**Lab 1: Digital Differential Analyzer (DDA)**

**Objective:** To draw a straight line starting from (x1, y1) to (x2, y2).

**Tools:** NetBeans

**Algorithm:**

1. Start
2. Input end points (x1, y1) and (x2, y2).​
3. Calculate dx = x2-x1 and dy = y2-y1​
4. if (abs(dx) > abs(dy))​
   * 1. steps = abs(dx);​

else​

* + 1. steps = abs(dy);​

1. incrx = dx/steps and incry = dy/steps;​
2. (x, y) = (x1, y1)    /\* first point to plot \*/​
3. putpixel(round(x), round(y),1); //1 is color parameter​
4. for (k = 1; k <= steps;k++)​

{​

x = x + incrx;​

y = y + incry;​

putpixel(round(x), round(y),1);​

 }​

1. Stop.

CODE:

import java.awt.\*;

import javax.swing.JFrame;

import java.util.Scanner;

import java.lang.\*;

public class DDA extends Canvas

{

static int x1,y1,x2,y2;

DDA(int x1,int y1,int x2,int y2)

{

this.x1=x1;

this.y1=y1;

this.x2=x2;

this.y2=y2;

}

public void paint(Graphics g)

{

int dx,dy;

float Xinc,Yinc,x,y,steps;

dy=y2-y1;

dx=x2-x1;

g.fillOval(x1,y1,10,10);

if(Math.abs(dy)>Math.abs(dx))

{

steps=Math.abs(dy);

}

else

{

steps=Math.abs(dx);

}

x=x1;

y=y1;

Yinc=dy/steps;

Xinc=dx/steps;

while(steps!=0)

{

steps--;

x=x+Xinc;

y=y+Yinc;

g.fillOval(Math.round(x),Math.round(y),10,10);

}

}

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter first end x1 and y1: ");

int x1=sc.nextInt();

int y1=sc.nextInt();

System.out.println("Enter last end x2 and y2: ");

int x2=sc.nextInt();

int y2=sc.nextInt();

DDA d=new DDA(x1,y1,x2,y2);

JFrame f=new JFrame();

f.add(d);

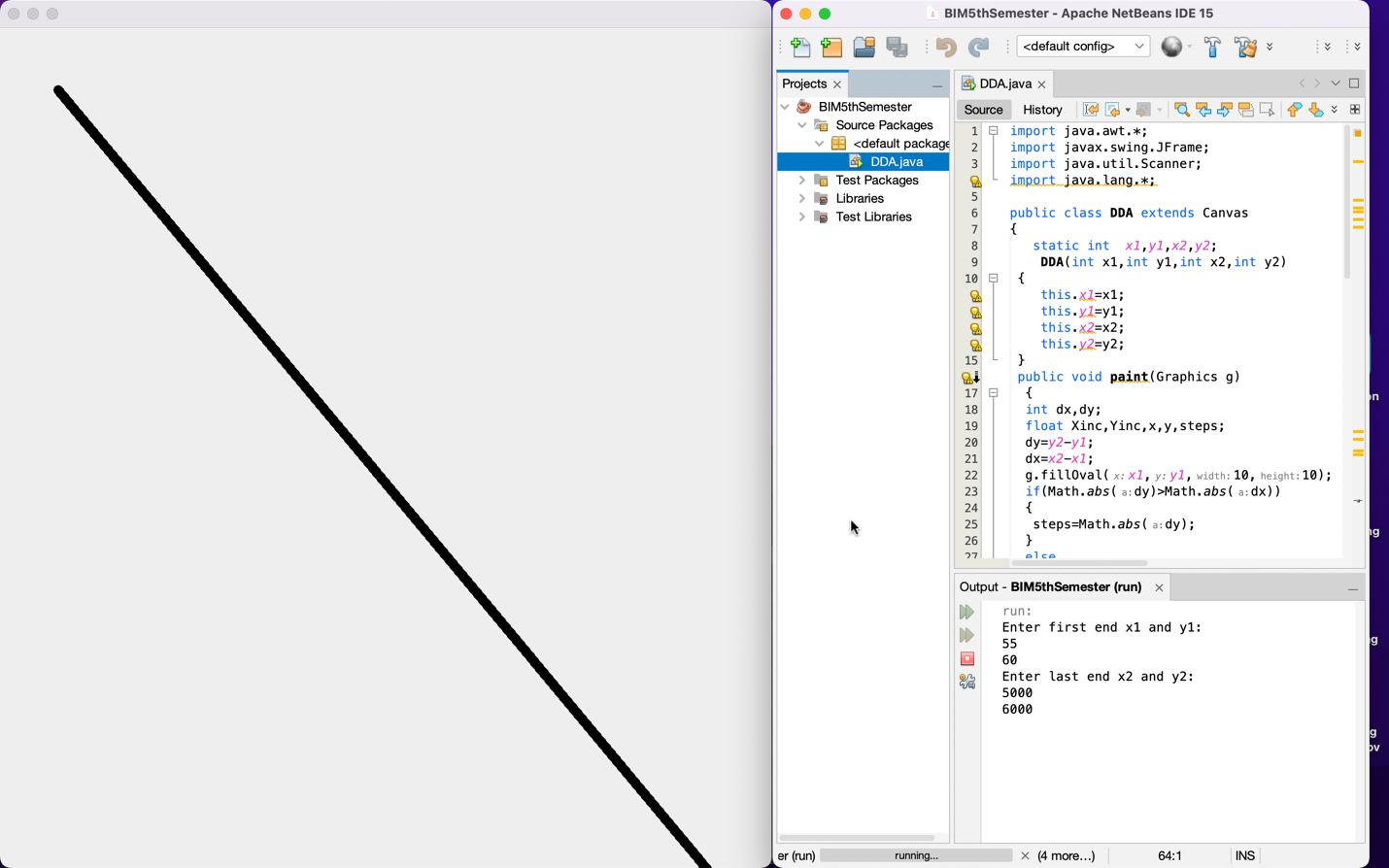
f.setSize(800,800);

f.setVisible(true);

}

}

**OUTPUT:**



**CONCLUSION:**

DDA Algorithm has been implemented to draw a straight line from one end point to another end point.

# Lab 2: Bresenham's Line Algorithm

**Objective:** To draw a straight line starting from (x1, y1) to (x2, y2).

**Tools:** NetBeans

**Algorithm:**

1. Start
2. Input end points (x1, y1) and (x2, y2).​
3. Calculate dx= |x2-x1| and dy= |y2-y1|.
4. If dx>dy, steps=dx, else, steps=dy.
5. If x1<x2, xinc= +1, else, xinc= -1.
6. If y1<y2, yinc= +1, else, yinc= -1.
7. Initially, (x1, y1) was plotted.
8. If (dy<dx) /\* m<1 \*/

{

P0=2\*dy-dx

for(k=1;k<=steps;k++)

{

xk+1=xk+xinc;

if (Pk<0){yk+1=yk; Pk+1=Pk+2\*dy;}

else{yk+1=yk+yinc; Pk+1=Pk+2\*dy-2\*dx;}

Plot (xk+1, yk+1)

}

}

1. Else /\* dy>=dx i.e. m>=1 \*/

{

P0=2\*dx-dy

for(k=1;k<=steps;k++)

{

yk+1=yk+yinc;

if (Pk<0){xk+1=xk; Pk+1=Pk+2\*dx;}

else{xk+1=xk+xinc; Pk+1=Pk+2\*dx-2\*dy;}

Plot (xk+1, yk+1)

}

}

1. Stop

CODE:

import java.awt.\*;

import javax.swing.JFrame;

import java.util.Scanner;

import java.lang.\*;

public class Bresenham extends Canvas

{

int x1,y1,x2,y2;

