Assignment 3

By Neeraj Pratap Hazarika
2020csb040

Part- I:

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

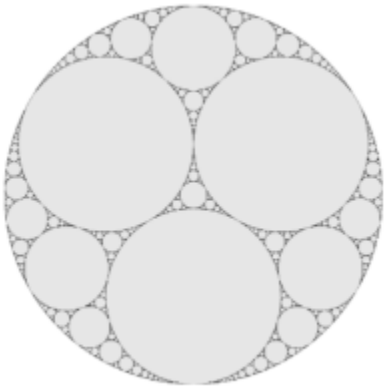


Fig. 5: A shape with circle

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.lang.Math;

public class test extends Applet implements ActionListener {
    // scale (distance between two points in coordinate plane
    int scale = 20;
    int originX;
    int originY;

    public void init() {
        setBackground(Color.white);
        Button zoom_in = new Button("Zoom In");
        Button zoom_out = new Button("Zoom Out");
        add(zoom_in);
        add(zoom_out);
        zoom_in.addActionListener(this);
        zoom_out.addActionListener(this);
    }
```

```
    public void plotpoint(Graphics g, int x, int y, Color C) {
        x = originX + x * scale;
        y = originY - y * scale;
        int plotpoint_x = x - scale / 2; // shifting x coordinate to
appletcoordinate
        int plotpoint_y = y - scale / 2; // shifting y coordinate to
appletcoordinate
        g.setColor(C);
        g.fillOval(plotpoint_x, plotpoint_y, scale, scale);
    }

    public int round(float coordinate, int axis) {

        if (axis == 0) {
            int prev = (int) (coordinate / scale) * scale;
            int next = prev + scale;

            if (coordinate - prev < next - coordinate)
                return prev;
            else
                return next;
        } else {
            int prev = (int) (coordinate / scale) * scale;
            int next = prev + scale;

            if (coordinate - prev < next - coordinate)
                return prev;
            else
                return next;
        }
    }

    /*
     * Mid Point Circle Drawing algorithm
     * - if radius is 0, we will plot only the center of the circle
     * - if radius is >0, we will start plotting from (r,0) to (r/2,r/2)
(for
     * example)
```

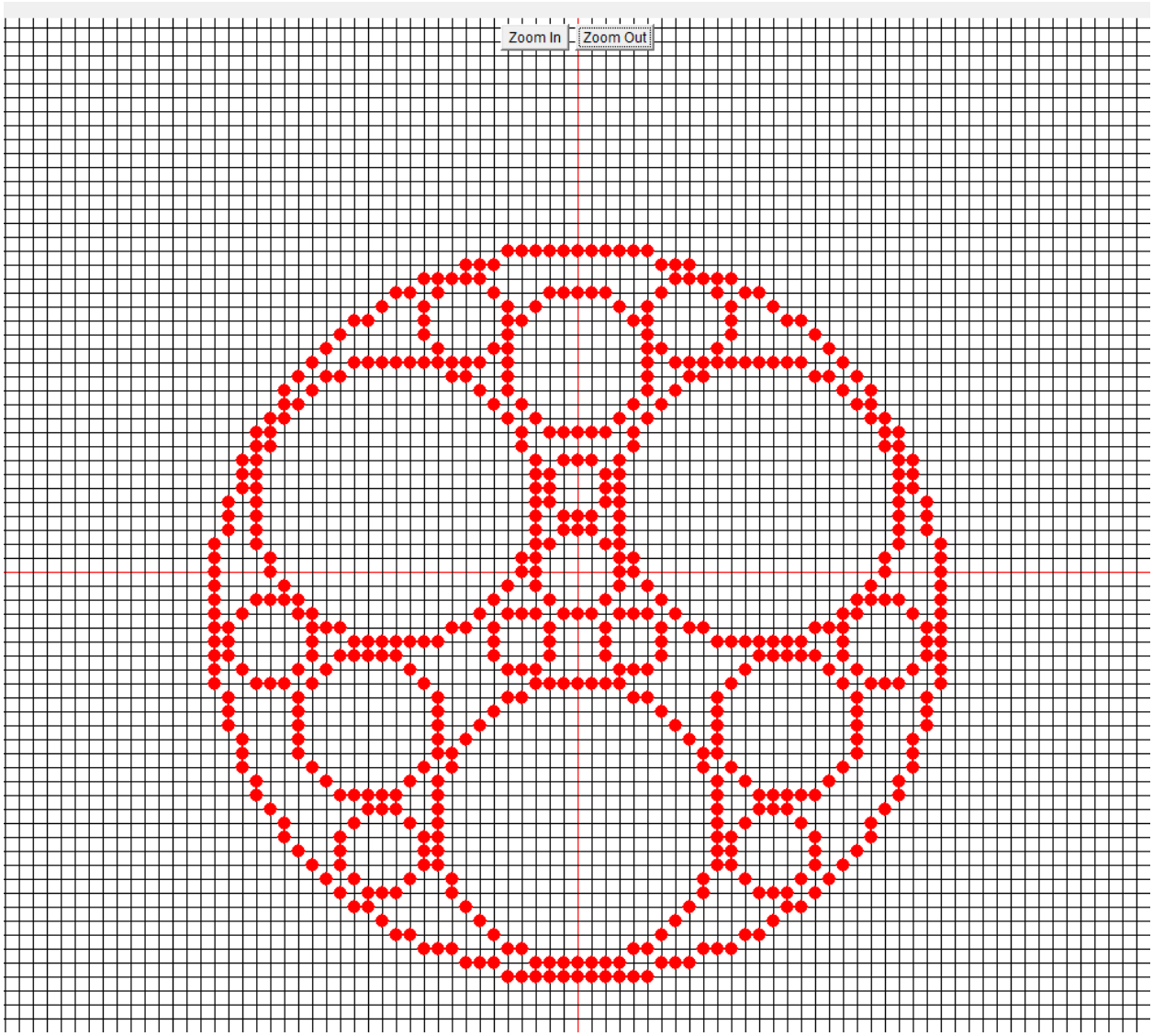
```
     * - while we plot the above points in 1st quadrant, its mirror image
will start
     * plotting from (0,r) to (r/2,r/2) in 1st quadrant itself
     * - this will happen for other 3 quadrants
     */

