FunctionPoint Value:

Source code:

```
import java.io.*;
import java.lang.*;
class FPValue
{
        int count[] = new int[5];
        int avg[] = new int[5];
        int ct[] = new int[5];
        int i;
        int ctotal;
        int fi;
        float fp;
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        FPValue()
        {
                System.out.println("\n\t Function Point(FP) Value : ");
                System.out.println("\n\t----------------------------");
                ctotal=0;
                fi=0;
                fp=0;
        }
        void input() throws IOException
        {
                for(i=1;i<=5;i++)
                {
                        if(i==1)
                        {
                                System.out.println("\nEnter numbr of external inputs : ");
                                count[i] = Integer.parseInt(br.readLine());
                        }
                        else if(i==2)
```

```
                {
                        System.out.println("\nEnter numbr of external outputs : ");
                        count[i] = Integer.parseInt(br.readLine());
                }
                else if(i==3)
                {
                        System.out.println("\nEnter numbr of external inquiries : ");
                        count[i] = Integer.parseInt(br.readLine());
                }
                else if(i==4)
                {
                        System.out.println("\nEnter numbr of internal logical files : ");
                        count[i] = Integer.parseInt(br.readLine());
                }
                else if(i==5)
                {
                        System.out.println("\nEnter numbr of external interfaces : ");
                        count[i] = Integer.parseInt(br.readLine());
                }
        }
        System.out.println("\n\n Enter average complexity adjustments values : \n");
        for(i=1;i<=5;i++)
        {
                avg[i] = Integer.parseInt(br.readLine());
        }
        System.out.println("\n Enter Sum of value adjustments factors(fi) : ");
        fi = Integer.parseInt(br.readLine());
}
void process() throws IOException
{
        for(int j =1;j<=5;j++)
        {
```

```
                ct[j] = count[j] * avg[j];
                ctotal = ctotal + ct[j];
            }
            fp = ctotal * (0.65 + (0.01 *fi));
        }
        void output() throws IOException
        {
            System.out.println("\n\n Count total is : " +ctotal);
            System.out.println("\n Sum of fi is : " +fi);
            System.out.println("\n\n\t FP = Count total * (0.65 + (0.01 * sum of fi)) ");
            System.out.println("\n Functional Point (FP) Value is : "+fp);
        }
}
class FPVsiree
{
    public static void main(String args[]) throws IOException
    {
        FPValue fp;
        fp.input();
        fp.process();
        fp.output();
    }
}
```

COCOMO:

Source code:

```
import java.io.*;

import java.util.*;
class COCOMO
{
        public static void main(String args[])
        {
                Scanner obj = new Scanner (System.in);
                int s,r,c,cw1,cw2,cw3,prod;
                float EE,NOP;
                System.out.println("Enter No of Screens:");
                s = obj.nextInt();
                System.out.println("Enter Complexity weight:");
                cw1 = obj.nextInt();
                System.out.println("Enter No of Reports:");
                r = obj.nextInt();
                System.out.println("Enter Complexity weight:");
                cw2 = obj.nextInt();
                System.out.println("Enter No of Components:");
                c = obj.nextInt();
                System.out.println("Enter Complexity weight:");
                cw3 = obj.nextInt();
                System.out.println("Enter vlaue of prod:");
                prod = obj.nextInt();
                NOP = (float) ((s*cw1)+(r*cw2)+(c*cw3));
                System.out.println("NOP = " + NOP);
                EE = (float) NOP/prod;
                System.out.println("The Estimated effort for simple ATM using Cocomo II model is : " +EE);
        }
}
```

CYCLOMATIC:

Source code:

```
import java.io.*;
import java.util.*;
class Cyclomatic
{
	public static void main(String args[])
	{
		int e,n,v;
		Scanner obj = new Scanner(System.in);
		System.out.println("Enter the no of Edges : ");
		e = obj.nextInt();
		System.out.println("Enter the no of nodes : ");
		n = obj.nextInt();
		v = e-n+2;
		System.out.println("The Cyclomatic complexity of a graph is : " +v);
	}
}
```

Source code:

```
import java.io.*;
import java.util.*;
class SMI
{
	public static void main(String args[])
	{
		Scanner obj = new Scanner(System.in);
		int MT,FA,FC,FD;
		float SMI;
		System.out.println("Enter MT:");
		MT = obj.nextInt();
		System.out.println("Enter FA:");
		FA = obj.nextInt();
		System.out.println("Enter FC:");
		FC = obj.nextInt();
		System.out.println("Enter FD:");
		FD = obj.nextInt();
		SMI = (float) ( MT - (FA+FC+FD))/MT;
		System.out.println("The resultant SMI is: " +SMI);
	}
}
```

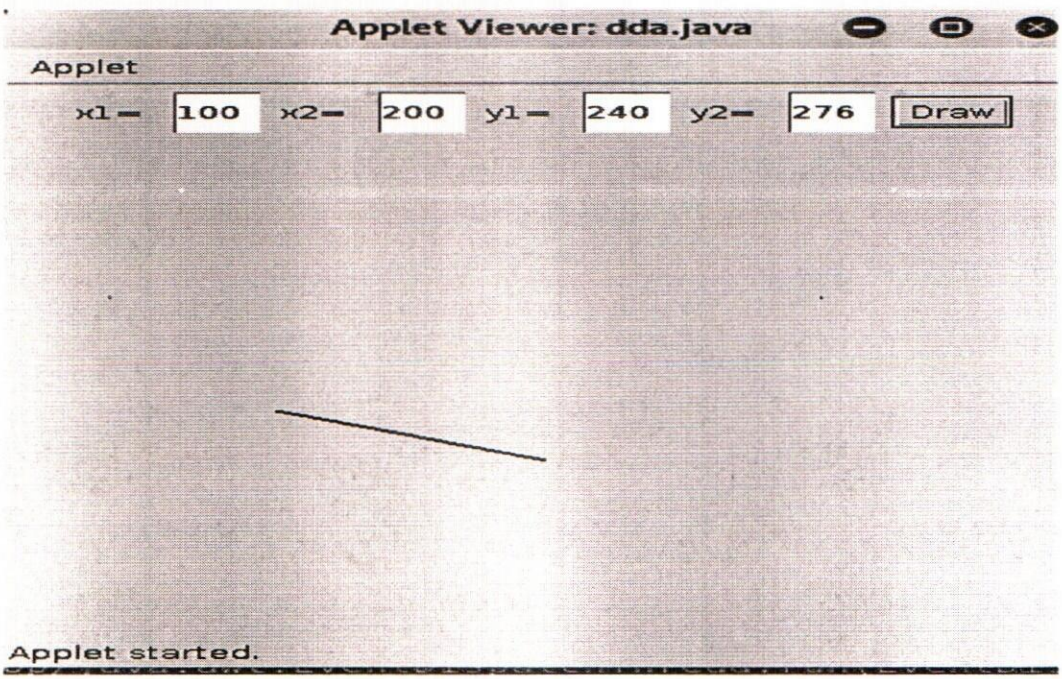
DDA:

