A

LAB REPORT

ON

COMPUTER GRAPHICS AND ANIMATION

By

Prajwal Dahal

Submitted to:

Rakesh Shrestha

Lecturer

Kantipur College of Management and Information Technology

In partial fulfillment of the requirements for the Course

Dot net Technology

Mid Baneshwor, Kathmandu

November 2022

TABLE OF CONTENTS

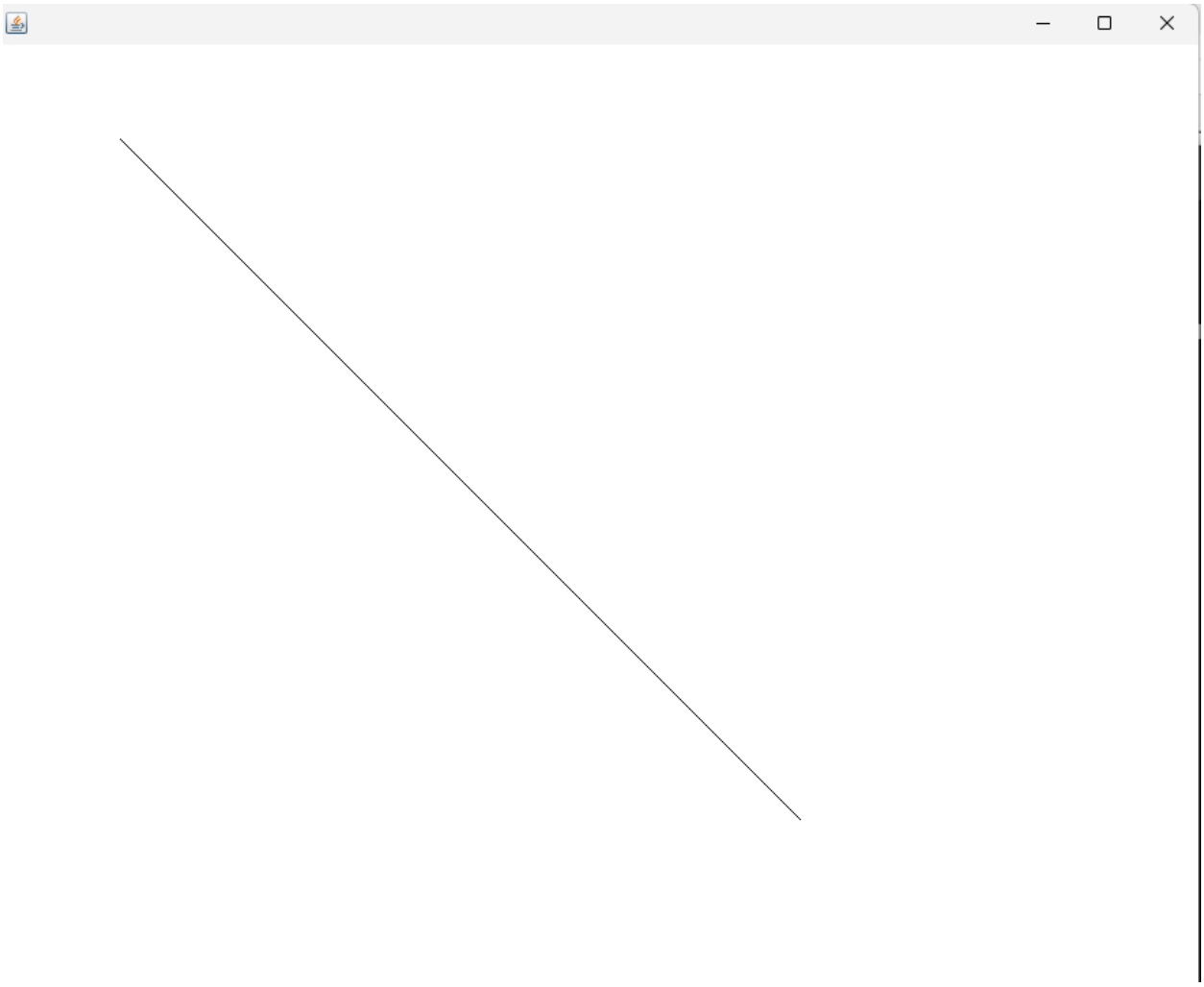
1 Write a program to implement DDA Algorithm in java.

1.1 Source Code

```
import java.awt.*;
import javax.swing.*;
public class DDA extends JFrame{
	public DDA()
	{
		JPanel p = new JPanel();
		getContentPane().add(p);
		setSize(900, 900);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}
	public void paint(Graphics g)
	{
		int x1=100, y1=100, x2=600, y2=600, steps=0;
		int dx=x2-x1;
		int dy = y2-y1;
		if(Math.abs(dx)>Math.abs(dy))
			steps = Math.abs(dx);
		else
			steps = Math.abs(dy);
		int xInc = (int)dx/steps;
		int yInc = (int)dy/steps;
		for(int i=0; i<steps; i++)
		{
			g.drawLine(x1, y1, x1, y1);
			x1= x1+ xInc;
			y1 = y1+yInc;
		}
```

```
	}

	public static void main(String[] args)
	{
		DDA d = new DDA();
		d.setVisible(true);
	}
}
```

1.2 Output Window



2 Write a program to implement bresenhem algorithm in java.

2.1 Source Code

```
import javax.swing.*;
import java.awt.*;
import java.util.Scanner;
class bresenhem extends JFrame{
      private int x1, y1, x2, y2;
      public bresenhem(){
            Scanner sc = new Scanner(System.in);
            System.out.print("enter x1 and y1: ");
            x1=sc.nextInt();
            y1=sc.nextInt();
            System.out.print("enter x2 and y2: ");
            x2=sc.nextInt();
            y2=sc.nextInt();
            setVisible(true);
            setBounds(200,100,400,400);
            setDefaultCloseOperation(EXIT_ON_CLOSE);
      }
      public void slopePositiveLess(int dx,int dy,Graphics g){
            g.drawLine(x1,y1,x1,y1);
            int p0=2*dy-dx;
            for(int i=0;i<dx;i++)
            {
                  if(p0<0){
                        x1++;
                        g.drawLine(x1,y1,x1,y1);
                        p0=p0+2*dy;
```

