NAME : GOURAV KUMAR SHAW

ENROLLMENT NO. : 2020CSB010

SECTION : GX

ASSIGNMENT-3

1. Develop a class for circle using Midpoint circle drawing algorithm. Hence draw the shape in Fig.5.

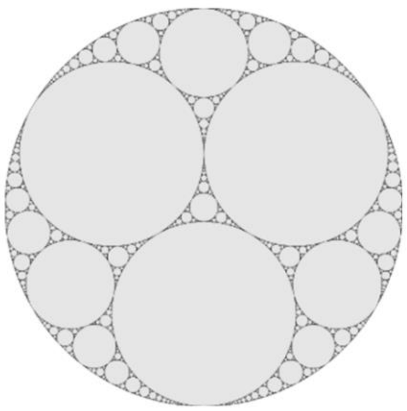


Fig. 5: A shape with circle

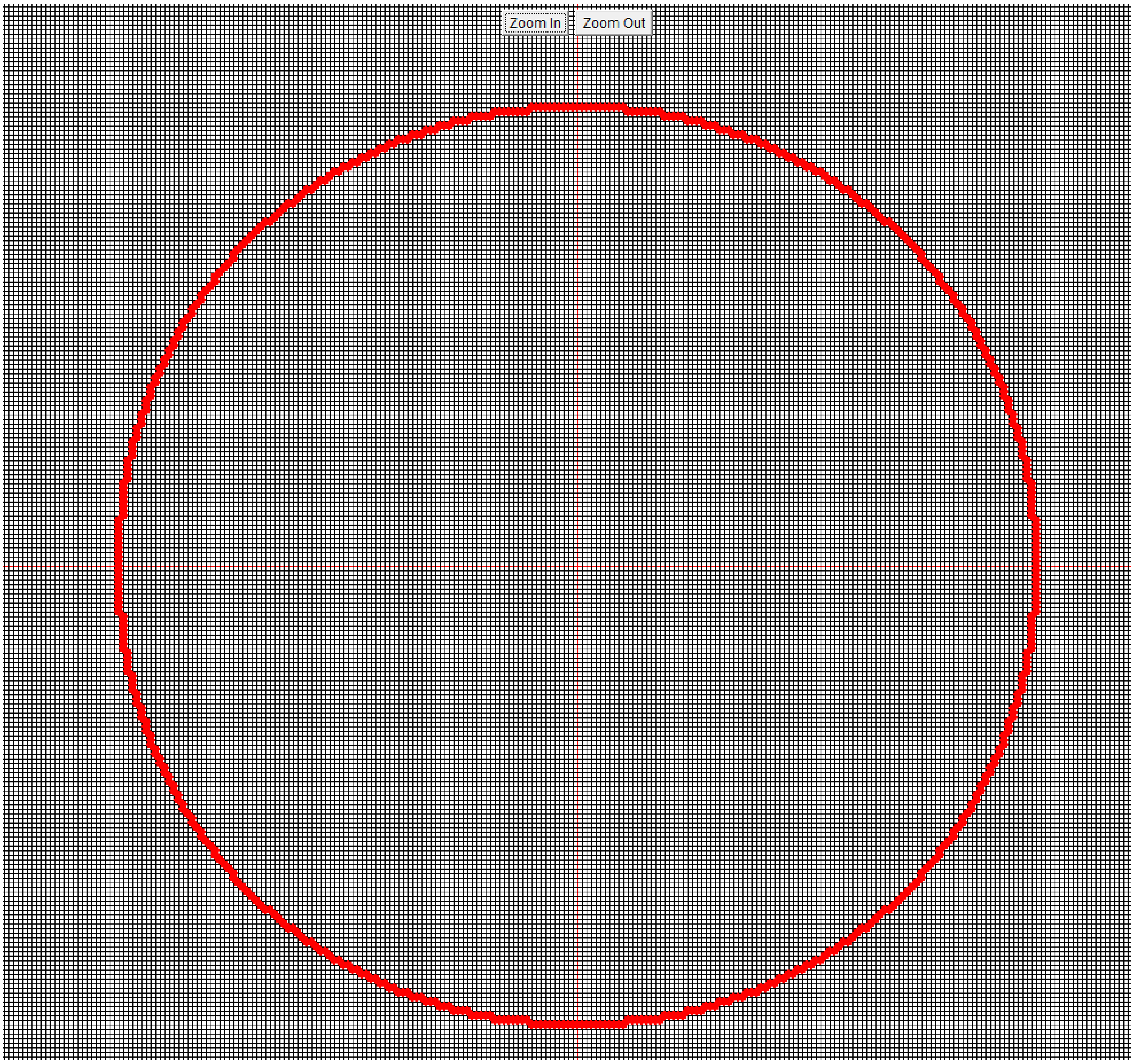
Code

```
//2020CSB010 GOURAV KUMAR SHAW

// 1. Develop a class for circle using Midpoint circle drawing algorithm.
Hence draw the shape given.

import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class circle extends Applet implements ActionListener {
    int scale = 40;
    int count1 = 0;

    public void init() {
        setBackground(Color.white);
        Button b1 = new Button("Zoom In");// Making the Button
        Button b2 = new Button("Zoom Out");// Making the Button
        add(b1);
        add(b2);
        b1.addActionListener(this);// adding the action Listener to allow the
interface to increase or decrease
                                    // with the clicking the button of Zoom In
or Zoon Out
        b2.addActionListener(this);// same as above
    }


    public void plotPoint(double x, double y, Graphics g, Color c) {
        int newX = (getX() + getWidth()) / 2;// changing the coordinate in the
        //origin at (0,0)
        int newY = (getY() + getHeight()) / 2;//
        g.setColor(c);

            g.fillOval((int)x * scale - (int) scale  + (int)newX, (int)newY -
(int) y * scale - (int) scale , (int) scale * 2, (int) scale * 2);// creating
the circle to be ploted
}



    public void paint(Graphics g) {
        int x = (getX() + getWidth()) / 2;// changing the coordinate to the
(0,0) base coordinate
        int y = (getY() + getHeight()) / 2;
        g.setColor(Color.red); // drawing the main coordinate for x and y
        g.drawLine(x, 0, x, getHeight());// x-axis as x value is changing
        g.drawLine(0, y, getWidth(), y);// y-axis as y value is changing
        g.setColor(Color.black);

        for (int i = x + scale; i < getWidth(); i += scale) {
            g.drawLine(i, 0, i, getHeight());
        }
        for (int i = scale; i < getWidth(); i += scale) {
            g.drawLine(x - i, 0, x - i, getHeight());
        }