    public void midPointCircleDraw1(Graphics g, int x_origin, int
y_origin, int radius, Color c) {

        if (radius == 0) {
            plotpoint(g, x_origin, y_origin, c);
        } else {
            int x = radius;
            int y = 0;

            int p = 1 - radius;

            while (x >= y) {
                plotpoint(g, x_origin + x, y_origin + y, c);
                plotpoint(g, x_origin - x, y_origin + y, c);
                plotpoint(g, x_origin + x, y_origin - y, c);
                plotpoint(g, x_origin - x, y_origin - y, c);

                // plotting on the mirror will be done by above plotpoints
(one is enough)
                if (x != y) {
                    // switching x and y axis to plot its mirror image in
respective quadrant
                    plotpoint(g, x_origin + y, y_origin + x, c);
                    plotpoint(g, x_origin + y, y_origin - x, c);
                    plotpoint(g, x_origin - y, y_origin + x, c);
                    plotpoint(g, x_origin - y, y_origin - x, c);
                }

                y++;

                if (p <= 0) {
                    p = p + 2 * y + 1;
                } else {
                    x--;
```

```
                    p = p + 2 * y - 2 * x + 1;
                }
            }
        }
    }

    public void shape(Graphics g) {
        midPointCircleDraw1(g, 0, 0, 3, Color.red); // center circle
        midPointCircleDraw1(g, 0, 6, 2, Color.red); // circle above center
circle
        midPointCircleDraw1(g, -4, -5, 2, Color.red); // circle bottom
left to center circle
        midPointCircleDraw1(g, 4, -5, 2, Color.red); // circle bottom
right to center circle
        midPointCircleDraw1(g, 13, 5, 10, Color.red); // circle circle top
right to center circle
        midPointCircleDraw1(g, -13, 5, 10, Color.red); // circle top left
to center circle
        midPointCircleDraw1(g, 0, -18, 10, Color.red); // circle bottom to
center circle
        midPointCircleDraw1(g, 0, 15, 5, Color.red); // circle above(x2)
to center center circle
        midPointCircleDraw1(g, 15, -11, 5, Color.red); // circle bottom
right (x2) to center circle
        midPointCircleDraw1(g, -15, -11, 5, Color.red); // circle bottom
right (x2) to center circle
        midPointCircleDraw1(g, -14, -20, 3, Color.red); // circle below
bottom left (x2) to center circle
        midPointCircleDraw1(g, 14, -20, 3, Color.red); // circle below
bottom right (x2) to center circle
        midPointCircleDraw1(g, 22, -5, 3, Color.red); // circle upper
bottom right (x2) to center circle
        midPointCircleDraw1(g, -22, -5, 3, Color.red); // circle upper
bottom left (x2) to center circle
        midPointCircleDraw1(g, 8, 18, 3, Color.red); // circle upper (x3)
right to center circle
        midPointCircleDraw1(g, -8, 18, 3, Color.red); // circle upper (x3)
left to center circle
        midPointCircleDraw1(g, 0, -3, 26, Color.red); // outer most circle
    }
```

