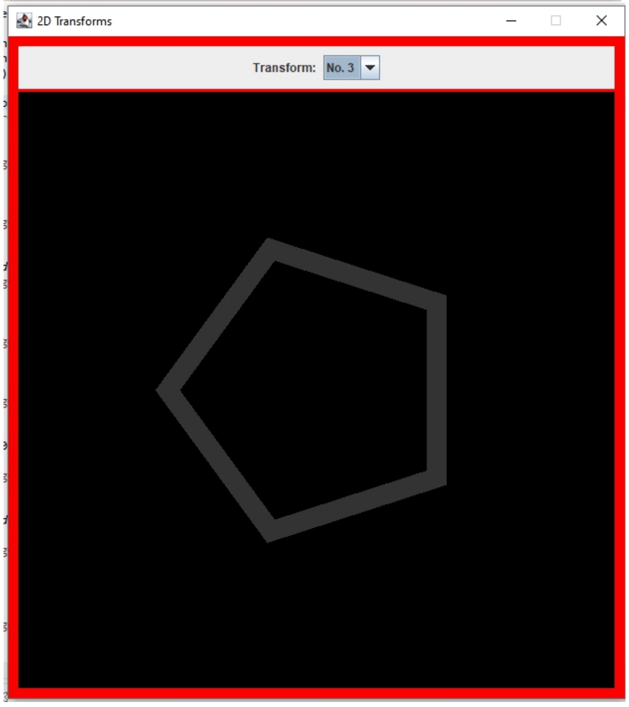
```
      import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.geom.AffineTransform;
import javax.swing.*;
import java.awt.geom.AffineTransform;
public class Main{
    public static void main(String[] args)  {
        JFrame window = new JFrame("Drawing With Transforms");
        window.setContentPane(new TransformedShapes());
        window.pack();
        window.setResizable(false);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Dimension screen = Toolkit.getDefaultToolkit().getScreenSize();
        window.setLocation( (screen.width - window.getWidth())/2,
(screen.height - window.getHeight())/2 );
        window.setVisible(true);
    }
}

class TransformedShapes extends JPanel {

    //------- For drawing ONLY while paintComponent is being executed! ----
----

    /**
       *
       */
      private static final long serialVersionUID = 1L;
      private Graphics2D g2; // A copy of the graphics context from
paintComponent.

    /**
     * Removes any transformations that have been applied to g2, so that
     * it is back to the standard default coordinate system.
     */
    private void resetTransform() {
        g2.setTransform(new AffineTransform());
    }

    /**
     * Draws a filled circle of radius 50 (diameter 100) centered at (0,0),
     * subject to whatever transform(s) have been applied to g2.
     */
    private void circle() {
        g2.fillOval(-50,-50,100,100);
    }

    /**
     * Draws a filled square with side 100 centered at (0,0), subject
     * to whatever transform(s) have been applied to g2.
     */
    private void square() {
        g2.fillRect(-50,-50,100,100);
    }




    //--------------------------------------------------------------------
--------------


    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g2 = (Graphics2D)g.create();
        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);

        // TODO Draw the required image, using ONLY the four methods
defined above,
        // along with g2.setColor, g1.scale, g2.translate, and g2.rotate.

        /* ----------------------------------------------------------------
------- */

        // REMOVE THE FOLLOWING CODE, which draws a big red X in the upper
right quadrant,
        // and insert your own code to draw the required pictures in the
four quadrants.

        // The next two line scale the X to be twice the original size
        // and then moves the center of the X from (0,0) to (150,150).


        g2.translate(150,150);
        g2.scale(2.5,2.5);
        g2.setColor(Color.BLACK);
        circle();

        resetTransform();



        g2.translate(150,150);
        g2.scale(1.25,1.25);
        g2.setColor(Color.YELLOW);
        square();
        resetTransform();



        /* ---------------------------------------------------------------
------- */

    } // end paintComponent()


    //---------------------------------------------------------------------
-----------------

    public TransformedShapes() {
        setPreferredSize(new Dimension(600,600) );
        setBackground(Color.WHITE);
        setBorder(BorderFactory.createLineBorder(Color.BLACK,4));
    }
}
```