Source code:

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
public class dda extends Applet implements ActionListener
{
int x1,x2,y1,y2;
Label L1=new Label("x1=");
Label L2=new Label("x2=");
Label L3=new Label("y1=");
Label L4=new Label("y2=");
TextField t1=new TextField(3);
TextField t2=new TextField(3);
TextField t3=new TextField(3);
TextField t4=new TextField(3);
Button b=new Button("Draw");
public void init(){
add(L1);
add(t1);
add(L2);
add(t2);
add(L3);
add(t3);
add(L4);
add(t4);
add(b);
b.addActionListener(this);
}
public void paint(Graphics g) {
float xinc,yinc;
int step;
int x1=Integer.parseInt(t1.getText());
int x2=Integer.parseInt(t2.getText());
int y1=Integer.parseInt(t3.getText());
int y2=Integer.parseInt(t4.getText());
int dx=x2-x1;
```

```
int dy=y2-y1;
if(Math.abs(dx)>Math.abs(dy))
step=Math.abs(dx);
else step=Math.abs(dy);
xinc=(float)dx/step;
yinc=(float)dy/step;
float x=x1;
float y=y1;
g.drawString(".",(int)x,(int)y);
for(int i=1;i<step;i++) {
x=x+xinc;
y=y+yinc;
g.drawString(".",(int)x,(int)y);
}}
public void actionPerformed(ActionEvent a)
{
repaint();
}}
/*<applet code=dda.class width=500 height=500>
</applet>*/
```

Output:



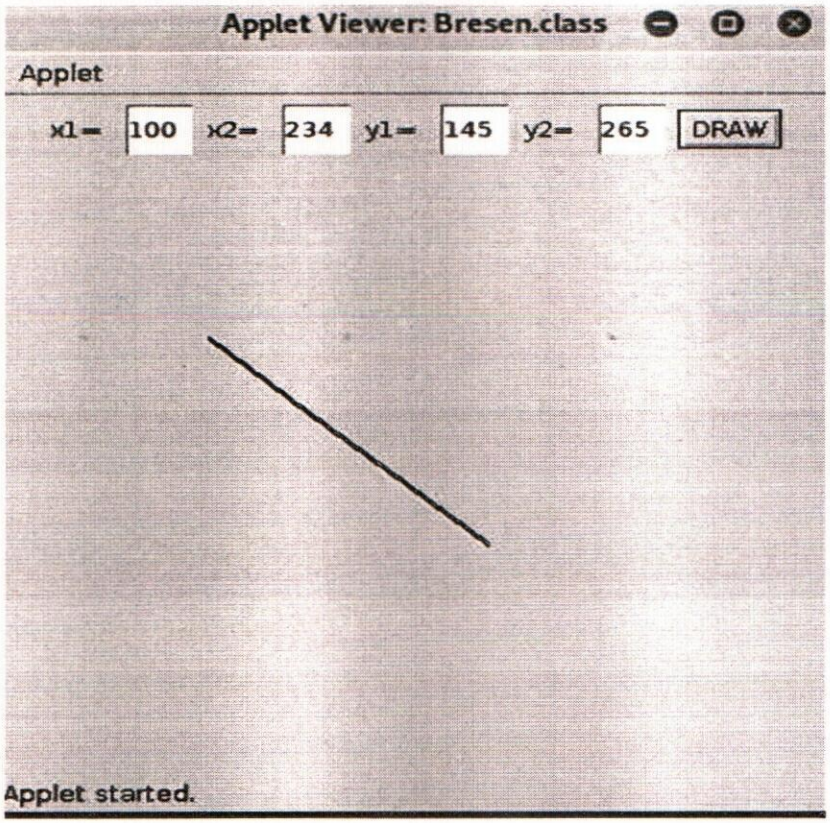
Bresanhams line Drawing Algorithm:

Source Code:

```
import java.lang.*;
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class bresan extends Applet implements ActionListener    {
Label l1=new Label("x1=");
Label l2=new Label("x2=");
Label l3=new Label("y1=");
Label l4=new Label("y2=");
TextField t1=new TextField(3);
TextField t2=new TextField(3);
TextField t3=new TextField(3);
TextField t4=new TextField(3);
Button b=new Button("DRAW");
public void init(){
add(l1);
add(t1);
add(l2);
add(t2);
add(l3);
add(t3);
add(l4);
add(t4);
add(b);
b.addActionListener(this);
}
public void paint(Graphics g) {
int x1=Integer.parseInt(t1.getText());
int x2=Integer.parseInt(t2.getText());
int y1=Integer.parseInt(t3.getText());
int y2=Integer.parseInt(t4.getText());
int dx=x2-x1;
int dy=y2-y1;
int count=dx;
int p=2*(dy-dx);
```