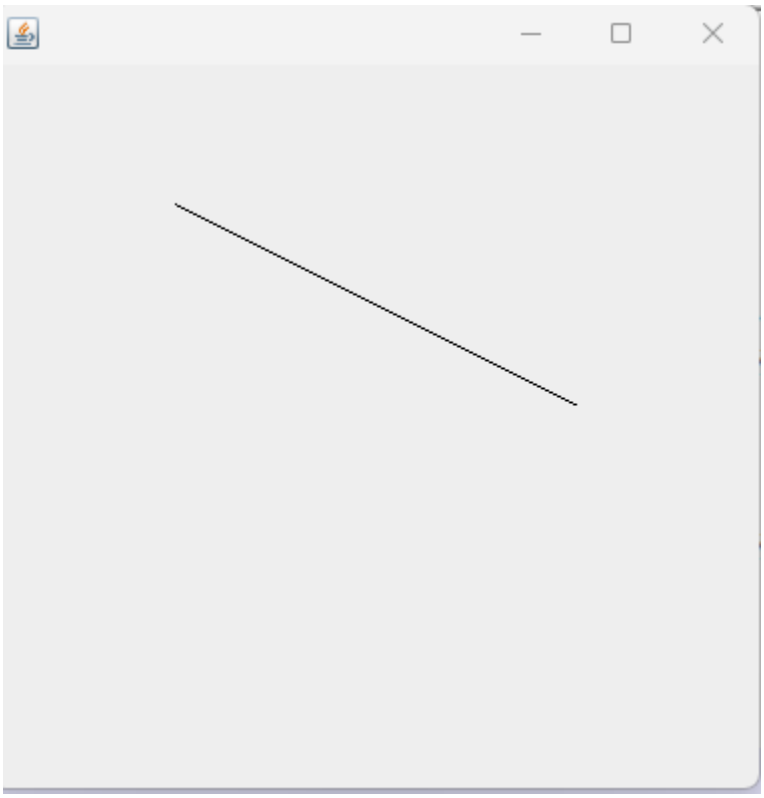
```
            }
            else{
                  x1++;
                  y1++;
                  g.drawLine(x1,y1,x1,y1);
                  p0=p0+2*dy-2*dx;
            }
      }
}
public void slopePositiveGreat(int dx,int dy,Graphics g){
      g.drawLine(x1,y1,x1,y1);
      int p0=2*dx-dy;
      for(int i=0;i<dy;i++){
            if(p0<0){
                  y1++;
                  g.drawLine(x1,y1,x1,y1);
                  p0=p0+2*dx;
            }
            else{
                  x1++;
                  y1++;
                  g.drawLine(x1,y1,x1,y1);
                  p0=p0+2*dx-2*dy;
            }
      }
}
public void slopeNegetiveLess(int dx,int dy,Graphics g){
      g.drawLine(x1,y1,x1,y1);
      int p0=2*dy-dx;
```

```
	for(int i=0;i<dx;i++){
		if(p0<0){
			x1++;
			g.drawLine(x1,y1,x1,y1);
			p0=p0+2*dy;
		}
		else{
			x1++;
			y1--;
			g.drawLine(x1,y1,x1,y1);
			p0=p0+2*dy-2*dx;
		}
	}
}
public void slopeNegetiveGreat(int dx,int dy,Graphics g){
	g.drawLine(x1,y1,x1,y1);
	int p0=2*dx-dy;
	for(int i=0;i<dy;i++){
		if(p0<0){
			y1--;
			g.drawLine(x1,y1,x1,y1);
			p0=p0+2*dx;
		}
		else{
			x1++;
			y1--;
			g.drawLine(x1,y1,x1,y1);
			p0=p0+2*dx-2*dy;
		}
```

```
            }
      }
      public void paint(Graphics g){
            super.paint(g);
            int dx=x2-x1;
            int dy=y2-y1;
            float m = (float)dy/dx;
            float mabs=Math.abs(m);
            dx=Math.abs(dx);
            dy=Math.abs(dy);
            if(mabs < 1 && m > 0)
                  slopePositiveLess(dx,dy,g);
            else if(mabs < 1 && m < 0)
                  slopeNegetiveLess(dx,dy,g);
            else if(mabs > 1 && m>0)
                  slopePositiveGreat(dx,dy,g);
            else
                  slopeNegetiveGreat(dx,dy,g);

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

2.2 Output Window