        for (int i = y + scale; i < getHeight(); i += scale) {
            g.drawLine(0, i, getWidth(), i);
        }
        for (int i = scale; i < getHeight(); i += scale) {
            g.drawLine(0, y - i, getWidth(), y - i);
        }


        midPointCircleDraw(g, 100, 0, 0);
        midPointCircleDraw(g, 100, 0, 0);
        midPointCircleDraw(g, 46, 46, 27);
        midPointCircleDraw(g, 46, -46, 27);
        midPointCircleDraw(g, 46, 0, -54);
        midPointCircleDraw(g, 8, 0, 0);
        midPointCircleDraw(g, 22, 0, 78);
        midPointCircleDraw(g, 22, 65, -38);
        midPointCircleDraw(g, 22, -65, -38);
        midPointCircleDraw(g, 10, 33, 82);
        midPointCircleDraw(g, 10, -33, 82);
        midPointCircleDraw(g, 10, -88, -14);
        midPointCircleDraw(g, 10, 88, -14);
        midPointCircleDraw(g, 10, -54, -70);
        midPointCircleDraw(g, 10, 54, -70);
        midPointCircleDraw(g, 5, 50, 80);
        midPointCircleDraw(g, 5, -50, 80);
        midPointCircleDraw(g, 6, 94, 3);
        midPointCircleDraw(g, 6, -94, 3);
        midPointCircleDraw(g, 6, 45, -83);
        midPointCircleDraw(g, 6, -45, -83);
        midPointCircleDraw(g, 4, 58, 76);
        midPointCircleDraw(g, 4, -58, 76);
        midPointCircleDraw(g, 3, 95, 14);
        midPointCircleDraw(g, 3, -95, 14);
        midPointCircleDraw(g, 3, 36, -88);
        midPointCircleDraw(g, 3, -36, -88);
        midPointCircleDraw(g, 4, 0, 50);
        midPointCircleDraw(g, 4, 42, -24);
        midPointCircleDraw(g, 4, -42, -24);
        midPointCircleDraw(g, 2, 65, 72);
        midPointCircleDraw(g, 2, -65, 72);
        midPointCircleDraw(g, 3, 95, 22);
        midPointCircleDraw(g, 3, -95, 22);
        midPointCircleDraw(g, 2, 31, -92);
        midPointCircleDraw(g, 2, -31, -92);
        midPointCircleDraw(g, 4, 23, 92);
        midPointCircleDraw(g, 4, -23, 92);
        midPointCircleDraw(g, 4, 91, -30);
        midPointCircleDraw(g, 4, -91, -30);
        midPointCircleDraw(g, 4, 69, -64);
        midPointCircleDraw(g, 4, -69, -64);

    }


    public void midPointCircleDraw(Graphics g, double r,double x_centre,
                                double y_centre)
    {
```