```
int Xend,X,Y;
if(x1>x2) {
X=x2;
Y=y2;
Xend=x1;
}
else {
X=x1;
Y=y1;
Xend=x2;
}
g.drawString(",",X,Y);
while(count>0) {
X+=1;
if(p<0) {
p=p+2*dy;
}
else {
Y+=1;
p=p+2*(dy-dx);
}
g.drawString(",",X,Y);
count--;
}}
public void actionPerformed(ActionEvent a)
{
repaint();
}}
/*<applet code= bresan.class width=400 height=400>
</applet>*/
```

Output:


Applet Viewer: Bresen.class
Applet
x1= 100 x2= 234 y1= 145 y2= 265 DRAW
Applet started.

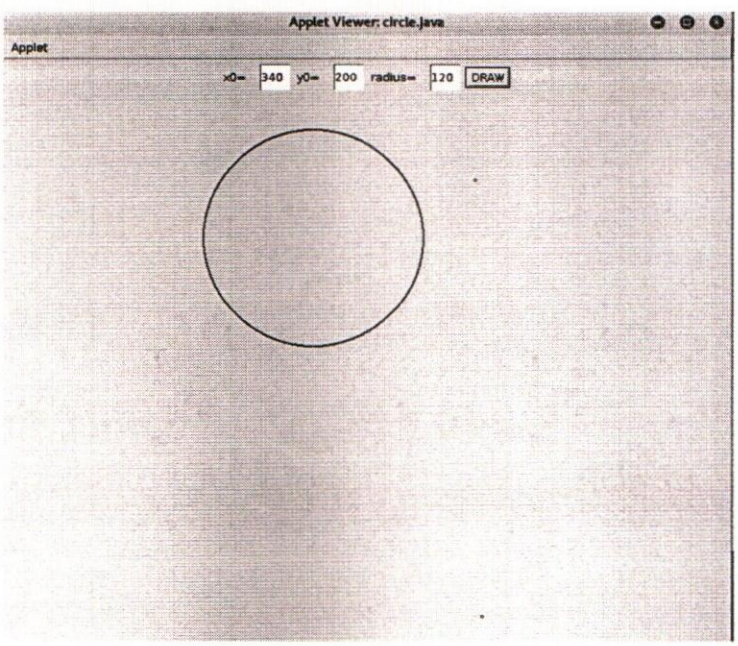
Mid-point Circle:

Source Code:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class circle extends Applet implements ActionListener
{
Label l1=new Label("x0= ");
Label l2=new Label("x0= ");
Label l3=new Label("radius= ");
TextField t1=new TextField(3);
TextField t2=new TextField(3);
TextField t3=new TextField(3);
Button bw=new Button("DRAW");
public void init() {
add(l1);
add(t1);
add(l2);
add(t2);
add(l3);
add(t3);
add(bw);
bw.addActionListener(this);
}
public void paint(Graphics g) {
int xc=Integer.parseInt(t1.getText());
int yc=Integer.parseInt(t2.getText());
int rad=Integer.parseInt(t3.getText());
int x=0;
int y=rad;
int p=5/4-rad;
while(x<y){
if(p<0){
x++;
```

```
p=p+2*x+2+1;
}
else{
x++;
y--;
p=p+2*x+2+1-2*y+2;
}
g.drawString("." ,xc+x,yc+y);
g.drawString(".",xc+y,yc-x);
g.drawString(".",xc-y,yc+x);
g.drawString(".",xc-x,yc+y);
g.drawString(".",xc+x,yc-y);
g.drawString("." ,xc+y,yc+x);
g.drawString(".",xc-y,yc-x);
g.drawString(".",xc-x,yc-y);
}}
public void actionPerformed(ActionEvent aw){ repaint();
}}
/*<applet code=circle.class width=400 height=400>
</applet>*/-
```

Output:



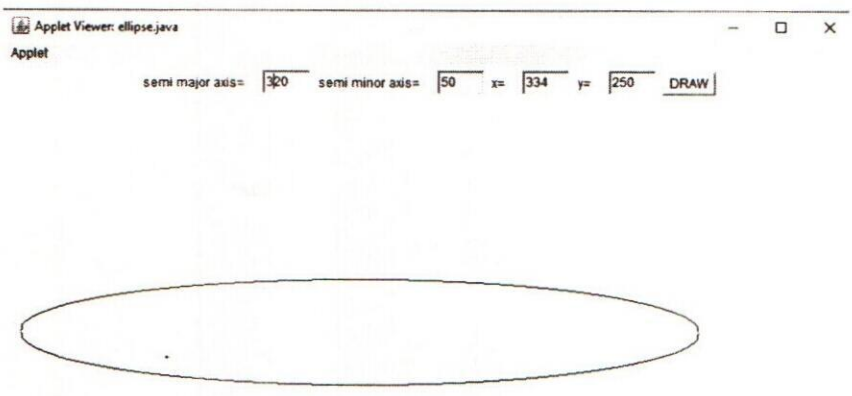
Mid-point Ellipse

Source Code:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class ellipse2 extends Applet implements ActionListener
{
float x,y,xc,yc,rx,ry,p1,p2;
Label l1=new Label("semi major axis= ");
Label l2=new Label("semi minor axis=");
Label l3=new Label("x=");
Label l4=new Label("y= ");
TextField t1=new TextField(3);
TextField t2=new TextField(3);
TextField t3=new TextField(3);
TextField t4=new TextField(3);
Button bw=new Button("DRAW");
public void init() {
add(l1);
add(t1);
add(l2);
add(t2);
add(l3);
add(t3);
add(l4);
add(t4);
add(bw);
bw.addActionListener(this);
}
public void paint(Graphics g){
rx=Integer.parseInt(t1.getText());
ry=Integer.parseInt(t2.getText());
xc=Integer.parseInt(t3.getText());
yc=Integer.parseInt(t4.getText());
x=0;
y=ry;
p1=(ry*ry)-(rx*rx*y)+(rx*rx/4);
```

```
do{
g.drawString(".",Math.round(xc-x),Math.round(yc+y));
g.drawString(".",Math.round(xc-x),Math.round(yc-y));
g.drawString(".",Math.round(xc+x),Math.round(yc+y));
g.drawString(".",Math.round(xc+x),Math.round(yc-y));
x++;
if(p1<0) p1=p1+(2*ry*ry*x)+(ry*ry);
else{
y-=1;
p1=p1+(2*ry*ry*x)-(2*rx*rx*y)+(ry*ry);
}}
while((2*ry*ry*x)<(2*rx*rx*y));
p2=(ry*ry*(x+1/2)*(x+1/2))+(rx*rx*(y-1)*(y-1))-(ry*ry*rx*rx);
while(y>0) {
g.drawString(".",Math.round(xc-x),Math.round(yc+y));
g.drawString(".",Math.round(xc-x),Math.round(yc-y));
g.drawString(".",Math.round(xc+x),Math.round(yc+y));
g.drawString(".",Math.round(xc+x),Math.round(yc-y));
y--;
if(p2>0)
p2=p2-(2*rx*rx*y)+(ry*ry);
else{
x+=1;
p2=p2+(2*ry*ry*x)+(2*rx*rx*y)+(ry*ry);}
}}
public void actionPerformed(ActionEvent aw) { repaint();
}}
/*<applet code=ellipse2.class width=400 height=400>
</applet>*/
```

Output:



2D-Basic Transformations:

Source Code:

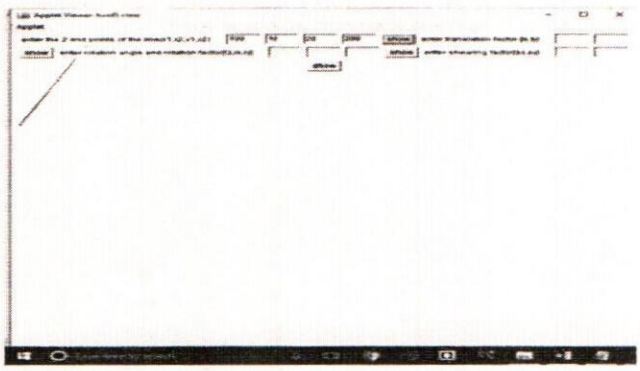
```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class twoD extends Applet implements ActionListener{
Label p1=new Label("enter the 2 end points of the line(x1,x2,y1,y2):");
Label p2= new Label("enter translation factor.[tx,ty]:");
Label p3= new Label("enter rotation angle and rotation factor[Q,rx,ry]:");
Label p4=new Label("enter shearing factor[sx,sy]:");
TextField t1= new TextField(3);
TextField t2= new TextField(3);
TextField t3= new TextField(3);
TextField t4= new TextField(3);
TextField t5= new TextField(3);
TextField t6= new TextField(3);
TextField t7= new TextField(3);
TextField t8= new TextField(3);
TextField t9= new TextField(3);
TextField t10= new TextField(3);
TextField t11= new TextField(3);
Button b1=new Button("show");
Button b2=new Button("show");
Button b3=new Button("show");
Button b4=new Button("show");
int x1,x2,y1,y2,tx,ty,Q,rx,ry,sx,sy,nx1,nx2,ny1,ny2,t=0;
public void init() {
add(p1);
add(t1);
add(t2);
add(t3);
add(t4);
add(b1);
add(p2);
add(t5);
add(t6);
add(b2);
```