```
E:\CG>java bresenhem
enter x1 and y1: 100 100
enter x2 and y2: 300 200
```



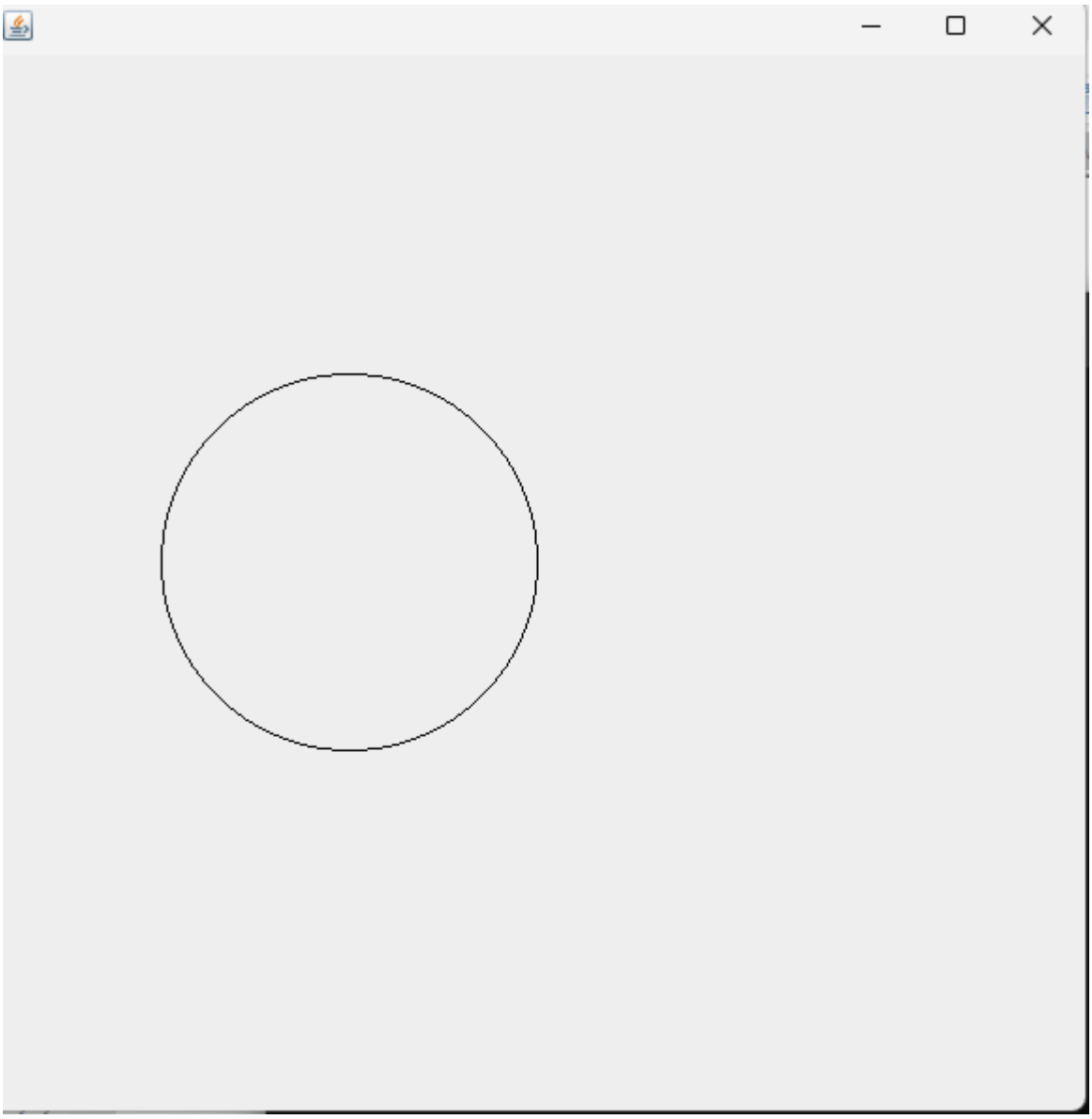
3 Write a program to implement midpoint circle algorithm in java.

3.1 Source Code

```
import java.awt.*;
import javax.swing.*;
class MidCircle extends JFrame{
      int x=200,y=300,r=100;
      public MidCircle(){
            setVisible(true);
            setSize(600,600);
            setDefaultCloseOperation(EXIT_ON_CLOSE);
      }

      public void drawPoints(Graphics g,int x1,int y1){
            g.drawLine(x+x1,y+y1,x+x1,y+y1);
            g.drawLine(x+x1,y-y1,x+x1,y-y1);
            g.drawLine(x-x1,y+y1,x-x1,y+y1);
            g.drawLine(x-x1,y-y1,x-x1,y-y1);
            g.drawLine(x+y1,y+x1,x+y1,y+x1);
            g.drawLine(x-y1,y+x1,x-y1,y+x1);
            g.drawLine(x+y1,y-x1,x+y1,y-x1);
            g.drawLine(x-y1,y-x1,x-y1,y-x1);
      }
      public void drawCircle(Graphics g){
            int x1=0,y1=r;

            drawPoints(g,x1,y1);
            int p0=1-r;
            while(true){
```

```
                if(p0<0){
                        x1++;
                        drawPoints(g,x1,y1);
                        p0=p0+2*x1+1;
                }
                else{
                        x1++;
                        y1--;
                        drawPoints(g,x1,y1);
                        p0=p0+2*x1+1-2*y1;
                }
                if(x1>=y1){
                        break;
                }
        }
    }
    public void paint(Graphics g){
        super.paint(g);
        drawCircle(g);
    }
    public static void main(String []args){
        MidCircle mc = new MidCircle();
    }
}
```

3.2 Output Window



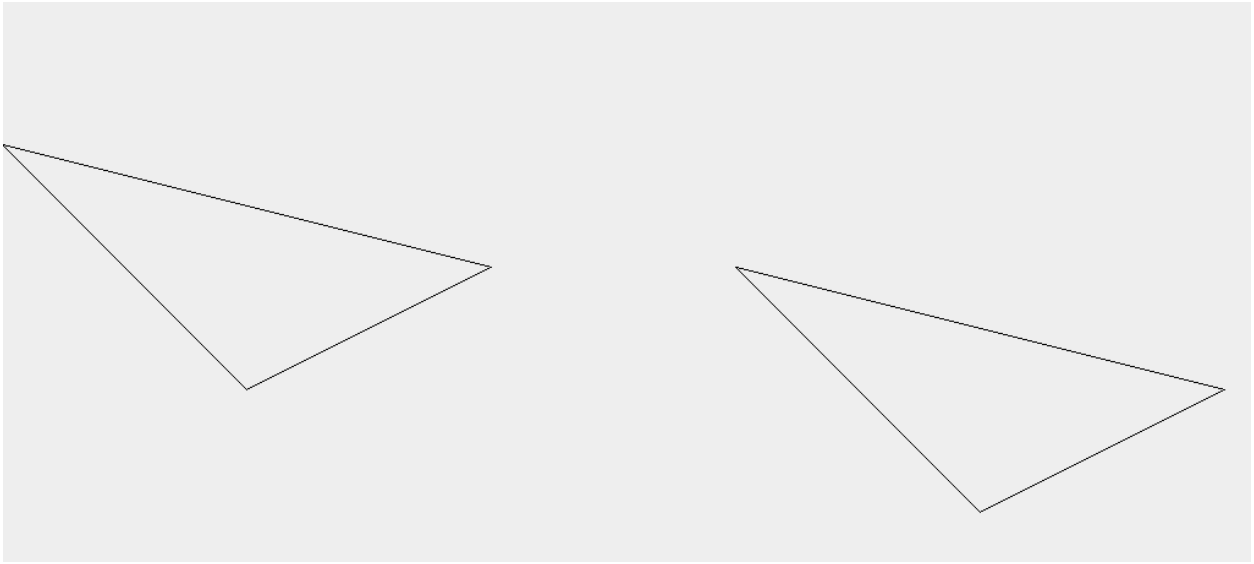
4 Write a program to demonstrate 2D translation in java.

4.1 Source Code