Bresenham(int x1,int y1,int x2,int y2)

{

this.x1=x1;

this.y1=y1;

this.x2=x2;

this.y2=y2;

}

public void paint(Graphics g)

{

int dx,dy,Xinc,Yinc,x,y,steps,p;

dy=Math.abs(y2-y1);

dx=Math.abs(x2-x1);

if(x2>x1)

Xinc=1;

else

Xinc=-1;

if(y2>y1)

Yinc=1;

else

Yinc=-1;

g.fillOval(x1,y1,10,10);

if(dy>dx)

steps=dy;

else

steps=dx;

x=x1;

y=y1;

if(dx>dy)

{

p=2\*dy-dx;

while(steps!=0)

{

steps--;

if(p<0)

{

x=x+Xinc;

y=y;

p=p+2\*dy;

}

else

{

x=x+Xinc;

y=y+Yinc;

p=p+2\*dy-2\*dx;

}

g.fillOval(Math.round(x),Math.round(y),10,10);

}

}

else

{

p=2\*dx-dy;

while(steps!=0)

{

steps--;

if(p<0)

{

x=x;

y=y+Yinc;

p=p+2\*dx;

}

else

{

x=x+Xinc;

y=y+Yinc;

p=p+2\*dx-2\*dy;

}

g.fillOval(Math.round(x),Math.round(y),10,10);

}

}

}

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter first end x1 and y1: ");

int x1=sc.nextInt();

int y1=sc.nextInt();

System.out.println("Enter last end x2 and y2: ");

int x2=sc.nextInt();

int y2=sc.nextInt();

Bresenham d=new Bresenham(x1,y1,x2,y2);

JFrame f=new JFrame();

f.add(d);

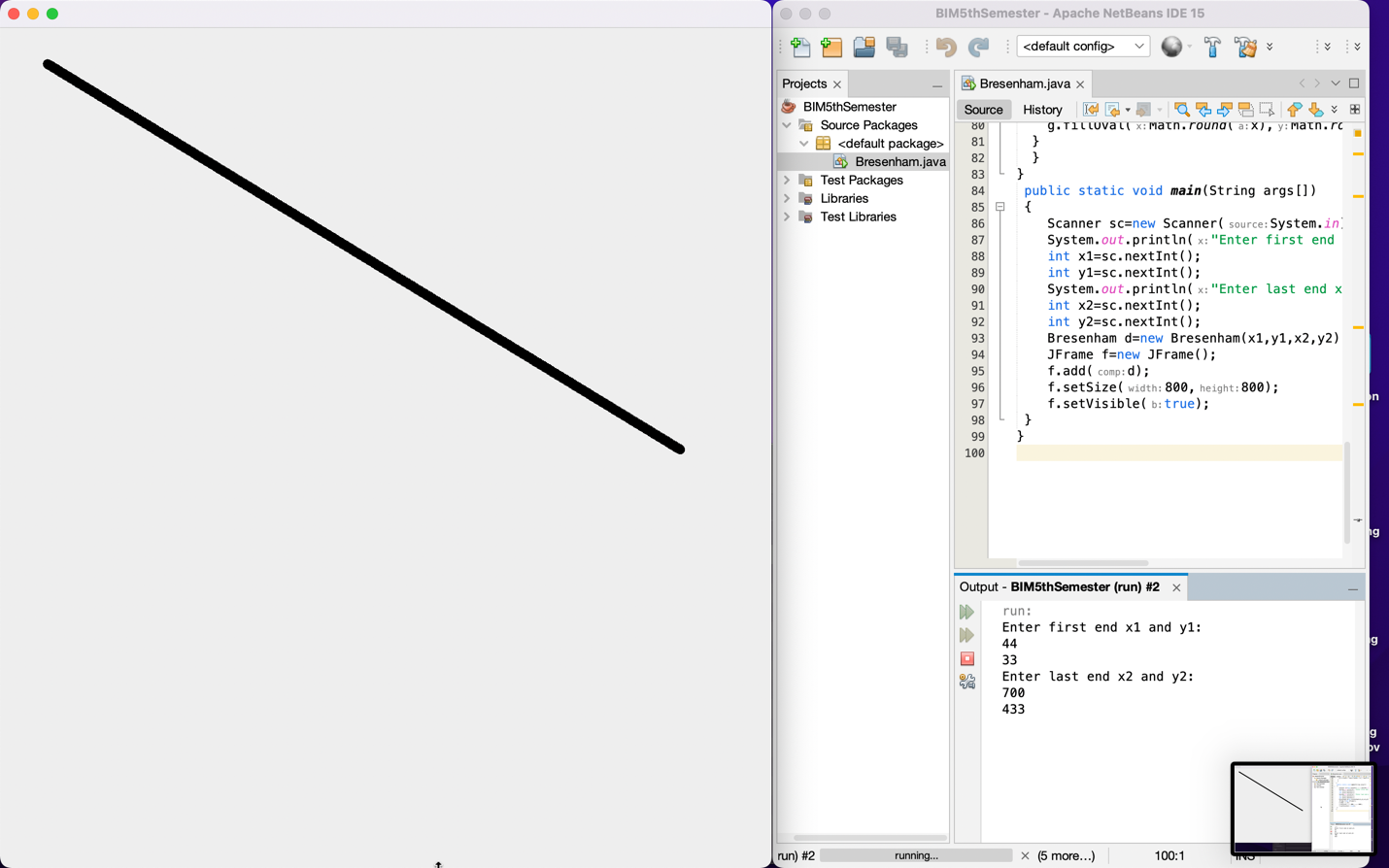
f.setSize(800,800);

f.setVisible(true);

}

}

**OUTPUT:**



**CONCLUSION:**

Bresenham Algorithm has been implemented to draw a straight line from one end point to another end point.

**LAB 3 : Mid-Point Circle Drawing Algorithm**

**Objective:** To draw a mid-point circle starting from (x1, y1) and radius r.

**Tools:** NetBeans

**Algorithm:**

1. Input radius r and circle centre (xc, yc) and obtain the first point on circle centered at origin as (x0, y0) = (0, r)
2. Calculate initial decision parameter  P0=5/4-r
3. At each xk position, starting at k=0 perform the tests:

If pk<0 next point along the circle centre at (0,0) is (xk+1, yk)

Pk+1=pk+2xk+1+1

Otherwise, the next point along circle is (xk+1, yk-1)

pk+1=pk+2xk+1+1-2yk+1

Where 2xk+1=2xk+2 and 2yk+1=2yk-2

1. Determine symmetry point on the other seven octants.
2. Move each calculated pixels positions (x, y) in to circle path centered at (xc, yc) as

x=x+xc, y=y+yc

1. Repeat 3 through 5 until x>=y.

CODE:

import java.awt.\*;

import javax.swing.JFrame;

import java.util.Scanner;

import java.lang.\*;

public class MP\_CIRCLE extends Canvas

{

static int xc,yc,r;

MP\_CIRCLE(int xc,int yc,int r)

{

this.xc=xc;

this.yc=yc;

this.r=r;

}

public void paint(Graphics g)

{

int x,y,p;

x=0;

y=r;

fill(g,x,y,xc,yc);

p=1-r;

while(x<y)

{

x=x+1;

if(p<0)

{

p=p+2\*x+1;

}

else

{

y=y-1;

p=p+2\*x+1-2\*y;

}

fill(g,x,y,xc,yc);

}

}

public void fill(Graphics g,int x,int y,int xc,int yc)

{

g.fillOval(xc+x,yc+y,5,5);

g.fillOval(xc+x,yc-y,5,5);

g.fillOval(xc-x,yc+y,5,5);

g.fillOval(xc-x,yc-y,5,5);

g.fillOval(xc+y,yc+x,5,5);

g.fillOval(xc+y,yc-x,5,5);

g.fillOval(xc-y,yc+x,5,5);

g.fillOval(xc-y,yc-x,5,5);