```
add(p3);
add(t7);
add(t8);
add(t9);
add(b3);
add(p4);
add(t10);
add(t11);
add(b4);
b1.addActionListener(this);
b2.addActionListener(this);
b3.addActionListener(this);
b4.addActionListener(this);
}
public void paint(Graphics g) {
x1=Integer.parseInt(t1.getText());
x2=Integer.parseInt(t2.getText());
y1=Integer.parseInt(t3.getText());
y2=Integer.parseInt(t4.getText());
g.drawLine(x1,y1,x2,y2);
if(t==1)
g.drawLine(nx1,ny1,nx2,ny2);
}
public void actionPerformed(ActionEvent a) {
if(a.getSource()==b1) {
showStatus("original");
repaint();
}
if(a.getSource()==b2) {
tx=Integer.parseInt(t5.getText());
ty=Integer.parseInt(t6.getText());
showStatus("aftertranslation");
 t=1;
nx1=x1+tx;
ny1=y1+ty;
nx2=x2+tx;
ny2=y2+ty;
repaint();
```

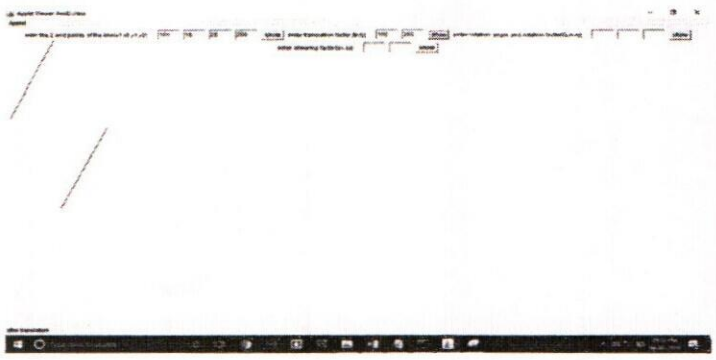
```
}
if(a.getSource()==b3) {
Q=Integer.parseInt(t7.getText());
rx=Integer.parseInt(t8.getText());
ry=Integer.parseInt(t9.getText());
showStatus("after rotation");
t=1;
double d=Math.toRadians(Q);
nx1=rx+(int)((x1-rx)*Math.cos(d))-(int)((y1-ry)*Math.sin(d));
nx2=rx+(int)((x2-rx)*Math.cos(d))-(int)((y2-ry)*Math.sin(d));
ny1=ry+(int)((x1-rx)*Math.sin(d))-(int)((y1-ry)*Math.cos(d));
ny2=ry+(int)((x1-rx)*Math.sin(d))-(int)((y1-ry)*Math.cos(d));
repaint();
}
if(a.getSource()==b4) {
sx=Integer.parseInt(t10.getText());
sy=Integer.parseInt(t11.getText());
showStatus("scaling");
t=1;
nx1=x1*sx;
nx2=x2*sx;
ny1=y1*sy;
ny2=y2*sy;
repaint();
}}}
/*<applet code=twoD.class width=400 height=400>
</applet>*/
```

Output:

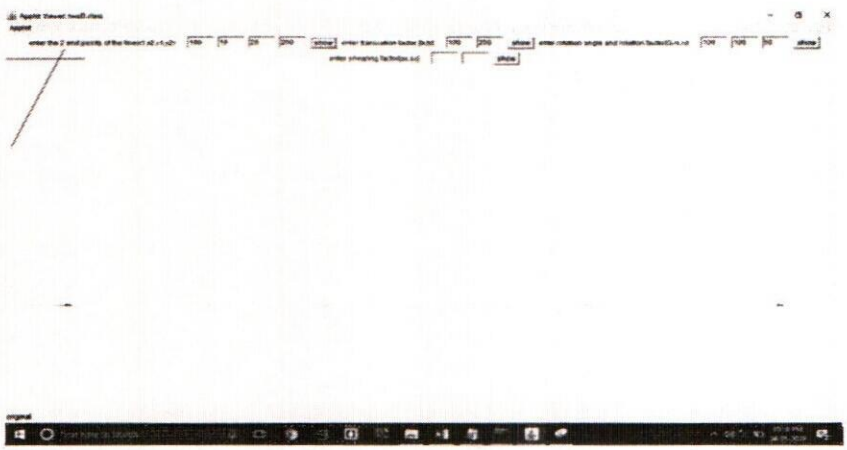
Initialline



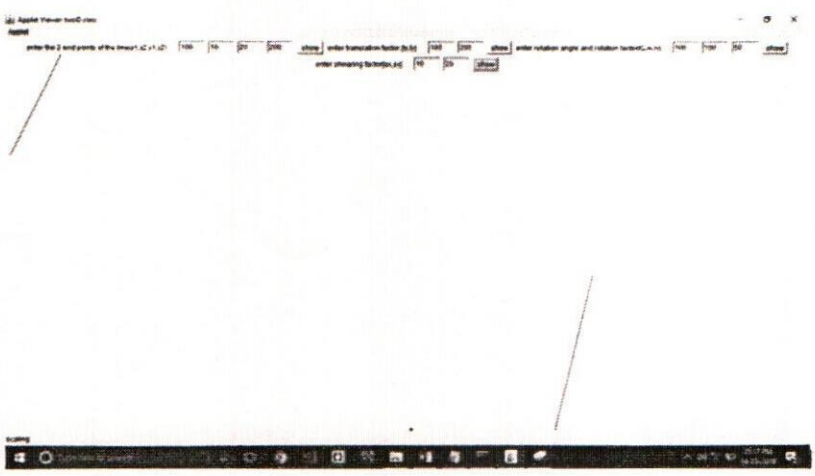
Translation



Scaling:



Shearing:



Line Clipping:

Source Code:

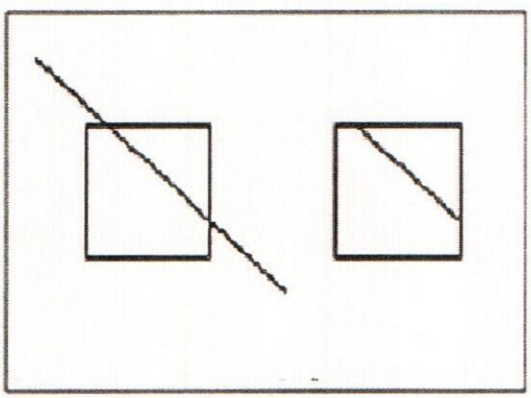
```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class lineclip extends Applet implements MouseListener,
MouseMotionListener,ActionListener {
Button b=new Button("clip");
Graphics key;
float x,y,xa,ya,xb,yb,dx,dy,m,p,xEnd;
public void init() {
add(b);
b.addActionListener(this);
addMouseListener(this);
addMouseMotionListener(this);
}
public void mouseEntered(MouseEvent me){}
public void mouseClicked(MouseEvent me){}
public void mouseExited(MouseEvent me){}
public void mouseMoved(MouseEvent me){}
public void mouseDragged(MouseEvent me){}
public void mousePressed(MouseEvent me) {
xa=me.getX();
ya=me.getY();
}
pubiic void mouseReleased(MouseEvent me) {
xb=me.getX();
yb=me.getY();
repaint();
}
public void paint(Graphics g) {
g.drawRect(200,200,200,200);
g.drawLine(Math.round(xa),Math.round(ya),Math.round(xb),Math.round(yb));
}
public void actionPerformed(ActionEvent a) {
key=getGraphics();
key.setColor(Color.white);
```