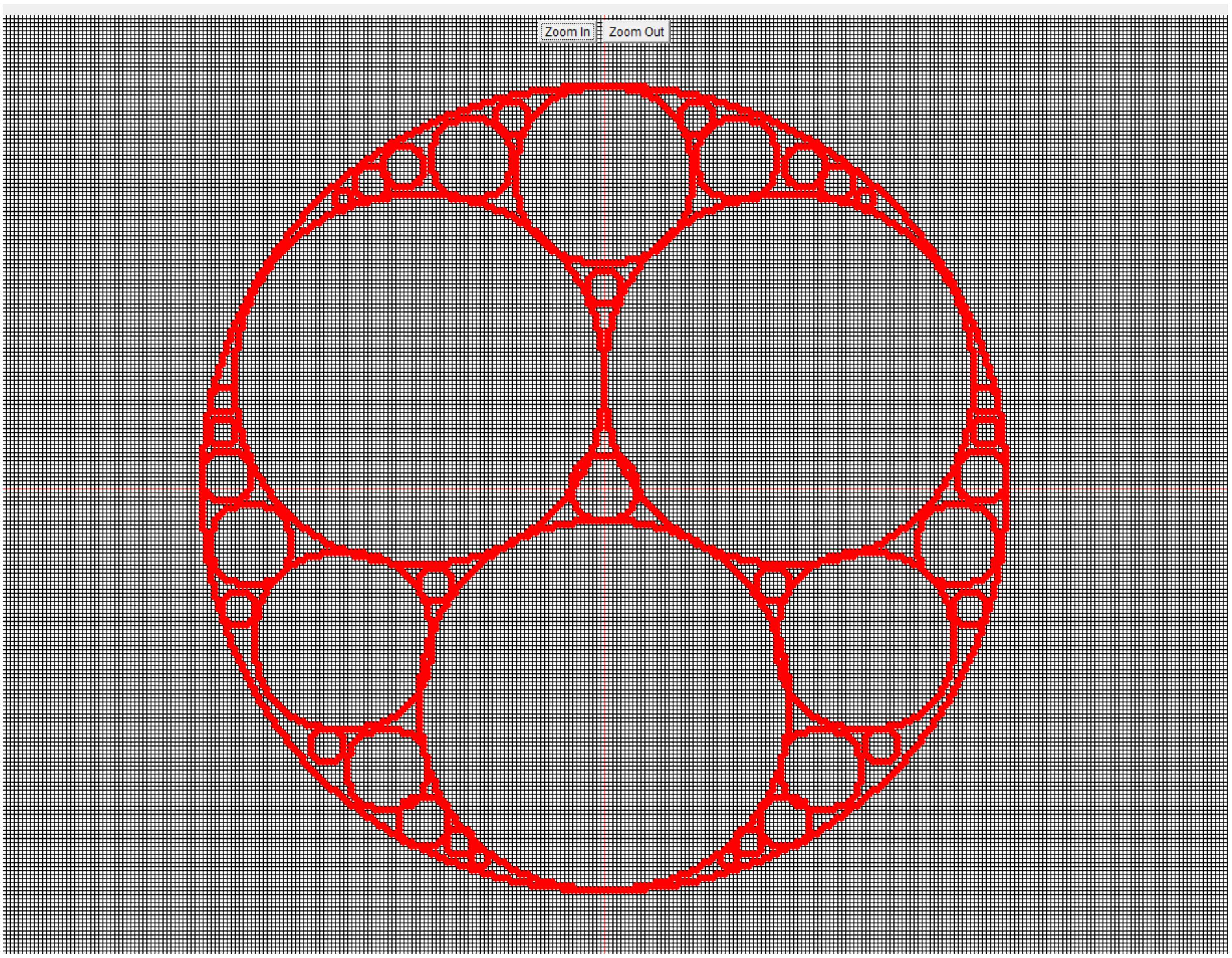
```
double x=0, y=r;
plotPoint(0+x_centre,r+y_centre,g,Color.red);
plotPoint(r+x_centre,0+y_centre,g,Color.red);
plotPoint(0+x_centre,-r+y_centre,g,Color.red);
plotPoint(-r+x_centre,0+y_centre,g,Color.red);

double P = 1 - r;
while (x < y) {
    // Mid-point is inside or on the perimeter
    if (P < 0){
        P= P+2*x+1;
        x=x+1;
        // y = y;
        plotPoint(x+x_centre,y+y_centre,g,Color.red);
        plotPoint(-x+x_centre,y+y_centre,g,Color.red);
        plotPoint(x+x_centre,-y+y_centre,g,Color.red);
        plotPoint(-x+x_centre,-y+y_centre,g,Color.red);
        plotPoint(y+x_centre,x+y_centre,g,Color.red);
        plotPoint(-y+x_centre,x+y_centre,g,Color.red);
        plotPoint(y+x_centre,-x+y_centre,g,Color.red);
        plotPoint(-y+x_centre,-x+y_centre,g,Color.red);

    }
    // Mid-point is outside the perimeter
    else {

        P = P + 2 * x - 2 * y + 1;
        x= x+1;
        y = y-1;
        plotPoint(x+x_centre,y+y_centre,g,Color.red);
        plotPoint(-x+x_centre,y+y_centre,g,Color.red);
        plotPoint(x+x_centre,-y+y_centre,g,Color.red);
        plotPoint(-x+x_centre,-y+y_centre,g,Color.red);
        plotPoint(y+x_centre,x+y_centre,g,Color.red);
        plotPoint(-y+x_centre,x+y_centre,g,Color.red);
        plotPoint(y+x_centre,-x+y_centre,g,Color.red);
        plotPoint(-y+x_centre,-x+y_centre,g,Color.red);

    }

    // All the perimeter points have already
    // been printed
    if (x > y)
        break;

    // Printing the generated point and its
```

```
            // reflection in the other octants after
            // translation
            System.out.print("(" + (x + x_centre)
                    + ", " + (y + y_centre) + ")");

            System.out.print("(" + (-x + x_centre)
                    + ", " + (y + y_centre) + ")");

            System.out.print("(" + (x + x_centre) +
                    ", " + (-y + y_centre) + ")");

            System.out.println("(" + (-x + x_centre)
                    + ", " + (-y + y_centre) + ")");

            // If the generated point is on the
            // line x = y then the perimeter points
            // have already been printed
            if (x != y) {

                System.out.print("(" + (y + x_centre)
                    + ", " + (x + y_centre) + ")");

                System.out.print("(" + (-y + x_centre)
                    + ", " + (x + y_centre) + ")");

                System.out.print("(" + (y + x_centre)
                    + ", " + (-x + y_centre) + ")");

                System.out.println("(" + (-y + x_centre)
                    + ", " + (-x + y_centre) +")");
            }
        }
    }

    public void actionPerformed(ActionEvent e) {
        String st = e.getActionCommand();
        if (st.equals("Zoom In"))
            scale = scale * 2;

        else
            scale = scale / 2;

        repaint();
    }
}
```