}

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter center (xc,yc): ");

int xc=sc.nextInt();

int yc=sc.nextInt();

System.out.println("Enter radius r: ");

int r=sc.nextInt();

MP\_CIRCLE c = new MP\_CIRCLE(xc,yc,r);

JFrame f=new JFrame();

f.add(c);

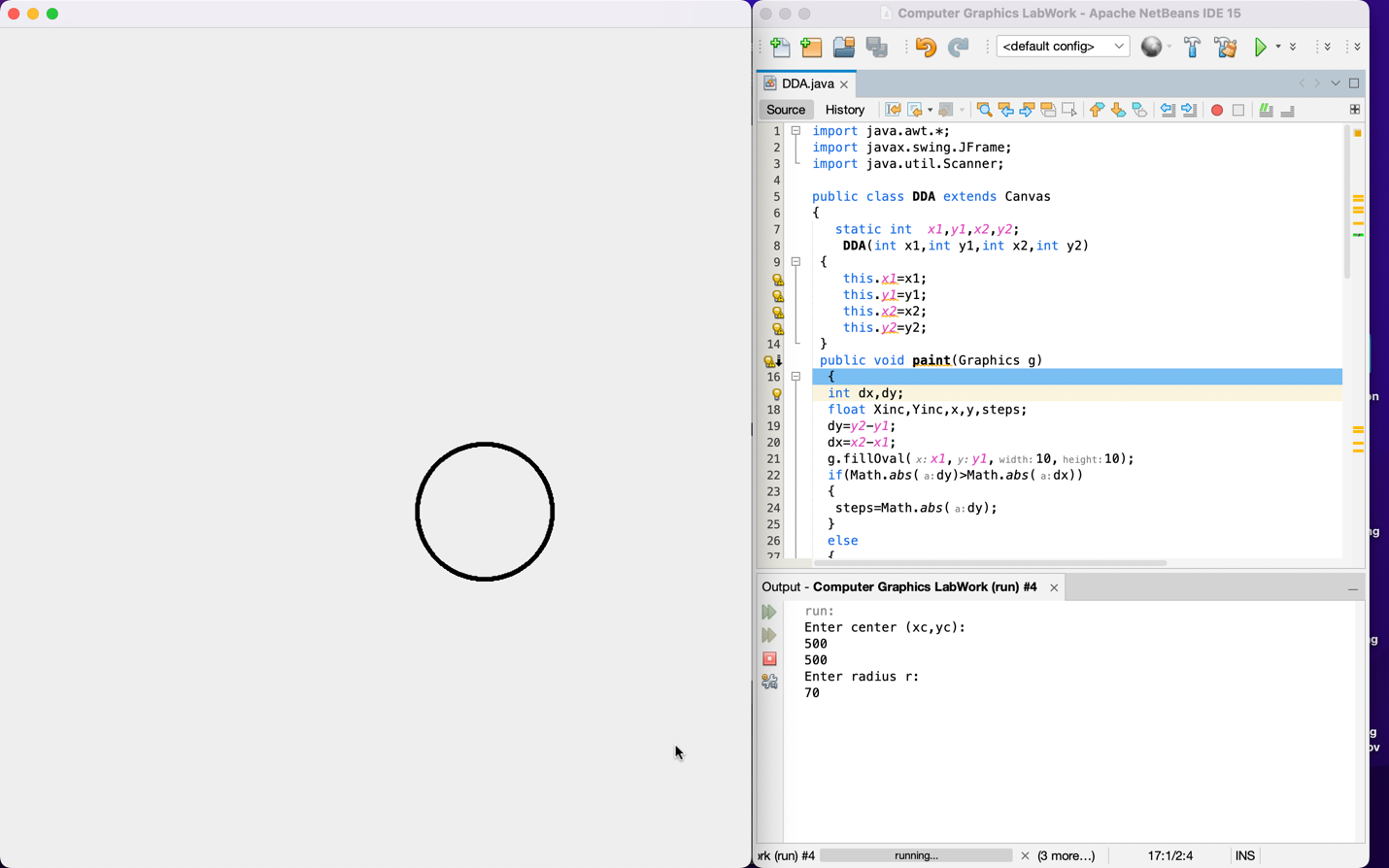
f.setSize(800,800);

f.setVisible(true);

}

}

**OUTPUT:**



**CONCLUSION:**

Mid-point Circle drawing algorithm has been used to draw a circle by entering user input.

**LAB 4 : Point Translation**

**Objective:** To reposition an point object from a coordinate location to another coordinate according to Translation algorithm.

**Tools:** NetBeans

**Algorithm:**

1. Start
2. Input end points (x1, y1)
3. Construct a 2D object (use drawline( )) e.g. (x,y)
4. Get the translation value tx, ty
5. Move the 2D object with tx, ty (x’=x+tx, y’=y+ty)
6. Plot (x’, y’)
7. Stop.

CODE:

import java.awt.\*;

import javax.swing.JFrame;

import java.util.Scanner;

import java.lang.\*;

public class Translation extends Canvas

{

static int x,y,tx,ty;

Translation(int x,int y,int tx,int ty)

{

this.x=x;

this.y=y;

this.tx=tx;

this.ty=ty;

}

public void paint(Graphics g)

{

int xn,yn;

g.setColor(Color.blue);

g.fillOval(x,y,10,10);

xn=x+tx;

yn=y+ty;

g.setColor(Color.red);

g.fillOval(xn,yn,10,10);}

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter point (x,y): ");

int x=sc.nextInt();

int y=sc.nextInt();

System.out.println("Enter translation factors: ");

int tx=sc.nextInt();

int ty=sc.nextInt();

Translation t=new Translation(x,y,tx,ty);

JFrame f=new JFrame();

f.add(t);