```
public void paint(Graphics g) {
    // shift the origin and put the coordinates in new variables
    originX = (getX() + getWidth()) / 2;
    originY = (getY() + getHeight()) / 2;

    // drawing coordinates lines
    g.setColor(Color.red);
    g.drawLine(originX, 0, originX, getHeight());
    g.drawLine(0, originY, getWidth(), originY);

    // drawing Grid
    // vertical lines
    g.setColor(Color.black);
    // right half vertical lines
    for (int i = originX + scale; i < getWidth(); i += scale) {
        g.drawLine(i, 0, i, getHeight());
    }
    // left half vertical lines
    for (int i = scale; originX - i >= 0; i += scale) {
        g.drawLine(originX - i, 0, originX - i, getHeight());
    }

    // horizontal lines
    // right half horizontal lines
    for (int i = originY + scale; i < getHeight(); i += scale) {
        g.drawLine(0, i, getWidth(), i);
    }
    // left half horizontal lines
    for (int i = scale; originY - i >= 0; i += scale) {
        g.drawLine(0, originY - i, getWidth(), originY - i);
    }

    shape(g);
}

public void actionPerformed(ActionEvent e) {
    String st = e.getActionCommand();
    if (st.equals("Zoom In"))
        scale += 4;
```

```
        else
            scale -= 4;
        repaint();
    }
}
```

Output:



Part- II:

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

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.lang.Math;

public class test extends Applet implements ActionListener {
    // scale (distance between two points in coordinate plane
    int scale = 20;
    int originX;
    int originY;

    public void init() {
        setBackground(Color.white);
        Button zoom_in = new Button("Zoom In");
        Button zoom_out = new Button("Zoom Out");
        add(zoom_in);
        add(zoom_out);
        zoom_in.addActionListener(this);
        zoom_out.addActionListener(this);
    }

    public void plotpoint(Graphics g, int x, int y, Color C) {
        x = originX + x * scale;
        y = originY - y * scale;
        int plotpoint_x = x - scale / 2; // shifting x coordinate to
appletcoordinate
        int plotpoint_y = y - scale / 2; // shifting y coordinate to
appletcoordinate
        g.setColor(C);
        g.fillOval(plotpoint_x, plotpoint_y, scale, scale);
    }

    public int round(float coordinate, int axis) {

        if (axis == 0) {
            int prev = (int) (coordinate / scale) * scale;
            int next = prev + scale;

            if (coordinate - prev < next - coordinate)
                return prev;
```