Circle



Shape drawing



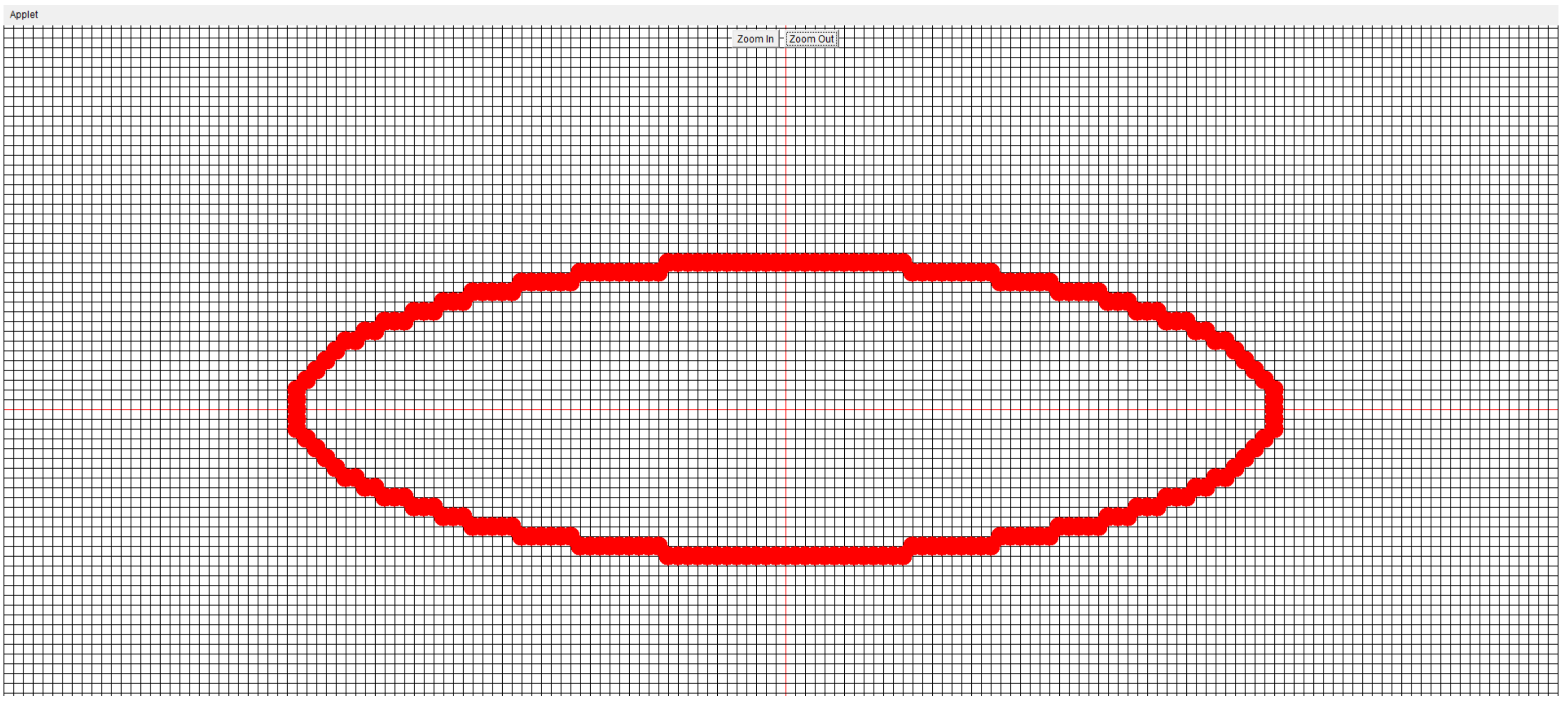
2. Develop a class for ellipse using Midpoint ellipse drawing algorithm.