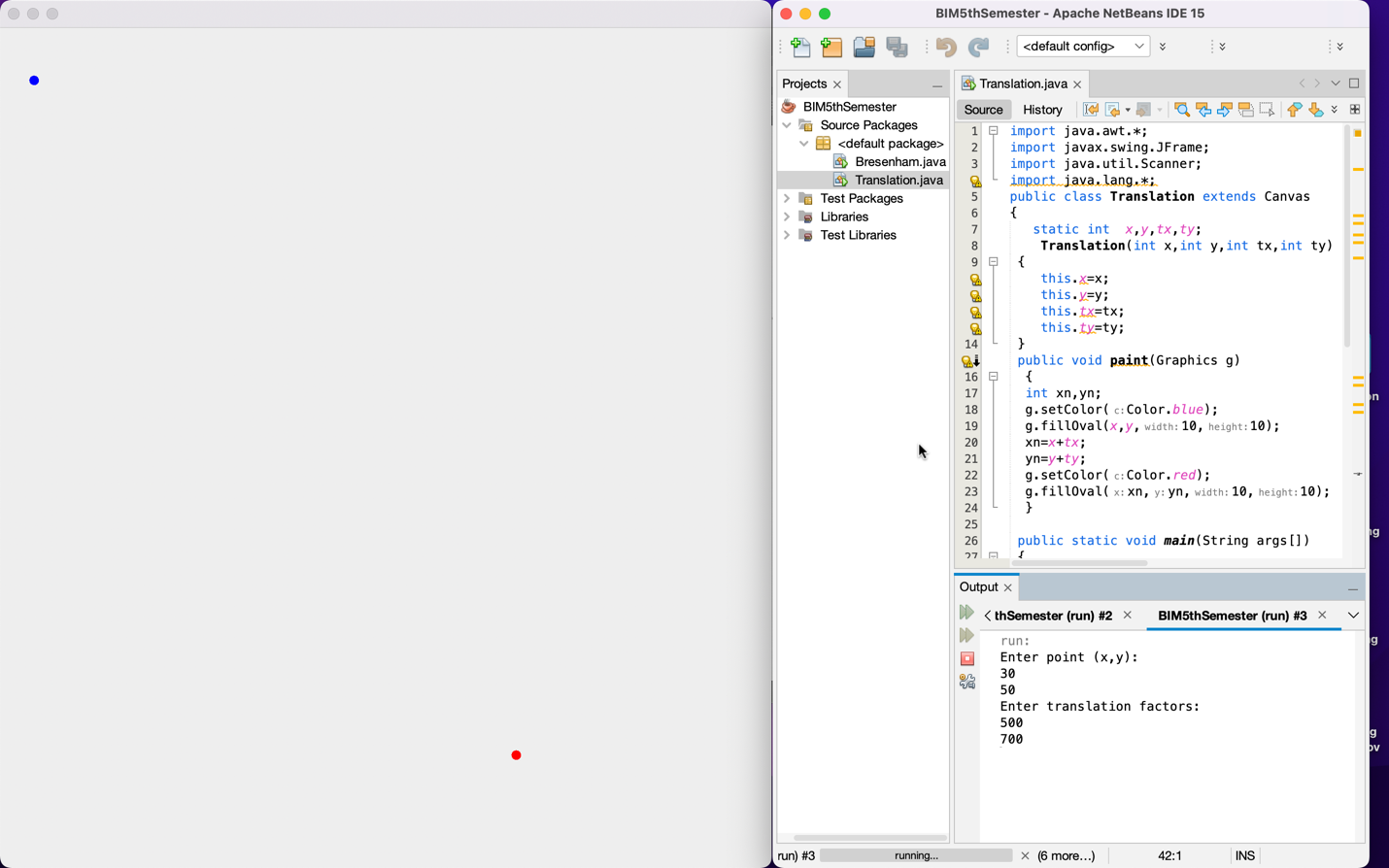
f.setSize(800,800);

f.setVisible(true);

}

}

**OUTPUT:**



**CONCLUSION:**

We have translated a point entered by user to another place from the effect of translation factors.

**Lab 5 : Line Translation**

**Objective:** To reposition an object along a straight-line path from a coordinate location to another coordinate according to Translation algorithm.

**Tools:** NetBeans

**Algorithm:**

1. Start
2. Input end points (x1, y1) and (x2, y2).​
3. Construct a 2D object (use drawline( )) e.g. (x,y)
4. Get the translation value tx, ty
5. Move the 2D object with tx, ty (x’=x+tx, y’=y+ty)
6. Plot (x’, y’)
7. Stop

CODE:

import java.awt.\*;

import javax.swing.JFrame;

import java.util.Scanner;

import java.lang.\*;

public class Translation2 extends Canvas

{

static int x1,y1,x2,y2,tx,ty;

Translation2(int x1,int y1,int x2,int y2,int tx,int ty)

{

this.x1=x1;

this.y1=y1;

this.x2=x2;

this.y2=y2;

this.tx=tx;

this.ty=ty;

}

public void paint(Graphics g)

{

int xn1,yn1,xn2,yn2;

g.setColor(Color.blue);

g.drawLine(x1,y1,x2,y2);

xn1=x1+tx;

yn1=y1+ty;

xn2=x2+tx;

yn2=y2+ty;

g.setColor(Color.red);

g.drawLine(xn1,yn1,xn2,yn2);

}

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter first point (x1,y1): ");

int x1=sc.nextInt();

int y1=sc.nextInt();

System.out.println("Enter last point (x2,y2): ");

int x2=sc.nextInt();

int y2=sc.nextInt();

System.out.println("Enter translation factors: ");

int tx=sc.nextInt();

int ty=sc.nextInt();

Translation2 t2=new Translation2(x1,y1,x2,y2,tx,ty);

JFrame f=new JFrame();

f.add(t2);

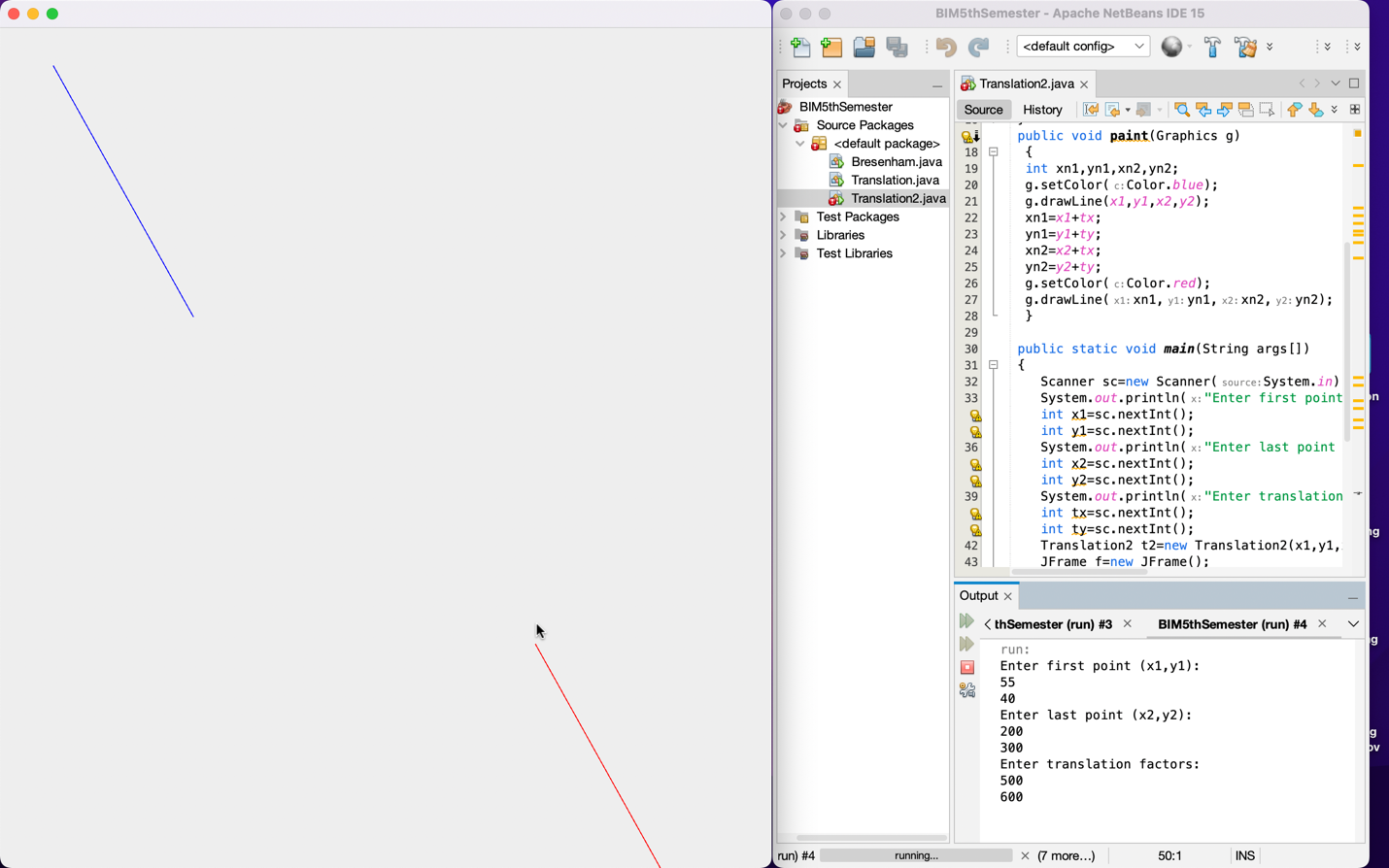
f.setSize(800,800);

f.setVisible(true);

}

}

**OUTPUT:**



**Conclusion:**

We have translated a line from one position to another by using the translation factors

**Lab 6 - Rectangle Translation**

**Objective:** To reposition an rectangle object from a coordinate location to another coordinate according to Translation algorithm.