```
import javax.swing.*;
import java.awt.*;
public class translation extends JFrame {

    private int[][]t={{1,0,600},{0,1,100},{0,0,1}}; //translation
matrix
    private int[][]v={{100,500,300},{300,400,500},{1,1,1}}; //vertex

    public translation()
    {
        setBounds(200,10,600,600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setVisible(true);
    }
    void drawTriangle(Graphics g){
        g.drawLine(v[0][0],v[1][0],v[0][1],v[1][1]); //drawing AB
        g.drawLine(v[0][2],v[1][2],v[0][0],v[1][0]); // drawing CA
        g.drawLine(v[0][1],v[1][1],v[0][2],v[1][2]); // drawing BC
    }
    @Override
    public void paint(Graphics g) {
        super.paint(g);
        drawTriangle(g);
        drawTranslation(g);
    }

    private void drawTranslation(Graphics g) {
```

```
            int[][]l= new int[3][3];
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                for (int k = 0; k < 3; k++) {
                    l[i][j]+=t[i][k]*v[k][j];
                }
            }
        }
        g.drawLine(l[0][0],l[1][0],l[0][1],l[1][1]); //drawing AB
        g.drawLine(l[0][2],l[1][2],l[0][0],l[1][0]); // drawing CA
        g.drawLine(l[0][1],l[1][1],l[0][2],l[1][2]); //drawing BC
    }

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

4.2 Output Window



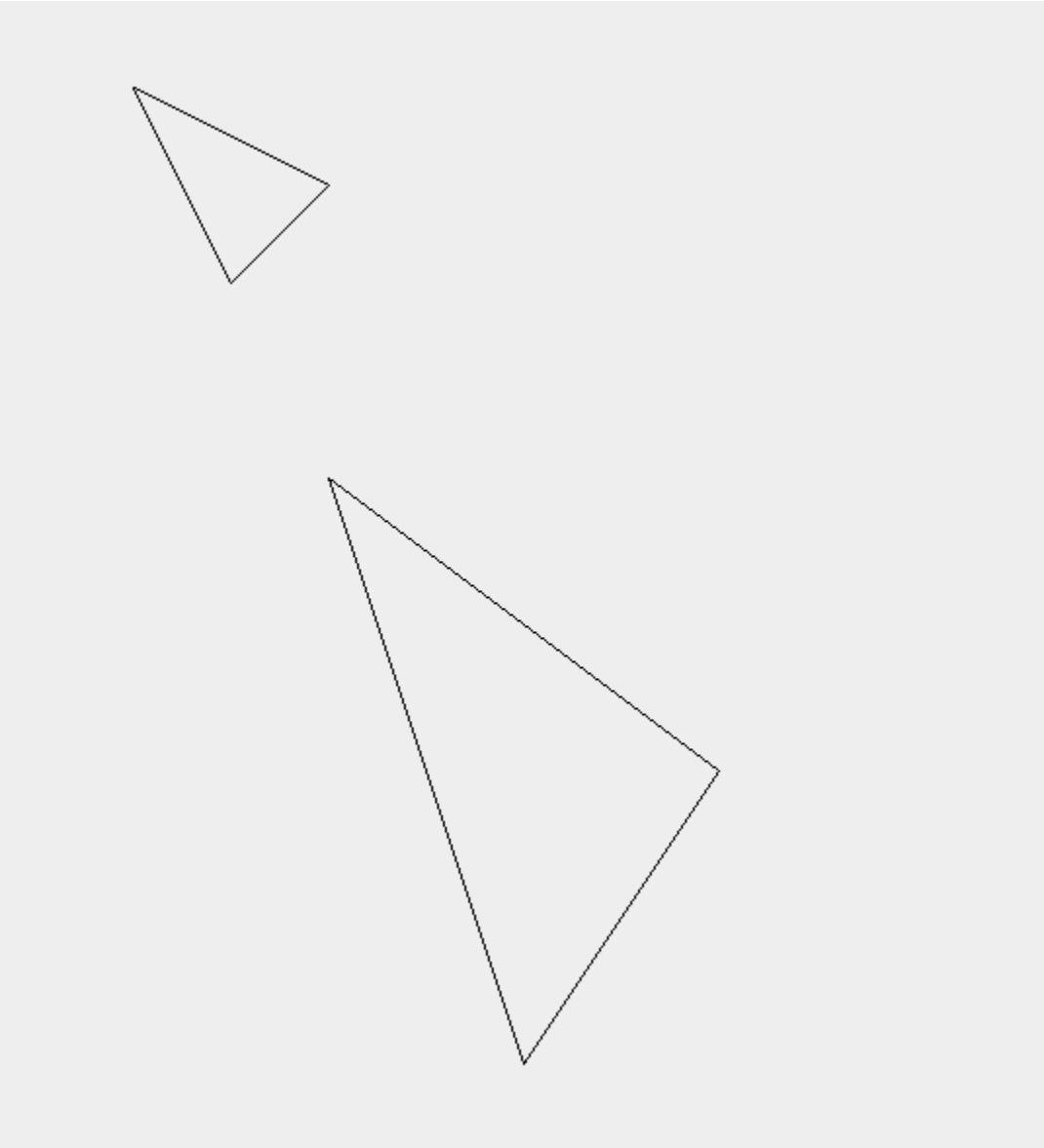
5 Write a program to implement 2D Scaling in java.