```
key.drawLine(Math.round(xa),Math.round(ya),Math.round(xb),Math.round(yb));
key.setColor(Color.black);
dx=Math.abs(xa-xb);
dy=Math.abs(ya-yb);
if(dx==0)
m=2;
else
m=(yb-ya)/(xb-xa);
if(m>=0&&m<=1)        {
p=2*dy-dx;
if(xa>xb){
x=xb;
y=yb;
xEnd=xa;
}
else{
x=xa;
y=ya;
xEnd=xb;
}
plot_clip(key); //comment
while(x<xEnd){
x=x+1;
if(p<0)
p=p+2*dy;
else{
y=y+1;
p=p+2*(dy-dx);
}
plot_clip(key);
}}
if(m<0&&m>=-1) {
p=2*dy-dx;
if(xa<xb){
x=xb;
y=yb;
xEnd=xa;
}
```

```
else{
x=xa;
y=ya;
xEnd=xb;
}
plot_clip(key);
while(x>xEnd) {
x=x-1;
if(p<0)
p=p+2*dy;
else{
y=y+1;
p=p+2*(dy-dx);
}
plot_clip(key);
}}
if(m<-1) {
p=2*dx-dy;
if(ya<yb) {
x=xb;
y=yb;
xEnd=ya;
}
else{
x=xa;
y=ya;
xEnd=yb;
}
plot_clip(key);
while(y>xEnd){
y=y-1;
if(p<0)
p=p+2*dx;
else{
x=x+1;
p=p+2*(dx-dy);
}
plot_clip(key);
```

```
}}}
public void plot_clip(Graphics key) {
if(x<200||x>400||y<200||y>400)
key.setColor(Color.white);
else
key.setColor(Color.black);
key.drawString(".",Math.round(x),Math.round(y));
}}
/*<applet code=lineclip.class width=600 height=600>
</applet>*/
```

Output:



Cohen Sutherland Line clipping Algorithm:

Source Code:

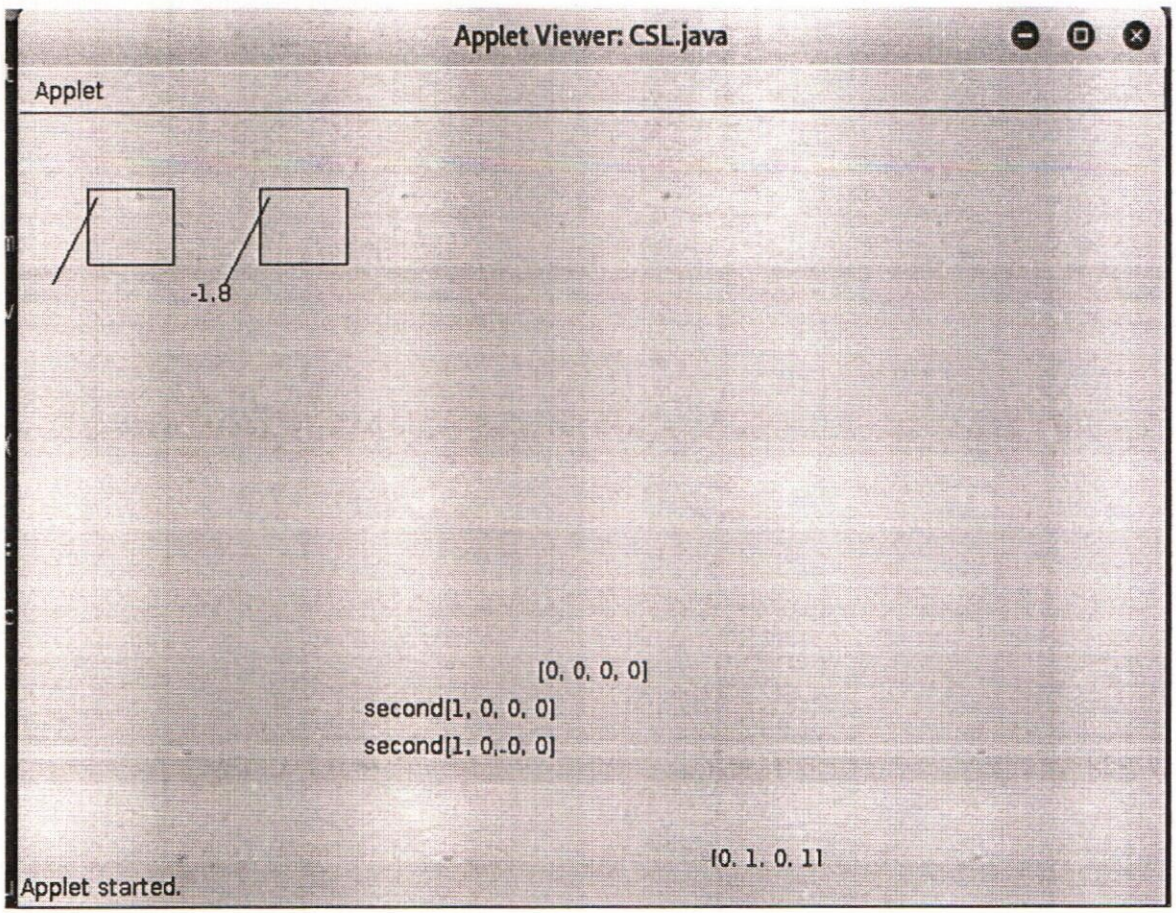
```
import java.util.*;
import java.lang.*;
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
public class CSL extends Applet                {
int Xmax=90,Ymax=80,Xmin=40,Ymin=40;
public int[] set(int x,int y) {
int a[]=new int [4];
if(x<Xmin)
a[3]=1;
else
a[3]=0;
if(x>Xmax)
a[2]=1;
else
a[2]=0;
if(y<Ymin)
a[0]=1;
else
a[0]=0;
if(y>Ymax)
a[1]=1;
else
a[1]=0;
return a;
}
boolean check(int a[]) {
for(int i=0;i<a.length;i++) if(a[i]==1)
return false;
return true;
}
int [] productXY(int i,int X1,int Y1,float m){
int a[]=new int[2];
float x=0,y=0;
```