**Tools:** NetBeans

**Algorithm:**

1. Start
2. Input end points (x1, y1) and (x2, y2).​
3. Construct a 2D object (use drawRect( )) e.g. (x,y)
4. Get the translation value tx, ty
5. Move the 2D object with tx, ty (x’=x+tx, y’=y+ty)
6. Plot (x’, y’)
7. stop

CODE:

import java.awt.\*;

import javax.swing.JFrame;

import java.util.Scanner;

import java.lang.\*;

public class Translation4 extends Canvas{

static int x1,y1,x2,y2,tx,ty;

Translation4(int x1,int y1,int x2,int y2,int tx,int ty)

{

this.x1=x1;

this.y1=y1;

this.x2=x2;

this.y2=y2;

this.tx=tx;

this.ty=ty;

}

public void paint(Graphics g)

{

int xn1,yn1,xn2,yn2;

g.setColor(Color.blue);

g.drawRect(x1,y1,x2,y2);

xn1=x1+tx;

yn1=y1+ty;

xn2=x2+tx;

yn2=y2+ty;

g.setColor(Color.red);

g.drawRect(xn1,yn1,xn2,yn2);

}

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter lower left corner (x1,y1): ");

int x1=sc.nextInt();

int y1=sc.nextInt();

System.out.println("Enter upper right corner (x2,y2): ");

int x2=sc.nextInt();

int y2=sc.nextInt();

System.out.println("Enter translation factors: ");

int tx=sc.nextInt();

int ty=sc.nextInt();

Translation4 t4=new Translation4(x1,y1,x2,y2,tx,ty);

JFrame f=new JFrame();

f.add(t4);

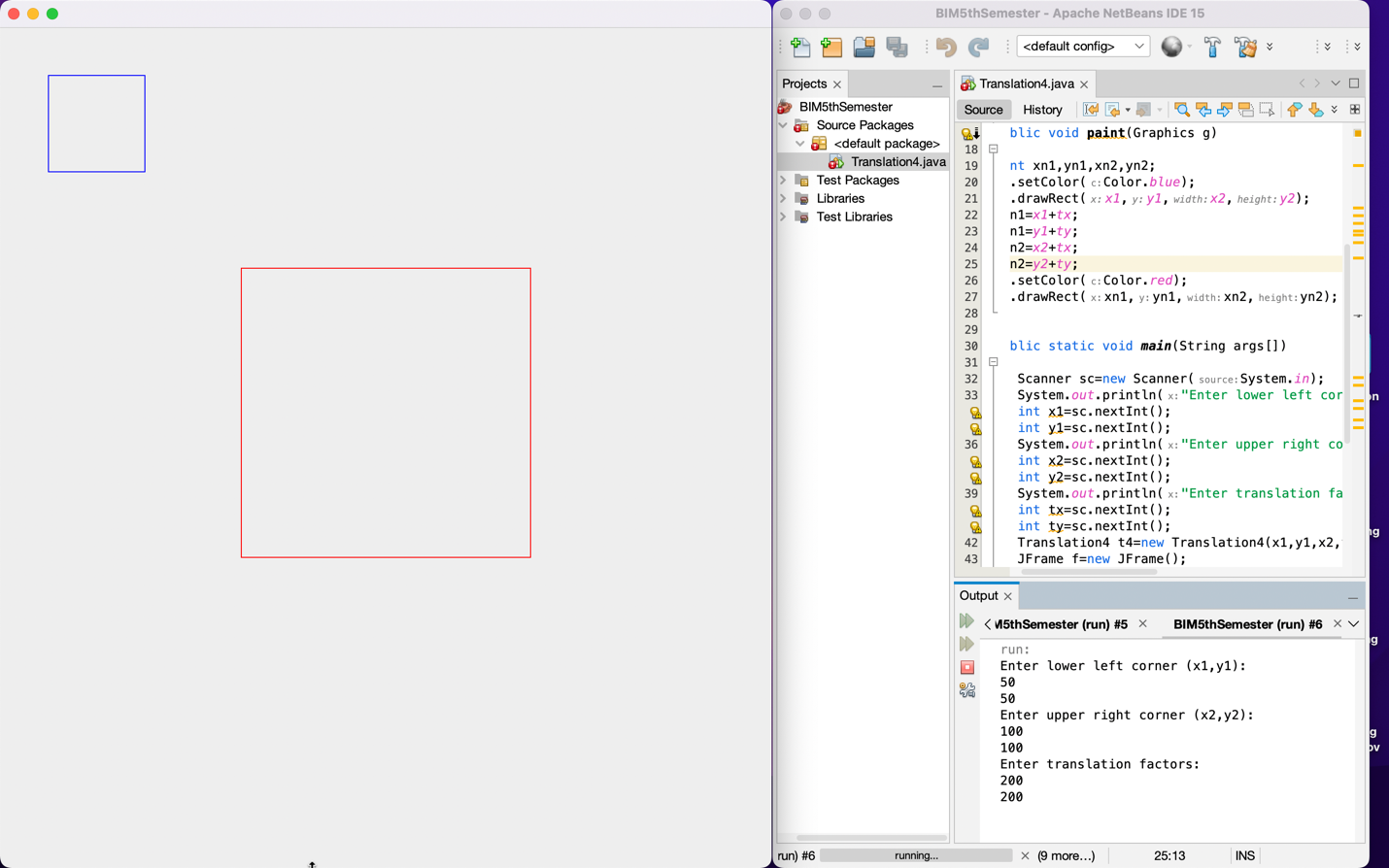
f.setSize(800,800);

f.setVisible(true);

}

}

**OUTPUT:**



**CONCLUSION:**

We have translated a rectangle from one position to another by the help of translation factors.

**Lab 7 - Scaling About Origin**

**Objective:** To alter the size of an object by multiplying the coordinate values (x, y) according to Scaling algorithm.

**Tools:** NetBeans

**Algorithm:**

1. Start
2. Input end points (x1, y1) and (x2, y2).​
3. Construct a 2D object (use drawRect( )) e.g. (x,y)
4. Get the scaling value Sx, Sy
5. Resize the object with Sx, Sy (x’= x\*Sx, y’= y\*Sy)
6. Plot (x’, y’)
7. Stop.

CODE:

import java.awt.\*;

import javax.swing.JFrame;

import java.util.Scanner;

import java.lang.\*;

public class Scaling00 extends Canvas

{

static int x1,y1,x2,y2,sx,sy;

Scaling00(int x1,int y1,int x2,int y2,int sx,int sy)

{

this.x1=x1;

this.y1=y1;

this.x2=x2;

this.y2=y2;

this.sx=sx;

this.sy=sy;

}

public void paint(Graphics g)

{

int xn1,yn1,xn2,yn2;

g.setColor(Color.blue);

g.drawRect(x1,y1,x2,y2);

xn1 = x1\*sx;

yn1 = y1\*sy;

xn2 = x2\*sx;

yn2 = y2\*sy;

g.setColor(Color.red);

g.drawRect(xn1,yn1,xn2,yn2);

}

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter lower left corner (x1,y1): ");

int x1=sc.nextInt();

int y1=sc.nextInt();

System.out.println("Enter upper right corner (x2,y2): ");

int x2=sc.nextInt();

int y2=sc.nextInt();

System.out.println("Enter scaling factors: ");

int sx=sc.nextInt();

int sy=sc.nextInt();

Scaling00 s0=new Scaling00(x1,y1,x2,y2,sx,sy);