5.1 Source Code

```
import javax.swing.*;
import java.awt.*;
public class Scaling  extends JFrame {

    private int[][]s={{2,0,0},{0,3,0},{0,0,1}}; //scaling matrix
    private int[][]v={{100,150,200},{100,200,150},{1,1,1}}; //vertex

    public Scaling()
    {
        setBounds(200,10,600,600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setVisible(true);
    }

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

    private void drawScaling(Graphics g) {
            int[][]l= new int[3][3];
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                for (int k = 0; k < 3; k++) {
                    l[i][j]+=s[i][k]*v[k][j];
```

```
                }
                        System.out.println(l[i][j]);
                }
            }
            g.drawLine(l[0][0],l[1][0],l[0][1],l[1][1]); //drawing AB
            g.drawLine(l[0][2],l[1][2],l[0][0],l[1][0]); // drawing CA
            g.drawLine(l[0][1],l[1][1],l[0][2],l[1][2]); //drawing BC
        }

         void drawTriangle(Graphics g){
            g.drawLine(v[0][0],v[1][0],v[0][1],v[1][1]); //drawing AB
            g.drawLine(v[0][2],v[1][2],v[0][0],v[1][0]); // drawing CA
            g.drawLine(v[0][1],v[1][1],v[0][2],v[1][2]); // drawing BC
        }

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

5.2 Output Windows



6 Write a program to demonstrate 2D rotation about origin.

6.1 Source Code

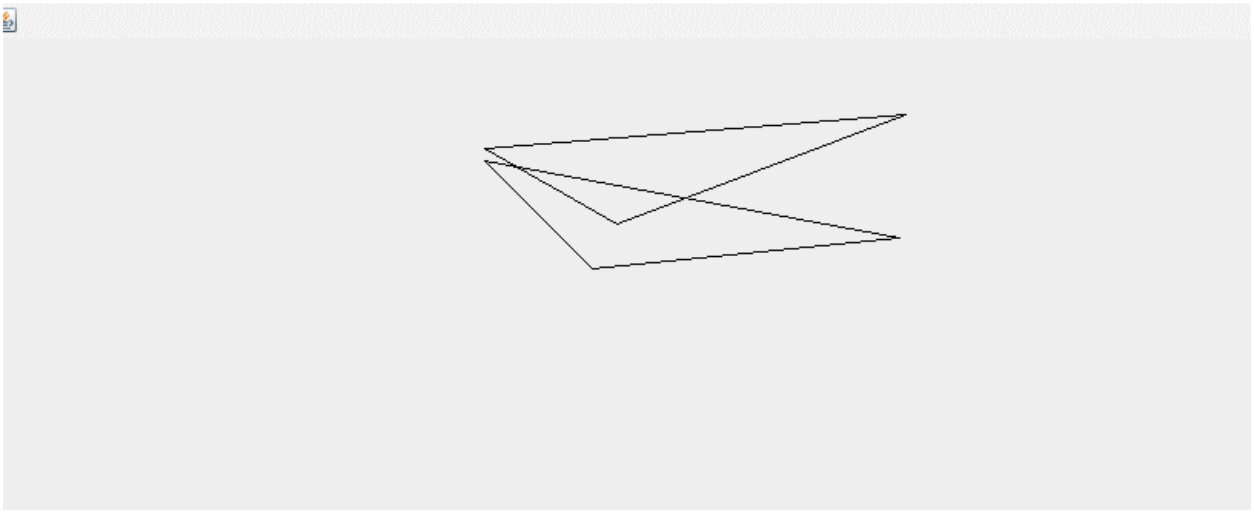
```
import javax.swing.*;
import java.awt.*;
import java.util.Scanner;

public class Rotation extends JFrame {

    private double[][]r; //rotation matrix
    private int[][]v={{100,30,300},{80,10,60},{1,1,1}}; //vertex

    public Rotation()
    {
           Scanner sc = new Scanner(System.in);
           System.out.print("enter a rotation degree: ");
           Double x2=sc.nextDouble();
           r= new double[][]{{Math.cos(x2), -Math.sin(x2), 0} ,
{Math.sin(x2),Math.cos(x2) , 0}, {0, 0, 1}};
        setSize(600,600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setVisible(true);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        drawTriangle(g);
        drawRotation(g);
    }
```

```
private void drawRotation(Graphics g) {
        int[][]l= new int[3][3];
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            for (int k = 0; k < 3; k++) {
                l[i][j]+=r[i][k]*v[k][j];
            }
            System.out.println(l[i][j]);
        }
    }

  g.drawLine(l[0][0]+300,l[1][0]+100,l[0][1]+300,l[1][1]+100);
  //drawing AB
   g.drawLine(l[0][2]+300,l[1][2]+100,l[0][0]+300,l[1][0]+100);
  // drawing CA
   g.drawLine(l[0][1]+300,l[1][1]+100,l[0][2]+300,l[1][2]+100);
  //drawing BC
}

  void drawTriangle(Graphics g){
    g.drawLine(v[0][0]+300, v[1][0]+100, v[0][1]+300,
    v[1][1]+100); //drawing AB
    g.drawLine(v[0][2]+300, v[1][2]+100, v[0][0]+300,
    v[1][0]+100); // drawing CA
    g.drawLine(v[0][1]+300, v[1][1]+100, v[0][2]+300,
    v[1][2]+100); // drawing BC
}

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

```
        new Rotation();
    }
}
```

6.2 Output Window

```
enter a rotation degree: 50
```



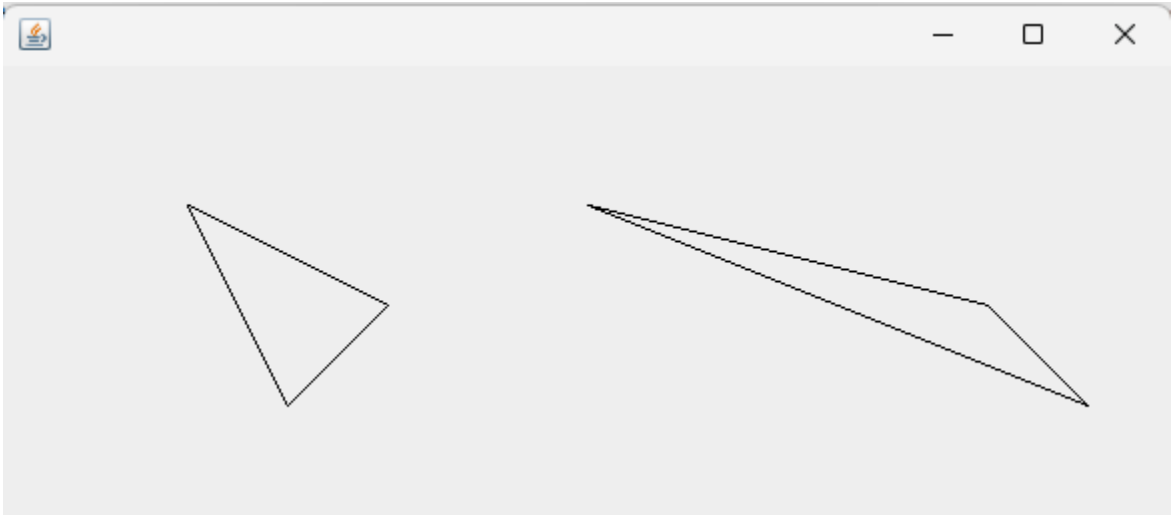
7 Write a program to demonstrate shearing in Y-direction in java.