```
switch(1) {
case 0:
x=Xmin;
y=Y1+m*(x-X1);
break;
case 1:
x=Xmax;
y=Y1+m*(x-X1);
break;
case 2:
y=Ymax;
x=X1+(y-Y1)/m;
break;
case 3:
y=Ymin;
x=X1+(y-Y1)/m;
break;
}
a[0]=(int)x;
a[1]=(int)y;
return a;
}
boolean doAnd(int a[],int b[]) {
for(int i=0; i<a.length;i++){
int k=a[i]&b[i];
if(k==1)
return false;
}
return true;
}
public void paint(Graphics g) {
g.drawRect(Xmin,Ymin,Xmax-Xmin,Ymax-Ymin);
g.drawRect(Xmin+100,Ymin,Xmax-Xmin,Ymax-Ymin);
int a[][]=new int[2][4];
int b[][]=new int[2][4];
int c[]=new int [2];
int c1=20;
int X1=45,Y1=45,X2=20,Y2=90;
```

```
float m=(Y2-Y1)*1.0f/(X2-X1);
g.drawString(m+" ",100,100);
g.drawLine(X1,Y1,X2,Y2); a[0]=set(X1,Y1);
a[1]=set(X2,Y2);
g.drawString(Arrays.toString(a[0]),300,300);
g.drawString(Arrays.toString(a[1]),400,400);
if(check(a[0])&& check(a[1])){
g.drawLine(X1,Y1,X2,Y2);
}
else{
if(doAnd(a[0],b[0])){
for(int i=a[0].length-1;i>=0;i--){
if(a[0][i]==1){
c=productXY(a[0].length-1-i,X1,Y1,m);
b[0]=set(c[0],c[1]);
g.drawString("first"+Arrays.toString(b[0]),200,300+c1);
if(check(b[0])){
X1=c[0];
Y1=c[1];
break;
}
c1+=20;
}}
for(int i=a[0].length-1;i>=0;i--){
if(a[1][i]==1){
c=productXY(a[0].length-1-i,X1,Y1,m);
b[1]=set(c[0],c[1]);
g.drawString("second"+Arrays.toString(b[1]),200,300+c1);
if(check(b[1])){
X2=c[0];
Y2=c[1];
break;
}
c1+=20;
}}
g.drawLine(X1+100,Y1,X2+100,Y2);
}}}}
/*<applet code=CSL.class width=400 height=400>
```

```
</applet>*/
```

Output:



Clock:

Source Code :