JFrame f=new JFrame();

f.add(s0);

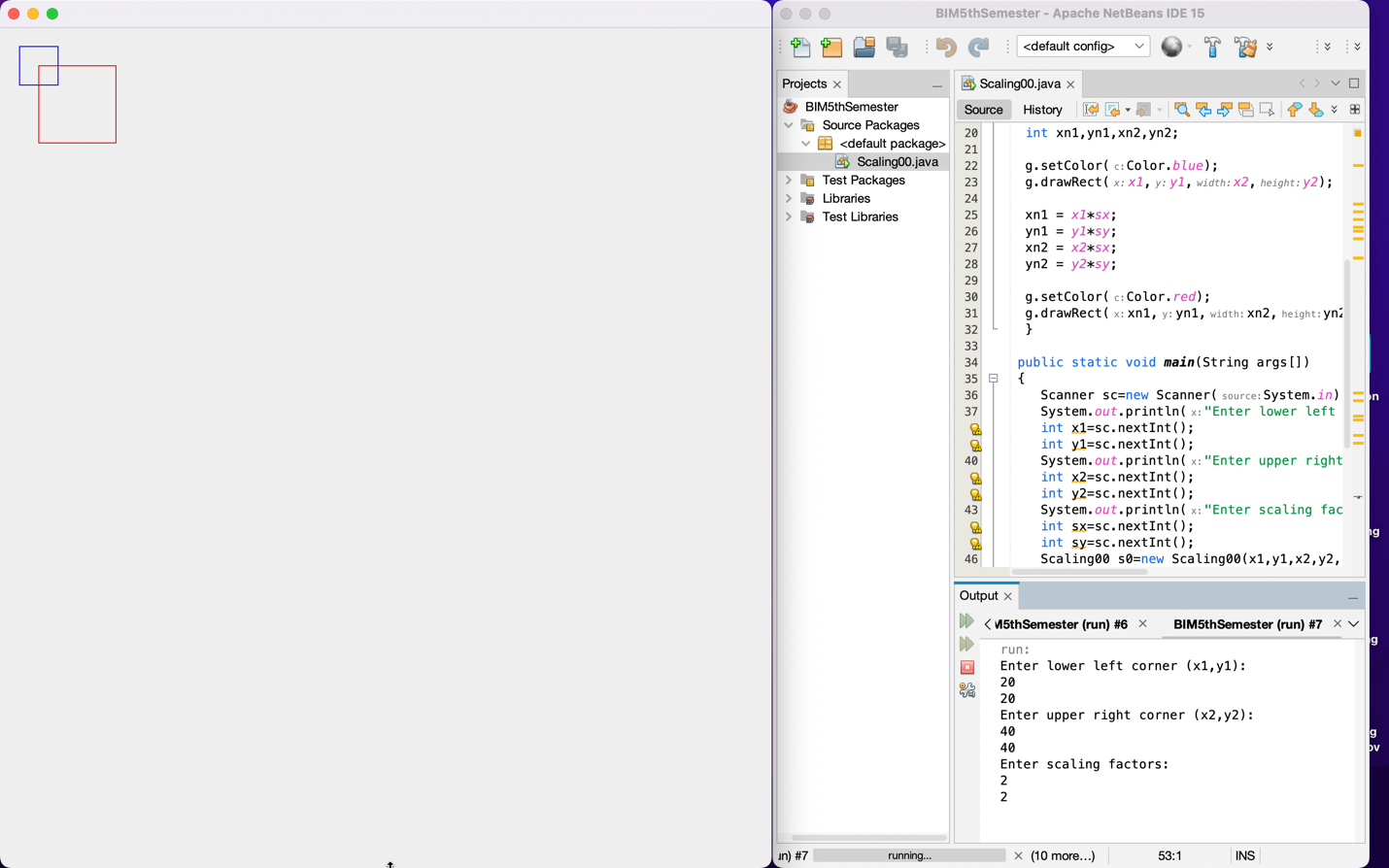
f.setSize(800,800);

f.setVisible(true);

}

}

**OUTPUT:**



**Conclusion:**

We have scaled a square by the help of scaling factors.

**Lab 8 - Fixed-Point Scaling**

**Objective:** To alter the size of an object by multiplying the coordinate values (x, y) with fixed point according to Scaling algorithm.

**Tools:** NetBeans

**Algorithm:**

1. Start
2. Input end points (x1, y1) and (x2, y2).​
3. Construct a 2D object (use drawRect( )) e.g. (x,y)
4. Get the scaling value Sx, Sy
5. Get the fixed point (xf, yf)
6. Translate the object to origin
7. Resize the object with Sx, Sy (x’= x\*Sx, y’= y\*Sy)
8. Re-translate the object
9. Plot (x’, y’)
10. Stop.

CODE:

import java.awt.\*;

import javax.swing.JFrame;

import java.util.Scanner;

import java.lang.\*;

public class ScalingXY extends Canvas

{

static int x1,y1,x2,y2,sx,sy,xf,yf;

ScalingXY(int x1,int y1,int x2,int y2,int sx,int sy,int xf,int yf)

{

this.x1=x1;

this.y1=y1;

this.x2=x2;

this.y2=y2;

this.sx=sx;

this.sy=sy;

this.xf=xf;

this.yf=yf;

}

public void paint(Graphics g)

{

int xn1,yn1,xn2,yn2;

g.setColor(Color.blue);

g.drawRect(x1,y1,x2,y2);

xn1=x1-xf;

yn1=y1-yf;

xn2=x2-xf;

yn2=y2-yf;

xn1 = xn1\*sx;

yn1 = yn1\*sy;

xn2 = xn2\*sx;

yn2 = yn2\*sy;

xn1=xn1+xf;

yn1=yn1+yf;

xn2=xn2+xf;

yn2=yn2+yf;

g.setColor(Color.red);

g.drawRect(xn1,yn1,xn2,yn2);

}

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter lower left corner (x1,y1): ");

int x1=sc.nextInt();

int y1=sc.nextInt();

System.out.println("Enter upper right corner (x2,y2): ");

int x2=sc.nextInt();

int y2=sc.nextInt();

System.out.println("Enter scaling factors: ");

int sx=sc.nextInt();

int sy=sc.nextInt();

System.out.println("Enter the fixed-point (xf,yf): ");

int xf=sc.nextInt();

int yf=sc.nextInt();

ScalingXY sxy=new ScalingXY(x1,y1,x2,y2,sx,sy,xf,yf);

JFrame f=new JFrame();

f.add(sxy);