7.1 Source Code

```
import javax.swing.*;
import java.awt.*;
public class Shearing extends JFrame {

    private int[][]s={{1,2,0},{0,1,0},{0,0,1}}; //shearing matrix
    private int[][]v={{100,150,200},{100,200,150},{1,1,1}}; //vertex

    public Shearing()
    {
        setBounds(200,10,600,600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setVisible(true);
    }

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

    private void drawShear(Graphics g) {
        int[][]l= new int[3][3];
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                for (int k = 0; k < 3; k++) {
                    l[i][j]+=s[i][k]*v[k][j];
```

```
                }
                System.out.println(l[i][j]);
            }
        }
        g.drawLine(l[0][0],l[1][0],l[0][1],l[1][1]); //drawing AB
        g.drawLine(l[0][2],l[1][2],l[0][0],l[1][0]); // drawing CA
        g.drawLine(l[0][1],l[1][1],l[0][2],l[1][2]); //drawing BC
    }

    void drawTriangle(Graphics g){
        g.drawLine(v[0][0],v[1][0],v[0][1],v[1][1]); //drawing AB
        g.drawLine(v[0][2],v[1][2],v[0][0],v[1][0]); // drawing CA
        g.drawLine(v[0][1],v[1][1],v[0][2],v[1][2]); // drawing BC
    }

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

7.2 Output Window



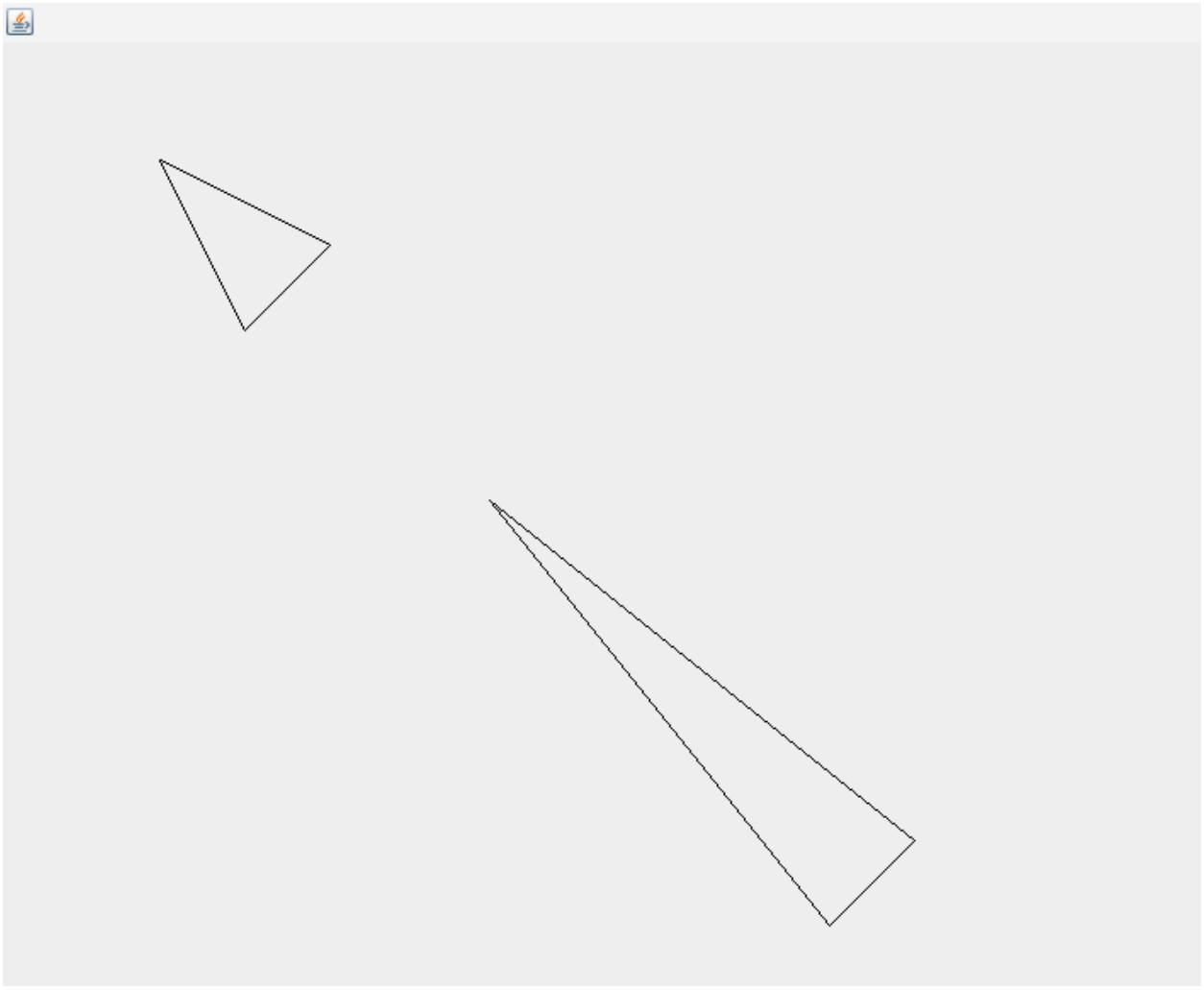
8 Write a program to demonstrate shearing in X-direction relative to other reference.

8.1 Source Code

```
import javax.swing.*;

import java.awt.*;
public class ShearingX extends JFrame {

    private int[][]s={{1,2,-2*3},{2,1,0},{0,0,1}}; //shearing matrix
    private int[][]v={{100,150,200},{100,200,150},{1,1,1}}; //vertex

    public ShearingX()
    {
        setBounds(200,10,600,600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setVisible(true);
    }