```
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
import java.awt.image.*;
import java.lang.*;
public class clock extends Applet implements ActionListener
{
int hour;
int min;
int second;
float h,m,s;
Label label1=new Label("hour ");
Label label2=new Label("minute ");
Label label3=new Label("second");
TextField text1=new TextField(3);
TextField text2=new TextField(3);
TextField text3=new TextField(3);
Button button=new Button("start");
public void init() {
add(label1);
add(text1);
add(label2);
add(text2);
add(label3);
add(text3);
add(button);
button.addActionListener(this);
second=-90;
min=-90;
hour=-90;
}
public void paint(Graphics g) {
g.setColor(Color.black);
int radius=91,j=0;
String num[]={"3","4","5","6","7","8","9","10","11","12","1","2"};
```

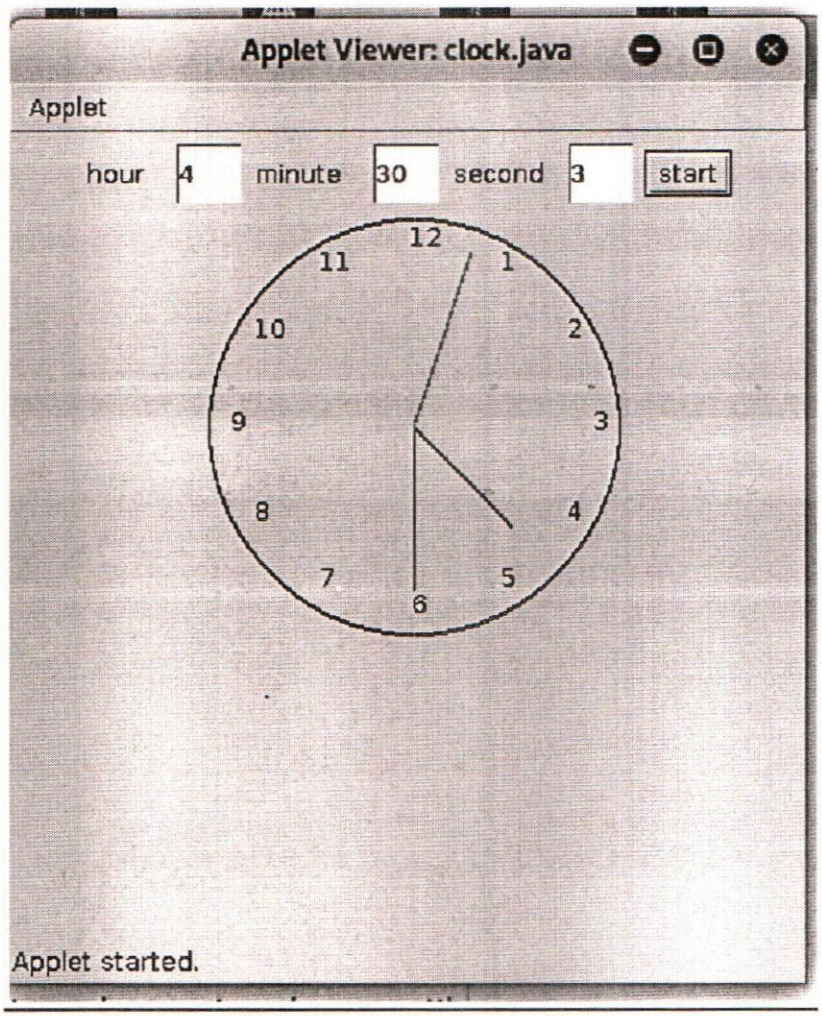
```
midcircle(202,147,103,g);
for(int i=0;i<360;i+=30)
g.drawString(num[j++],(int)(202+radius*Math.cos(Math.PI*i/180)),(int)(147+radius*Math.sin(Math.PI*i/180))
);
g.setColor(Color.red);
ddaline(202,147,(int)(202+(radius-20)*Math.cos(Math.PI*(int)hour/180)),(int)(147+(radius-
20)*Math.sin(Math.PI*(int)hour/180)),g);
g.setColor(Color.blue);
ddaline(202,147,(int)(202+(radius-10)*Math.cos(Math.PI*min/180)),(int)(147+(radius-
10)*Math.sin(Math.PI*min/180)),g);
g.setColor(Color.green);
ddaline(202,147,(int)(202+radius*Math.cos(Math.PI*second/180)),(int)(147+radius*Math.sin(
Math.PI*second/180)),g);
g.setColor(Color.red);
second++; if(second>270) {
second=-90;
min+=6;
hour+=(float)1/2;
if(hour>270)
hour=-90;
if(min>270)
min=-90;
}
for(double r=0;r<100000000;r++);
for(double r=0;r<100000000;r++);
for(double r=0;r<100000000;r++);
for(double r=0;r<100000000;r++);
for(double r=0;r<20000000;r++);
for(double r=0;r<20000000;r++);
for(double r=0;r<20000000;r++);
for(double r=0;r<20000000;r++);
for(double r=0;r<10000;r++);
for(double r=0;r<10000;r++);
for(double r=0;r<10000;r++);
for(double r=0;r<10000;r++);
repaint();
}
public void actionPerformed(ActionEvent ae) {
h=Integer.parseInt(text1.getText());
```

```
m=Integer.parseInt(text2.getText());
s=Integer.parseInt(text3.getText());
min=(int)(6*m-90);
hour=(int)((30*((m/60)+h))-90);
second=(int)(6*s-90);
if(ae.getSource()==button) {
repaint();
}}
public void ddaline(int x1,int y1,int x2,int y2,Graphics g1) {
int dx=x2-x1,dy=y2-y1,step;
float xinc,yinc,x,y;
if(Math.abs(dx)>Math.abs(dy))
step=Math.abs(dx);
else step=Math.abs(dy);
xinc=(float)dx/step;
yinc=(float)dy/step;
x=x1;y=y1;
g1.drawString(".",(int)x,(int)y);
for(int i=1;i<step;i++) {
x=x+xinc;
y=y+yinc;
g1.drawString(".",(int)x,(int)y);
}}
public void midcircle(int xc,int yc,int r,Graphics g2) {
int x2=0;
int y2=r;
int p=5/4-r;
while(x2<y2){
if(p<0){
x2++;
p=p+2*x2+2+1;
}
else {
x2++; y2--;
p=p+2*x2+2+1-2*y2+2;
}
g2.drawString(".",xc+x2,yc+y2);
g2.drawString(".",xc+y2,yc+x2);
```

```
g2.drawString(".",xc-y2,yc+x2);
g2.drawString(".",xc-x2,yc+y2);
g2.drawString(".",xc+x2,yc-y2);
g2.drawString(".",xc+y2,yc-x2);
g2.drawString(".",xc-y2,yc-x2);
g2.drawString(".",xc-x2,yc-y2);
}}}

/*<applet code=clock.class width=400 height=400>
</applet>*/
```

Output:



Wheel Rotation:

Source Code:

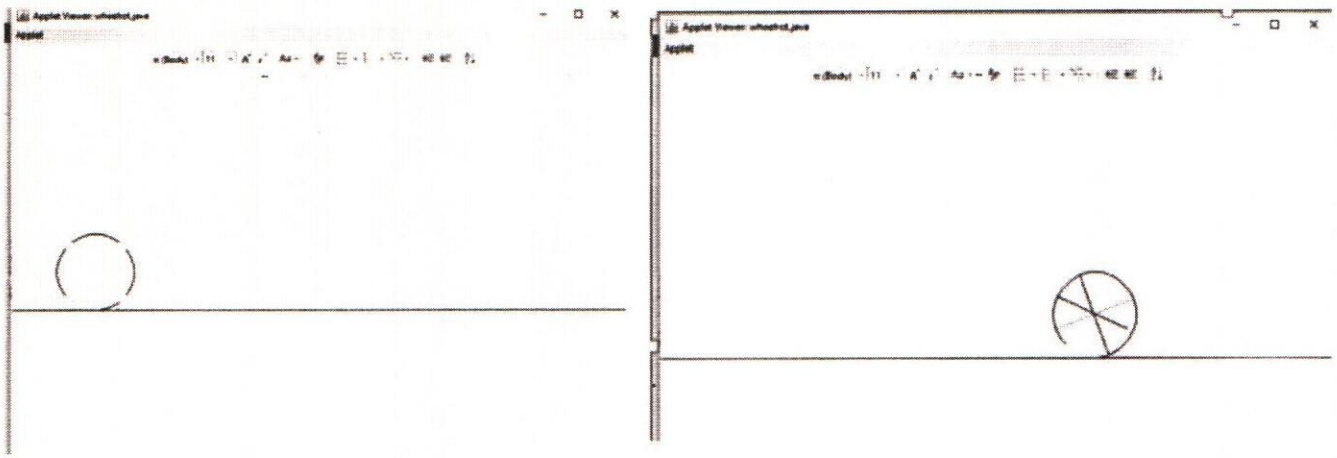
```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class wheelrot extends Applet implements ActionListener {
TextField t1,t2,t3;
Button b;
public void init() {
Label l1=new Label("xc(0 to 1000)");
Label l2=new Label("yc(0 to 600)");
Label l3=new Label("radius");
t1=new TextField(4);
t2=new TextField(4);
t3=new TextField(4);
b=new Button("start");
add(l1);
add(t1);
add(l2);
add(t2);
add(l3);
add(t3);
add(b);
b.addActionListener(this);
}
public void paint(Graphics g) {
Graphics g1=g;
int xc,yc,i,r,k;
double angle,pi,d;
pi=3.14;
angle=pi/180;
xc=Integer.parseInt(t1.getText());
yc=Integer.parseInt(t2.getText());
r=Integer.parseInt(t3.getText());
```