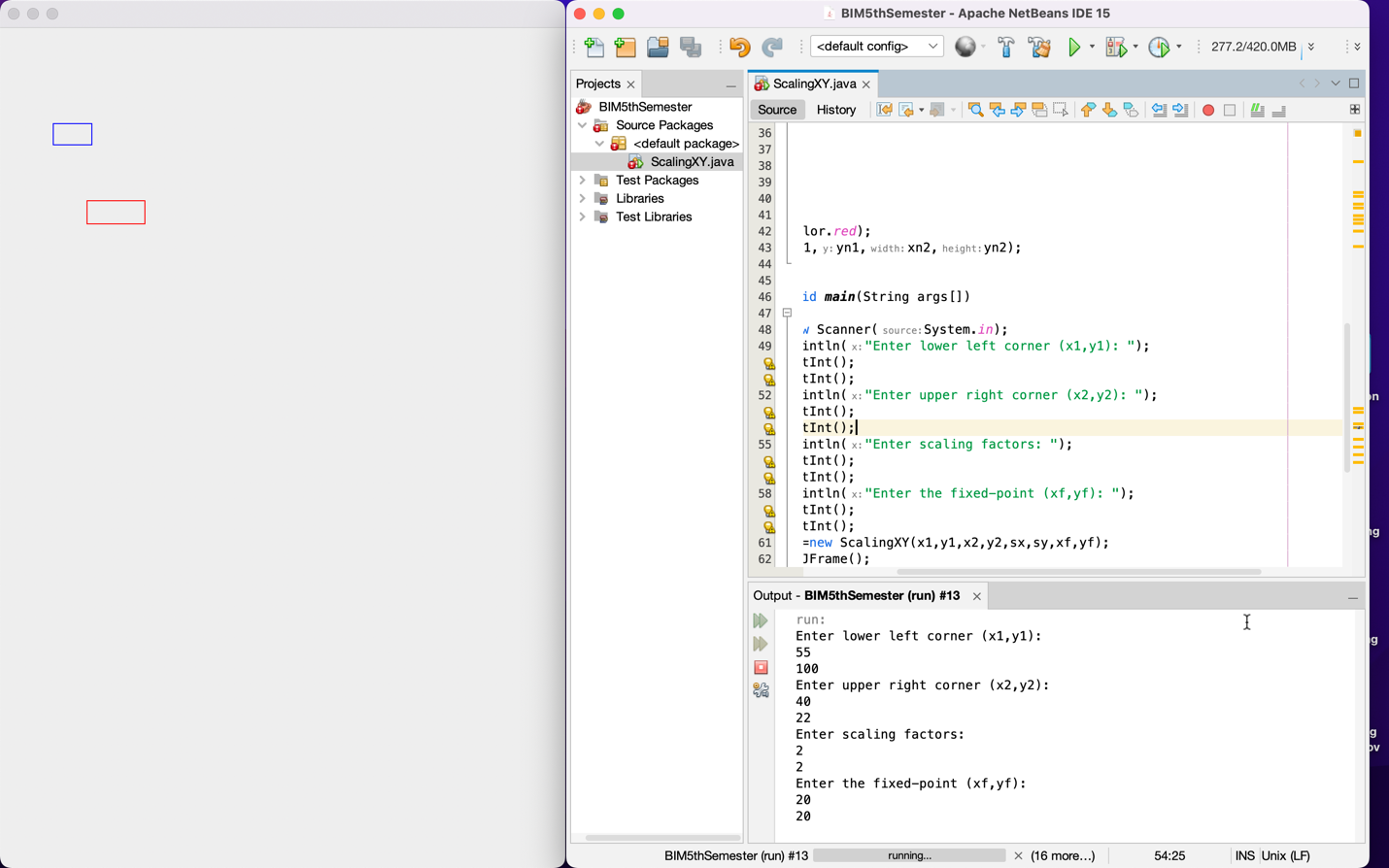
f.setSize(800,800);

f.setVisible(true);

}

}

# OUTPUT:



**Conclusion:**

We have scaled a shape by the help of scaling factors through an arbitrary point.

# Lab 9 - Rotation About Origin

**Objective:** To reposition an object along a circular path (anti-clockwise) in the xy plane according to Rotation algorithm.

**Tools:** NetBeans

**Algorithm:**

1. Start
2. Input end points (x1, y1) and (x2, y2).​
3. Construct a 2D object (use drawline( )) e.g. (x,y)
4. Get the rotation angle
5. Rotate the object by the angle θ

x’= xcosθ - ysinθ

y’= xsinθ + ycosθ

1. Plot (x’, y’)
2. stop

# CODE:

import java.awt.\*;

import javax.swing.JFrame;

import java.util.Scanner;

import java.lang.\*;

public class Rotation00 extends Canvas

{

static int x1,y1,x2,y2,angle;

Rotation00(int x1,int y1,int x2,int y2,int angle)

{

this.x1=x1;

this.y1=y1;

this.x2=x2;

this.y2=y2;

this.angle=angle;

}

public void paint(Graphics g)

{

double xn1,yn1,xn2,yn2,ang;

g.setColor(Color.blue);

g.drawLine(x1,y1,x2,y2);

ang=Math.toRadians(angle);

xn1 = x1\*Math.cos(ang) - y1\*Math.sin(ang);

yn1 = x1\*Math.sin(ang) + y1\*Math.cos(ang);

xn2 = x2\*Math.cos(ang) - y2\*Math.sin(ang);

yn2 = x2\*Math.sin(ang) + y2\*Math.cos(ang);

g.setColor(Color.red);

g.drawLine((int)Math.round(xn1),(int)Math.round(yn1),(int)Math.round(xn2),(int)Math.round(yn2));

}

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter first point (x1,y1): ");

int x1=sc.nextInt();

int y1=sc.nextInt();

System.out.println("Enter last point (x2,y2): ");

int x2=sc.nextInt();

int y2=sc.nextInt();

System.out.println("Enter rotation angle: ");

int angle=sc.nextInt();

Rotation00 r0=new Rotation00(x1,y1,x2,y2,angle);

JFrame f=new JFrame();

f.add(r0);

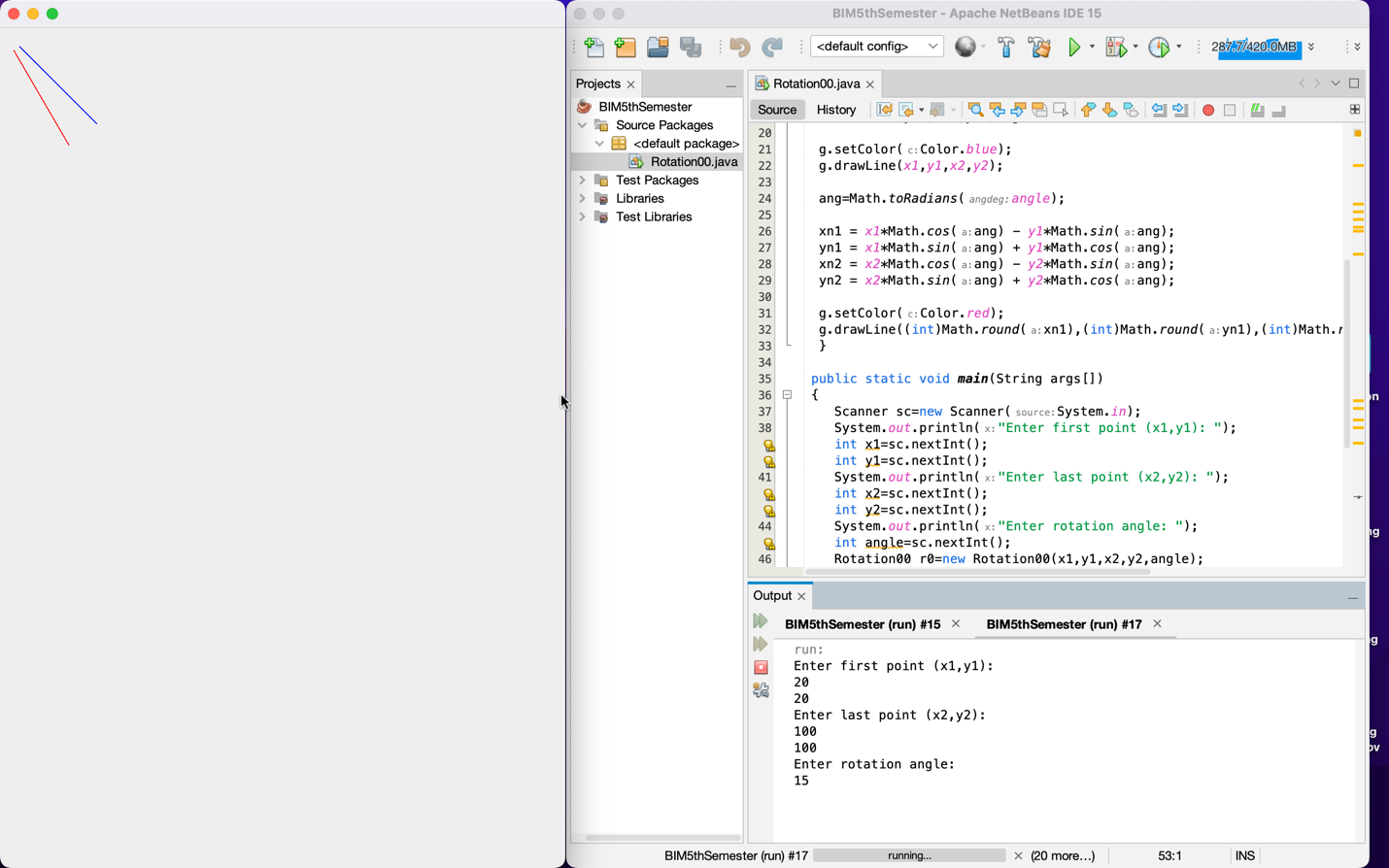
f.setSize(800,800);

f.setVisible(true);

}

}

# OUTPUT:



**Conclusion:**

We have rotated a line about origin.