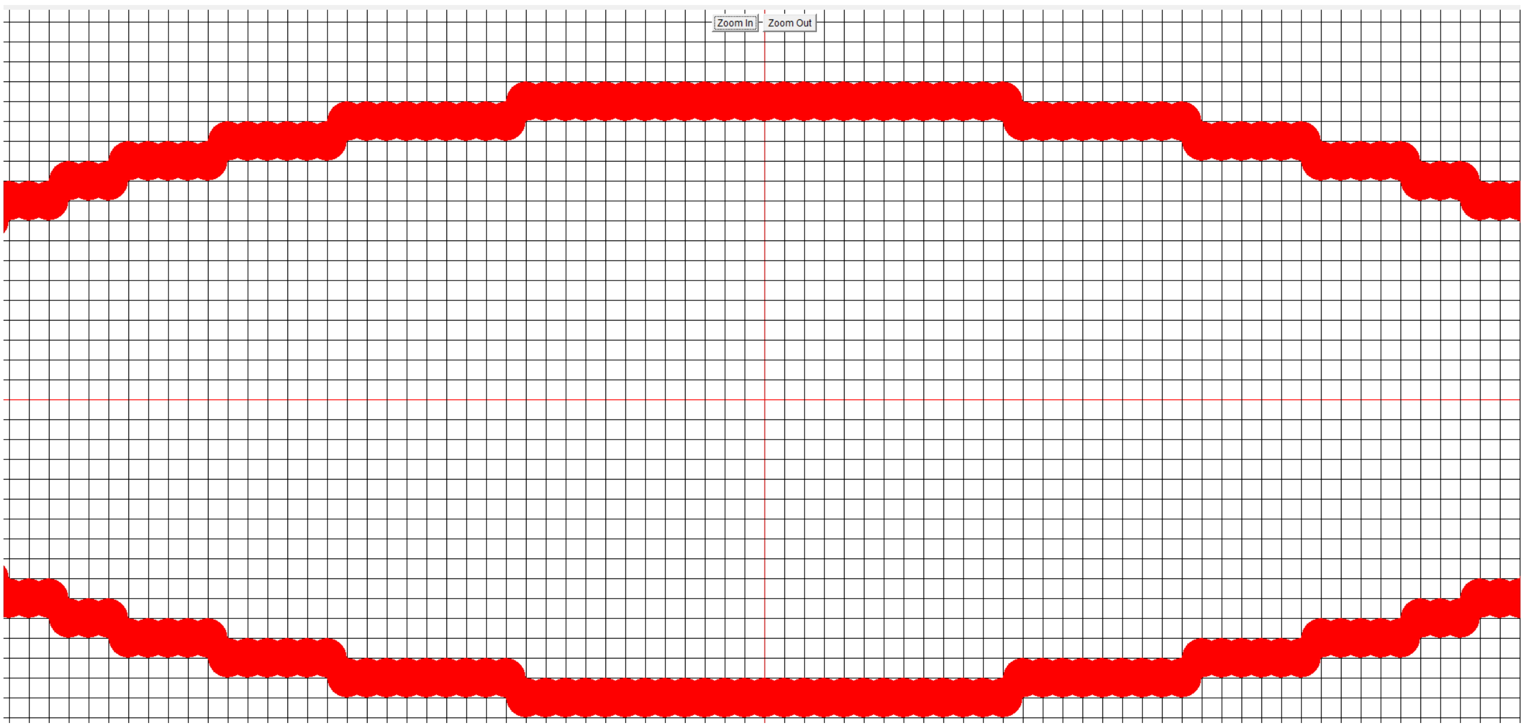
Code

```
//2020CSB010 GOURAV KUMAR SHAW

// 2. Develop a class for ellipse using Midpoint ellipse drawing algorithm

import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.text.DecimalFormat;

public class ellipse extends Applet implements ActionListener {
    int scale = 25;
    int count1 = 0;

    public void init() {
        setBackground(Color.white);

        Button b1 = new Button("Zoom In");// Making the Button
        Button b2 = new Button("Zoom Out");// Making the Button
        add(b1);
        add(b2);
        b1.addActionListener(this);// adding the action Listener to allow the
interface to increase or decrease
                                   // with the clicking the button of Zoom In
or Zoon Out
        b2.addActionListener(this);// same as above
    }


    public void plotPoint(double x, double y, Graphics g, Color c) {
        int newX = (getX() + getWidth()) / 2;// changing the coordinate in the
        //origin at (0,0)
        int newY = (getY() + getHeight()) / 2;//
        g.setColor(c);

        g.fillOval((int)x * scale - (int) scale  + (int)newX, (int)newY -(int)
y * scale - (int) scale , (int) scale * 2,
        (int) scale * 2);// creating the circle to be ploted
}
```

```java
    public void paint(Graphics g) {
        int x = (getX() + getWidth()) / 2;// changing the coordinate to the
(0,0) base coordinate
        int y = (getY() + getHeight()) / 2;
        g.setColor(Color.red); // drawing the main coordinate for x and y
        g.drawLine(x, 0, x, getHeight());// x-axis as x value is changing
        g.drawLine(0, y, getWidth(), y);// y-axis as y value is changing
        g.setColor(Color.black);

        for (int i = x + scale; i < getWidth(); i += scale) {
            g.drawLine(i, 0, i, getHeight());
        }
        for (int i = scale; i < getWidth(); i += scale) {
            g.drawLine(x - i, 0, x - i, getHeight());
        }
        for (int i = y + scale; i < getHeight(); i += scale) {
            g.drawLine(0, i, getWidth(), i);
        }
        for (int i = scale; i < getHeight(); i += scale) {
            g.drawLine(0, y - i, getWidth(), y - i);
        }


      midPointEllipse(50,15,0,0,g);

    }
public void midPointEllipse(float rx, float ry,
                        float xc, float yc,Graphics g)
{

    double dx, dy, d1, d2, x, y;
    x = 0;
    y = ry;

    // Initial decision parameter of region 1
    d1 = (ry * ry) - (rx * rx * ry) +
                    (0.25f * rx * rx);
    dx = 2 * ry * ry * x;
    dy = 2 * rx * rx * y;
    DecimalFormat df = new DecimalFormat("#,###,##0.00000");

    // For region 1
    while (dx < dy)