```
while(xc!=0) {
for(double j=0;
j<100000;j+=0.05);
g.setColor(Color.black);
g.fillRect(1,1,1360,1000);
g.setColor(Color.green);
drwLine(1,yc+r,1360,yc+r,g1);
drwOval(xc,yc,r,g1);
drwLine((int)(xc+r*Math.sin(angle)),
(int)(yc-r*Math.cos(angle)),
(int)(xc-r*Math.sin(angle)),
(int)(yc+r*Math.cos(angle)),g1);
drwLine((int)(xc-r*Math.cos(angle)),
(int)(yc-r*Math.sin(angle)),
(int)(xc+r*Math.cos(angle)),
(int)(yc+r*Math.sin(angle)),g1);
drwLine((int)(xc+r*Math.sin(angle+3.14/4)),
(int)(yc-r*Math.cos(angle+3.14/4)),
(int)(xc+r*Math.sin(angle+3.14/4)),
(int)(yc+r*Math.cos(angle+3.14/4)),g1);
drwLine((int)(xc-r*Math.cos(angle-3.14/4)),
(int)(yc-r*Math.sin(angle-3.14/4)),
(int)(xc+r*Math.cos(angle-3.14/4)),
(int)(yc+r*Math.sin(angle-3.14/4)),g1);
xc+=10*pi*r/180;
angle=angle+10*pi/180;
if(xc>(1000-r))
xc=20;
}}
public void actionPerformed(ActionEvent aa) {
repaint();
}
public void drwLine(int c,int d,int e,int f, Graphics g2) {
int x1=c;
```

```
int y1=d;
int x2=e;
int y2=f;
int step;
float xinc,yinc,x,y;
int dy=y2-y1;
int dx=x2-x1;
if(dx>dy)
step=dx;
else
step=dy;
xinc=(float)dx/step;
yinc=(float)dy/step;
x=x1;
y=y1;
g2.drawString(".",(int)x,(int)y);
for(int k=1;k<=step;k++) {
x=x+xinc;
y=y+yinc;
g2.drawString(".",(int)x,(int)y);
}}
public void drwOval(int a,int b,int r,Graphics g1){
int xc=a;
int yc=b;
int radius=r;
int x=0;
int y=radius;
float p=5/4-r;
do {
g1.drawString(".",Math.round(xc+x),Math.round(yc-y));
g1.drawString(".",Math.round(xc+y),Math.round(yc-x));
g1.drawString(".",Math.round(xc+y),Math.round(yc+x));
g1.drawString(".",Math.round(xc+x),Math.round(yc+y));
g1.drawString(".",Math.round(xc-y),Math.round(yc+x));
```

```
g1.drawString(".",Math.round(xc-y),Math.round(yc-x));
g1.drawString(".",Math.round(xc-x),Math.round(yc-y));
if(p<0) {
x=x+1;
p=p+(2*x)+3;
}
else {
x=x+1;
y=y-1;
p=p-2*y+2*x+5;
}}
while(x<=y);
}}
/*<applet code=wheelrot.class width=600 height=400>
</applet>*/
```

Output:



Beizer Curve:

Source Code:

```
import java.awt.*;
import java.applet.*;
import java.awt.geom.GeneralPath;
import java.awt.image.BufferedImage;
public class KBezier extends Applet {
Point[] controlPoints,curvePoints;
public void init() {
controlPoints=new Point[4];
curvePoints=new Point[25];
controlPoints[0]=new Point(20,260);
controlPoints[1]=new Point(50,10);
controlPoints[2]=new Point(250,50);
controlPoints[3]=new Point(450,290);
for(int i=0;i<curvePoints.length;i++)
curvePoints[i]=new Point(0,0);}
public void SubDivide(Point p1,Point p2,double t) {
if(p1.x>p2.x)
p1.x-=Math.abs(p1.x-p2.x)*t;
else
p1.x+=Math.abs(p1.x-p2.x)*t;
if (p1.y>p2.y)
p1.y-=Math.abs(p1.y-p2.y)*t;
else
p1.y+=Math.abs(p1.y-p2.y)*t;}
public void Compute() {
Point[] tmp=new Point[controlPoints.length];
for (int i=0; i<tmp.length; i++)
tmp[i] = new Point(0,0);
for (int i=0;i<curvePoints.length;i++){
double t = ((double) i)/(curvePoints.length-1);
for (int j=0; j<controlPoints.length; j++)
tmp[j]=new Point(controlPoints[j].x,controlPoints[j].y);
int Depth = tmp.length;
while (Depth>1) {
for (int j=0; j<Depth-1; j++)
```

```
SubDivide(tmp[j], tmp[j+1], t);
Depth--;}
curvePoints[i]=new Point(tmp[0].x, tmp[0].y);
}}
public void Draw(Graphics2D g2d) {
g2d.setColor(Color.BLACK);
for (int i=0; i<controlPoints.length-1; i++)
g2d.drawLine((int) controlPoints[i].x,(int) controlPoints[i].y, (int) controlPoints[i+1].x,(int) controlPoints[i+1].y);
GeneralPath path = new GeneralPath();
g2d.setColor(Color.RED);
path.moveTo(curvePoints[0].x, curvePoints[0].y);
for (int i=1; i<curvePoints.length; i++)
path.lineTo(curvePoints[i].x, curvePoints[i].y);
g2d.draw(path);}
public void paint(Graphics g) {
BufferedImage image = new BufferedImage(getWidth(), getHeight(), BufferedImage.TYPE_INT_RGB);
Graphics2D g2d = image.createGraphics();
g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON); g2d.setColor(Color.WHITE);
g2d.fillRect(0, 0, getWidth(), getHeight());
Compute();
Draw(g2d);
g.drawImage(image, 0, 0,this);
}}
/*<applet code=KBezier.class width=400 height=400></applet>*/
```

Output:-