# Lab 10 - Rotation About Arbitrary Point

**Objective:** To reposition an object along a circular path (anti-clockwise) in the xy plane with fixed point according to Rotation algorithm.

**Tools:** NetBeans

**Algorithm:**

1. Start
2. Input end points (x1, y1) and (x2, y2).​
3. Construct a 2D object (use drawline( )) e.g. (x,y)
4. Get the rotation angle
5. Get the fixed points (xf,yf)
6. Translate the object to origin
7. Rotate the object by the angle θ

x’= xcosθ - ysinθ

y’= xsinθ + ycosθ

1. Re-translate the object
2. Plot (x’, y’)
3. Stop.

# CODE:

import java.awt.\*;

import javax.swing.JFrame;

import java.util.Scanner;

import java.lang.\*;

public class RotationXY extends Canvas

{

static int x1,y1,x2,y2,xf,yf,angle;

RotationXY(int x1,int y1,int x2,int y2,int xf,int yf,int angle)

{

this.x1=x1;

this.y1=y1;

this.x2=x2;

this.y2=y2;

this.xf=xf;

this.yf=yf;

this.angle=angle;

}

public void paint(Graphics g)

{

double xn1,yn1,xn2,yn2,ang;

g.setColor(Color.blue);

g.drawLine(x1,y1,x2,y2);

ang=Math.toRadians(angle);

xn1=x1-xf;

yn1=y1-yf;

xn2=x2-xf;

yn2=y2-yf;

xn1 = xn1\*Math.cos(ang) - yn1\*Math.sin(ang);

yn1 = xn1\*Math.sin(ang) + yn1\*Math.cos(ang);

xn2 = xn2\*Math.cos(ang) - yn2\*Math.sin(ang);

yn2 = xn2\*Math.sin(ang) + yn2\*Math.cos(ang);

xn1=xn1+xf;

yn1=yn1+yf;

xn2=xn2+xf;

yn2=yn2+yf;

g.setColor(Color.red);

g.drawLine((int)Math.round(xn1),(int)Math.round(yn1),(int)Math.round(xn2),(int)Math.round(yn2));

}

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter first point (x1,y1): ");

int x1=sc.nextInt();

int y1=sc.nextInt();

System.out.println("Enter last point (x2,y2): ");

int x2=sc.nextInt();

int y2=sc.nextInt();

System.out.println("Enter fixed point (xf,yf): ");

int xf=sc.nextInt();

int yf=sc.nextInt();

System.out.println("Enter rotation angle: ");

int angle=sc.nextInt();

RotationXY rxy=new RotationXY(x1,y1,x2,y2,xf,yf,angle);

JFrame f=new JFrame();

f.add(rxy);

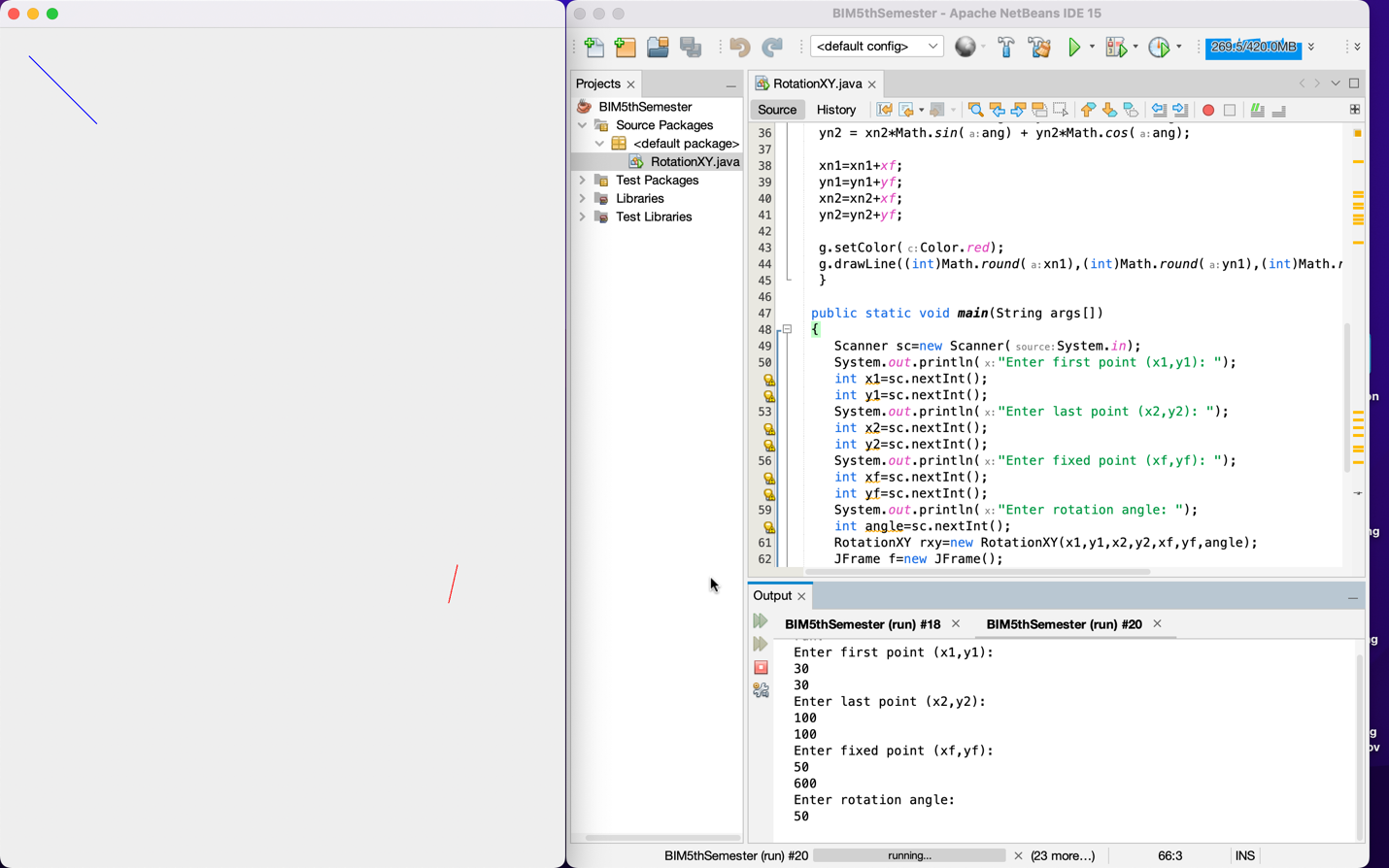
f.setSize(800,800);

f.setVisible(true);

}

}

**OUTPUT:**



**Conclusion:**

We have rotated a line about arbitrary point.