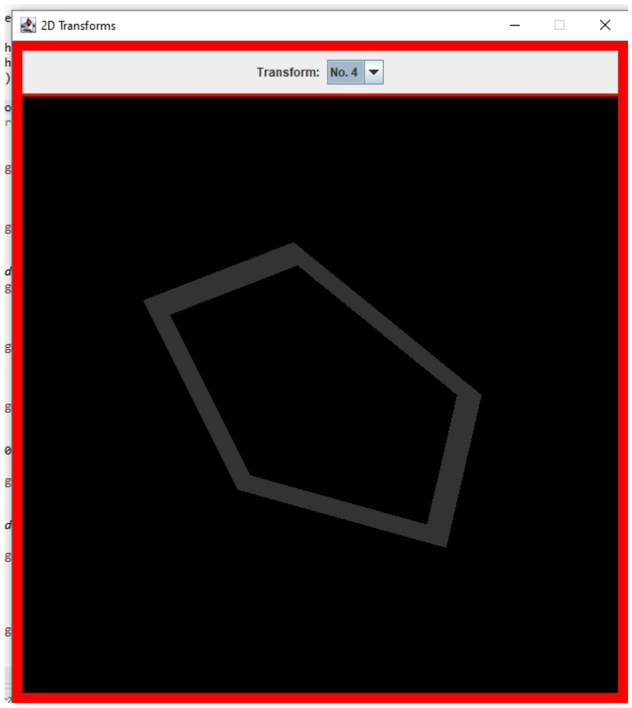
4. Wyniki działania:

a)

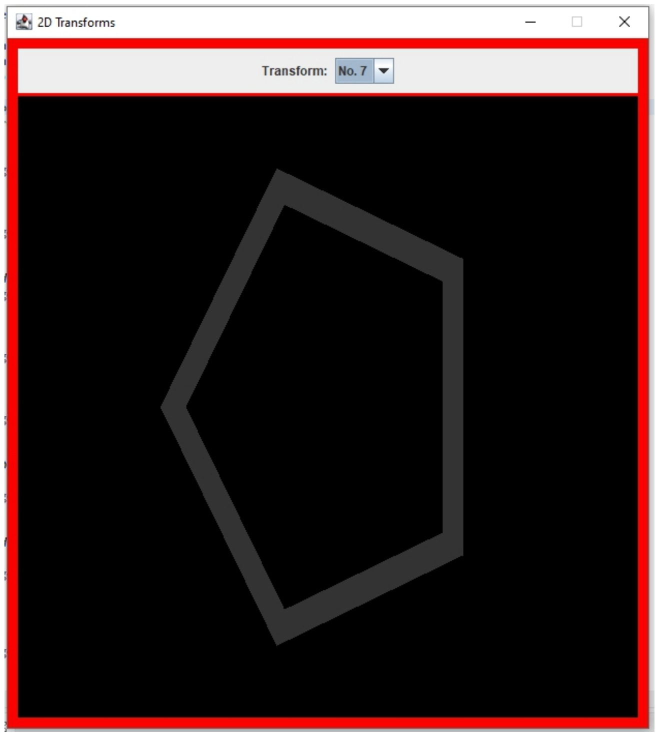



2D Transforms
Transform: No. 2


2D Transforms
Transform: No. 3

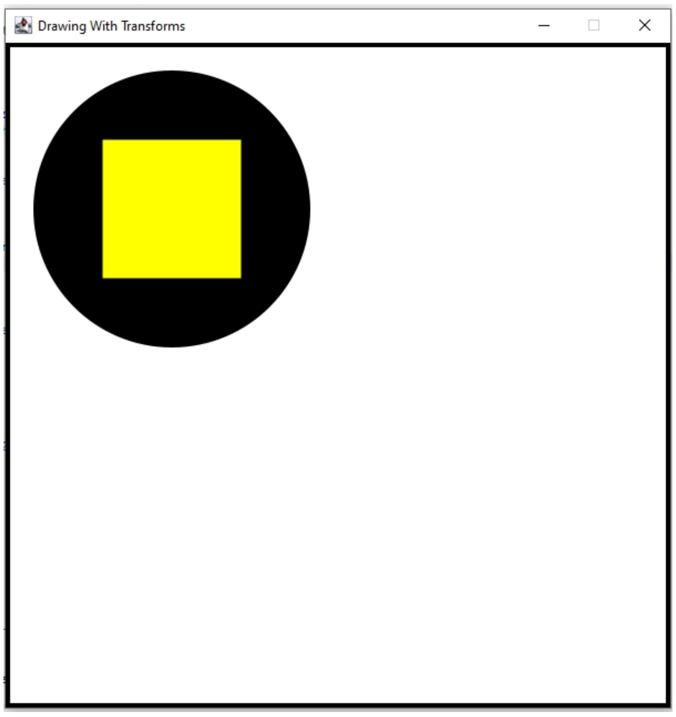

2D Transforms

Transform: No. 4


2D Transforms

Transform: No. 7

b)



5. Wnioski

Na podstawie wykonanych zadań można wyciągnąć następujące wnioski:

A) Wystarczy niewielka zmiana w kodzie by zrobić duże zmiany w grafice

B) Możemy tworzyć różne kombinacje z tworzonych figur, co pozwala na stworzenie dowolnego żądanego obrazu. Jeżeli rozwiniemy kod o dodatkowe figury to możemy stworzyć bardziej zaawansowane figury.