    {

        // Print points based on 4-way symmetry
        System.out.println(df.format((x + xc)) +
                            ", "+df.format((y + yc)));
        plotPoint(x + xc,y + yc,g,Color.red);

        System.out.println(df.format((-x + xc)) +
                            ", "+ df.format((y + yc)));
        plotPoint(-x + xc,y + yc,g,Color.red);


        System.out.println(df.format((x + xc)) +
                            ", "+ df.format((-y + yc)));
        plotPoint(x + xc,-y + yc,g,Color.red);


        System.out.println(df.format((-x + xc)) +
                            ", "+df.format((-y + yc)));
        plotPoint(-x + xc,-y + yc,g,Color.red);



        // Checking and updating value of
        // decision parameter based on algorithm
        if (d1 < 0)
        {
            x++;
            dx = dx + (2 * ry * ry);
            d1 = d1 + dx + (ry * ry);
        }
        else
        {
            x++;
            y--;
            dx = dx + (2 * ry * ry);
            dy = dy - (2 * rx * rx);
            d1 = d1 + dx - dy + (ry * ry);
        }
    }

    // Decision parameter of region 2
    d2 = ((ry * ry) * ((x + 0.5f) * (x + 0.5f)))
        + ((rx * rx) * ((y - 1) * (y - 1)))
        - (rx * rx * ry * ry);

    // Plotting points of region 2
    while (y >= 0) {

        // printing points based on 4-way symmetry
        System.out.println(df.format((x + xc)) +
                            ", " + df.format((y + yc)));
        plotPoint(x + xc,y + yc,g,Color.red);

        System.out.println(df.format((-x + xc)) +
                            ", "+ df.format((y + yc)));
        plotPoint(-x + xc,y + yc,g,Color.red);

        System.out.println(df.format((x + xc)) +
                            ", " + df.format((-y + yc)));
        plotPoint(x + xc,-y + yc,g,Color.red);

        System.out.println(df.format((-x + xc)) +
                            ", " + df.format((-y + yc)));
        plotPoint(-x + xc,-y + yc,g,Color.red);


        // Checking and updating parameter
        // value based on algorithm
        if (d2 > 0) {
            y--;
            dy = dy - (2 * rx * rx);
            d2 = d2 + (rx * rx) - dy;
        }
        else {
            y--;
            x++;
            dx = dx + (2 * ry * ry);
            dy = dy - (2 * rx * rx);
            d2 = d2 + dx - dy + (rx * rx);
        }
    }
}

    public void actionPerformed(ActionEvent e) {
        String st = e.getActionCommand();
        if (st.equals("Zoom In"))
            scale = scale * 2;

        else
            scale = scale / 2;

        repaint();
    }
}
```

Ellipse drawing


Applet
Zoom In
Zoom Out


Zoom In
Zoom Out