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

    private void drawScaling(Graphics g) {
        int[][]l= new int[3][3];
        for (int i = 0; i < 3; i++) {
```

```
            for (int j = 0; j < 3; j++) {
                for (int k = 0; k < 3; k++) {
                    l[i][j]+=s[i][k]*v[k][j];
                }
                System.out.println(l[i][j]);
            }
        }
        g.drawLine(l[0][0],l[1][0],l[0][1],l[1][1]); //drawing AB
        g.drawLine(l[0][2],l[1][2],l[0][0],l[1][0]); // drawing CA
        g.drawLine(l[0][1],l[1][1],l[0][2],l[1][2]); //drawing BC
    }

    void drawTriangle(Graphics g){
        g.drawLine(v[0][0],v[1][0],v[0][1],v[1][1]); //drawing AB
        g.drawLine(v[0][2],v[1][2],v[0][0],v[1][0]); // drawing CA
        g.drawLine(v[0][1],v[1][1],v[0][2],v[1][2]); // drawing BC
    }

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

8.2 Output Window



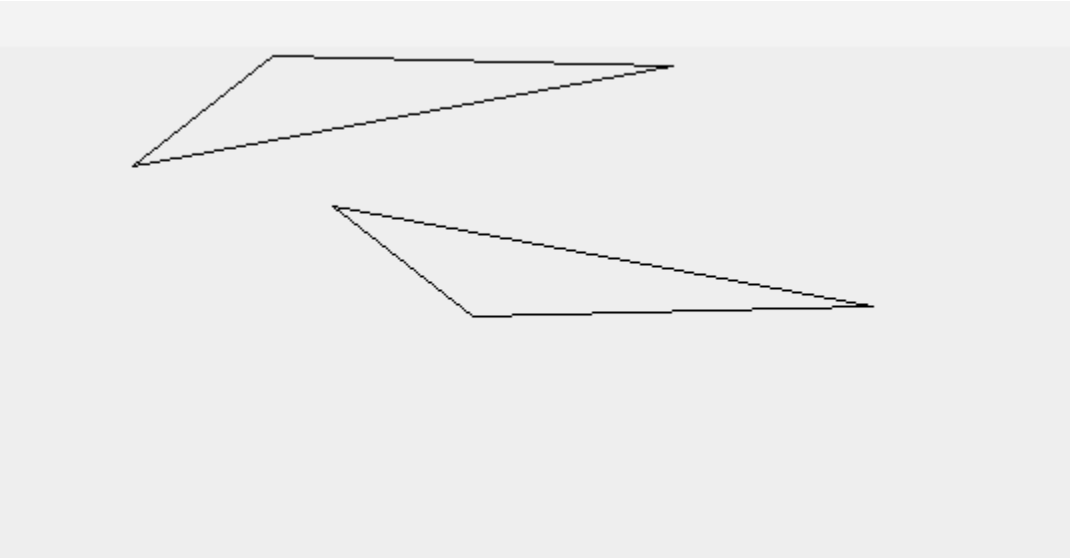
9 Write a Program to demonstrate reflection about X-axis.

9.1 Source Code

```
import javax.swing.*;
import java.awt.*;
public class Reflection extends JFrame {

    private int[][]s={{1,0,0},{0,-1,0},{0,0,1}}; //reflection matrix
    private int[][]v={{100,30,300},{65,10,60},{1,1,1}}; //vertex

    public Reflection()
    {
        setSize(600,600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setVisible(true);
    }

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

    private void drawScaling(Graphics g) {
        int[][]l= new int[3][3];
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                for (int k = 0; k < 3; k++) {
                    l[i][j]+=s[i][k]*v[k][j];
```

```
                }
                System.out.println(l[i][j]);
            }
        }

      g.drawLine(l[0][0]+200,l[1][0]+100,l[0][1]+200,l[1][1]+100);
      //drawing AB
      g.drawLine(l[0][2]+200,l[1][2]+100,l[0][0]+200,l[1][0]+100); //
      drawing CA
      g.drawLine(l[0][1]+200,l[1][1]+100,l[0][2]+200,l[1][2]+100);
      //drawing BC
    }

    void drawTriangle(Graphics g){
        g.drawLine(v[0][0]+300, v[1][0]+100, v[0][1]+300, v[1][1]+100);
      //drawing AB
       g.drawLine(v[0][2]+300, v[1][2]+100, v[0][0]+300, v[1][0]+100);
      // drawing CA
      g.drawLine(v[0][1]+300, v[1][1]+100, v[0][2]+300, v[1][2]+100);
      // drawing BC
    }

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

9.2 Output Window



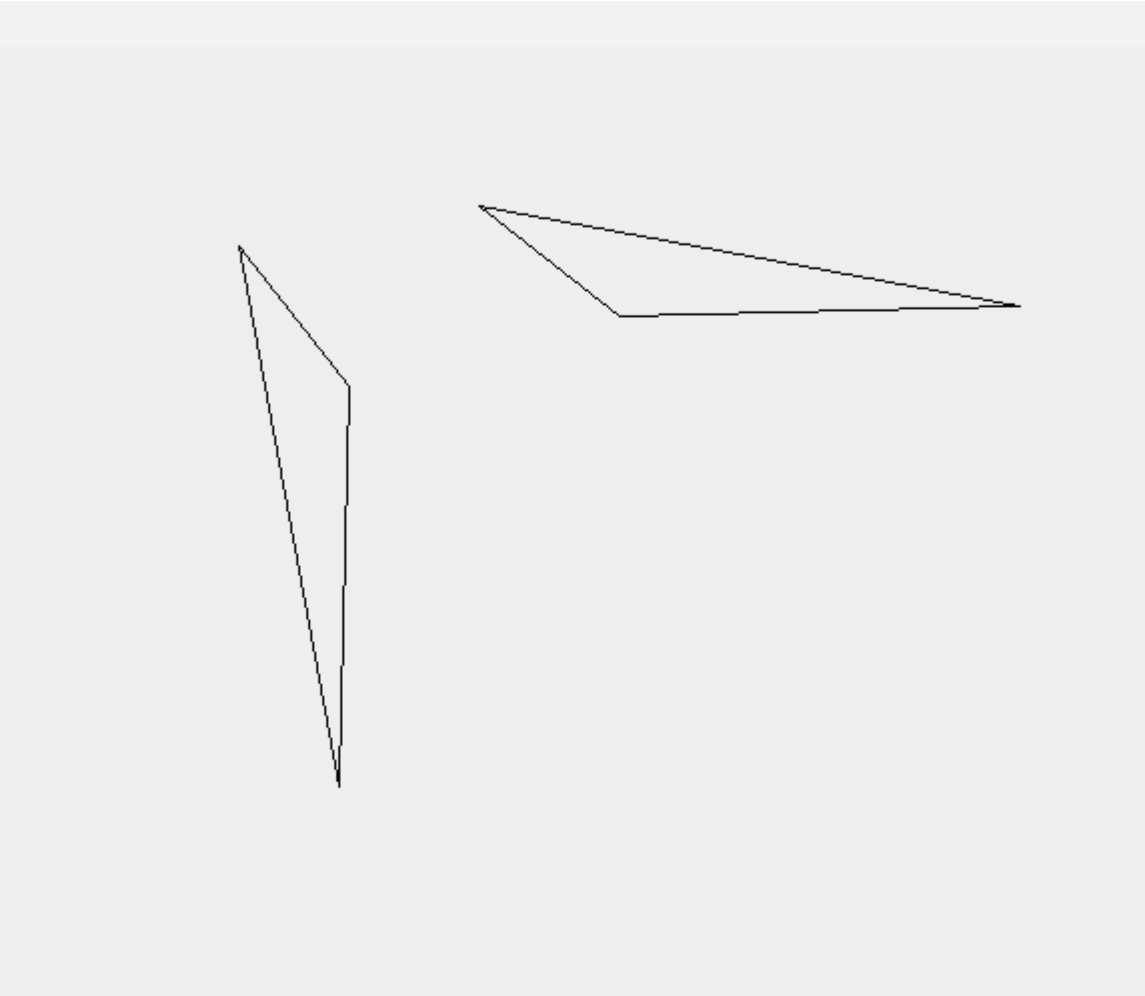
10 Write a program to demonstrate reflection about diagonal y=x.

10.1 Source Code

```
import javax.swing.*;

import java.awt.*;
public class Reflection extends JFrame {

    private int[][]s={{0,1,0},{1,0,0},{0,0,1}}; //reflection matrix
    private int[][]v={{100,30,300},{65,10,60},{1,1,1}}; //vertex

    public Reflection()
    {
        setSize(600,600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setVisible(true);
    }

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

    private void drawScaling(Graphics g) {
        int[][]l= new int[3][3];
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                for (int k = 0; k < 3; k++) {
```

```
                    l[i][j]+=s[i][k]*v[k][j];
                }
                System.out.println(l[i][j]);
            }
        }

      g.drawLine(l[0][0]+200,l[1][0]+100,l[0][1]+200,l[1][1]+100);
      //drawing AB
      g.drawLine(l[0][2]+200,l[1][2]+100,l[0][0]+200,l[1][0]+100); //
      drawing CA
      g.drawLine(l[0][1]+200,l[1][1]+100,l[0][2]+200,l[1][2]+100);
      //drawing BC
    }

    void drawTriangle(Graphics g){
       g.drawLine(v[0][0]+300, v[1][0]+100, v[0][1]+300, v[1][1]+100);
      //drawing AB
      g.drawLine(v[0][2]+300, v[1][2]+100, v[0][0]+300,
      v[1][0]+100); // drawing CA
      g.drawLine(v[0][1]+300, v[1][1]+100, v[0][2]+300, v[1][2]+100);
      // drawing BC
    }

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

10.2 Output Window



11 Write a Program to draw ellipse using midpoint ellipse algorithm.

11.1 Source Code

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

public class MidEllipse extends JFrame {
    int x=300,y=300,rx=100,ry=200,twory,tworx;
    public MidEllipse(){
        tworx=2*rx*rx;
        twory=2*ry*ry;
        setVisible(true);
        setSize(600,600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    public void drawPoint(Graphics g,int x1,int y1){
        g.drawLine(x+x1,y+y1,x+x1,y+y1);
        g.drawLine(x+x1,y-y1,x+x1,y-y1);
        g.drawLine(x-x1,y+y1,x-x1,y+y1);
        g.drawLine(x-x1,y-y1,x-x1,y-y1);
    }

    public void drawRadiusX(Graphics g){
        int x1=0,y1=ry;
        int ryd=twory*x1;
        int rxd=tworx*y1;

        double p0=ry*twory-rx*rx*ry+(1/4*rx*rx);
        System.out.println(p0+ " first");
```

```
        while(ryd<rxd){

            if(p0<0){
                x1++;
                drawPoint(g,x1,y1);
                p0=p0+ryd+ry*ry;
                System.out.println(p0+","+x1+","+y1);
            }
            else{
                x1++;
                y1--;
                drawPoint(g,x1,y1);
                p0=p0+ryd+ry*ry-rxd;
                System.out.println(p0+","+x1+","+y1);
            }
            ryd=2*ry*ry*x1;
            rxd=2*rx*rx*y1;

        }
        drawRadiusY(g,x1,y1);
    }

    private void drawRadiusY(Graphics g, int x1, int y1) {
        int ryd=twory*x1;
        int rxd=twory*y1;
        int p0= ry*ry*(x1+1/2)*(x1+1/2)+rx*rx*(y1-1)*(y1-1)-
rx*rx*ry*ry;

        System.out.println("r2"+p0);
```

```
        while(y1>0){

            if(p0>0){
                y1--;
                drawPoint(g,x1,y1);
                p0=p0-rxd+rx*rx;
                System.out.println(p0+","+x1+","+y1);
            }
            else{
                x1++;
                y1--;
                drawPoint(g,x1,y1);
                p0=p0+ryd+rx*rx-rxd;
                System.out.println(p0+","+x1+","+y1);
            }
            ryd=2*ry*ry*x1;
            rxd=2*rx*rx*y1;
        }
    }

    public void paint(Graphics g){
        super.paint(g);
        drawRadiusX(g);
    }
    public static void main(String []args){
        new MidEllipse();
    }
}
```

11.2 Output Window