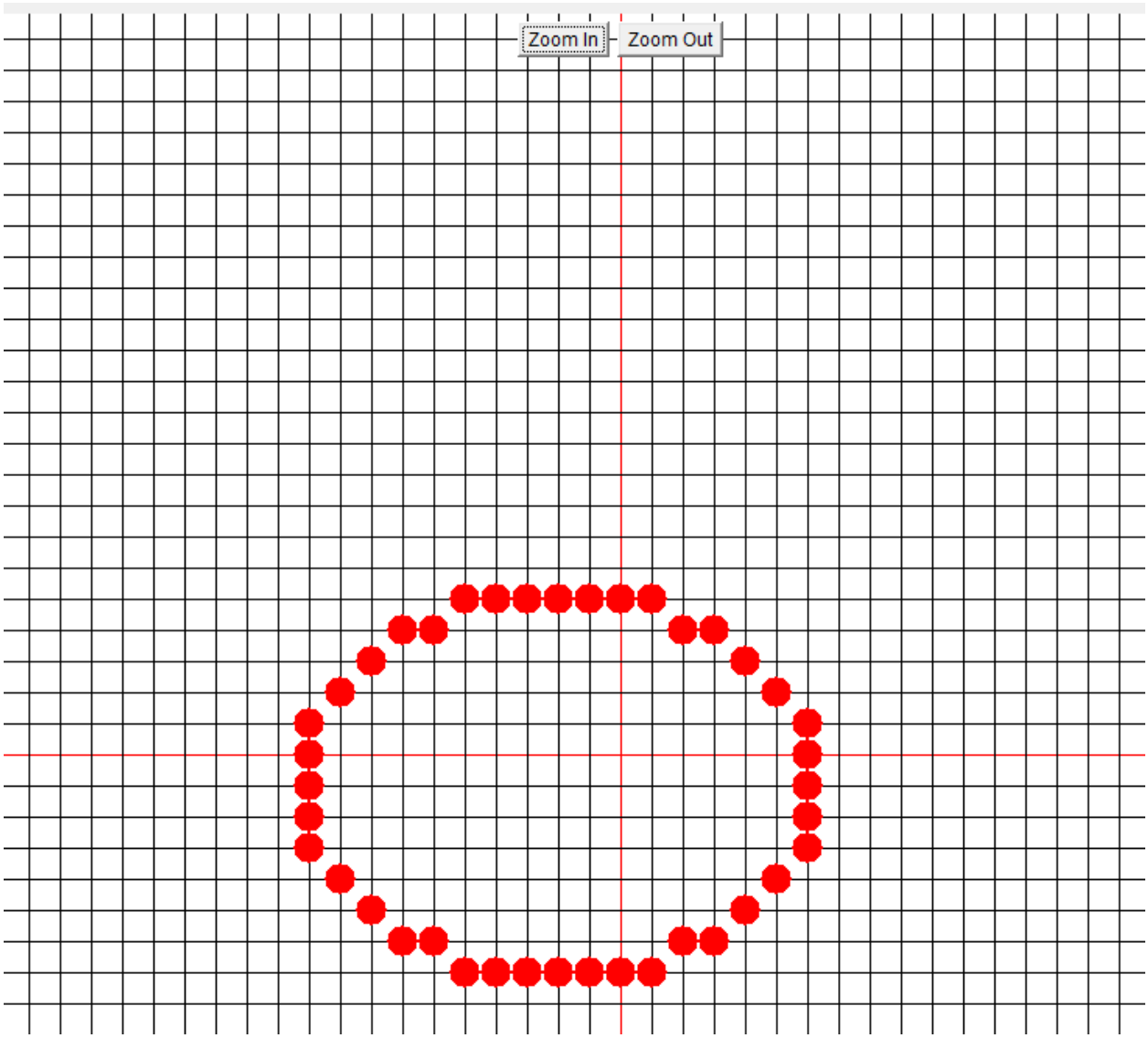
```
            else
                return next;
        } else {
            int prev = (int) (coordinate / scale) * scale;
            int next = prev + scale;

            if (coordinate - prev < next - coordinate)
                return prev;
            else
                return next;
        }
    }

    /*
     * Mid Point Ellipse Drawing Algorithm
     *
     * - there will be two parts
     * - we plot first part with p1 as decision parameter
     * - find dy and dx
     * - set x = 0, y = y_radius
     * - plot (x,y),(x,-y),(-x,y),(-x,-y)
     * - x++
     * - based on p1, if p1>0, then y-- else no change in y.
     * - based on p1, we will also set the next p1
     * - this will continue until dy>dx
     * - keep updating dy and dx inside loop
     *
     * - second part
     * - similar thing will happen
     * - decision parameter will be p2
     * - increment of decision parameter will happen differently based on
sign of p2
     */

    public void midPointEllipseDrawingAlgorithm(Graphics g, int x_origin,
int y_origin, int radius_x, int radius_y,
            Color c) {

        double dy, dx, d1, d2;
        int x = 0;
```

```
        int y = radius_y;

        /*
         * f(x,y) = (radius_y^2)*(x^2) + (radius_x^2)*(y^2) -
(radius_x^2)*(radius_y^2)
         * dx is partial derivative of f(x,y) wrt x
         * dy is partial derivative of f(x,y) wrt y
         */
        dx = 2 * radius_y * radius_y * x;
        dy = 2 * radius_x * radius_x * y;
        d1 = (radius_y * radius_y) - (radius_x * radius_x * radius_y) +
(0.25 * radius_x * radius_x);

        // first plot part is until slope = 0 to -1, or dy<dx
        while (dx < dy) {

            plotpoint(g, x_origin + x, y_origin + y, c);
            plotpoint(g, x_origin - x, y_origin + y, c);
            plotpoint(g, x_origin + x, y_origin - y, c);
            plotpoint(g, x_origin - x, y_origin - y, c);

            x++;
            dx = dx + (2 * radius_y * radius_y);

            if (d1 < 0) {
                d1 = d1 + dx + (radius_y * radius_y);
            } else {
                y--;
                dy = dy - (2 * radius_x * radius_x);
                d1 = d1 + dx - dy + (radius_y * radius_y);
            }
        }

        // Decision parameter of region 2, after slope = -1 to -inf
        d2 = ((radius_y * radius_y) * ((x + 0.5) * (x + 0.5))) +
((radius_x * radius_x) * ((y - 1) * (y - 1)))
                - (radius_x * radius_x * radius_y * radius_y);

        while (y >= 0) {
```

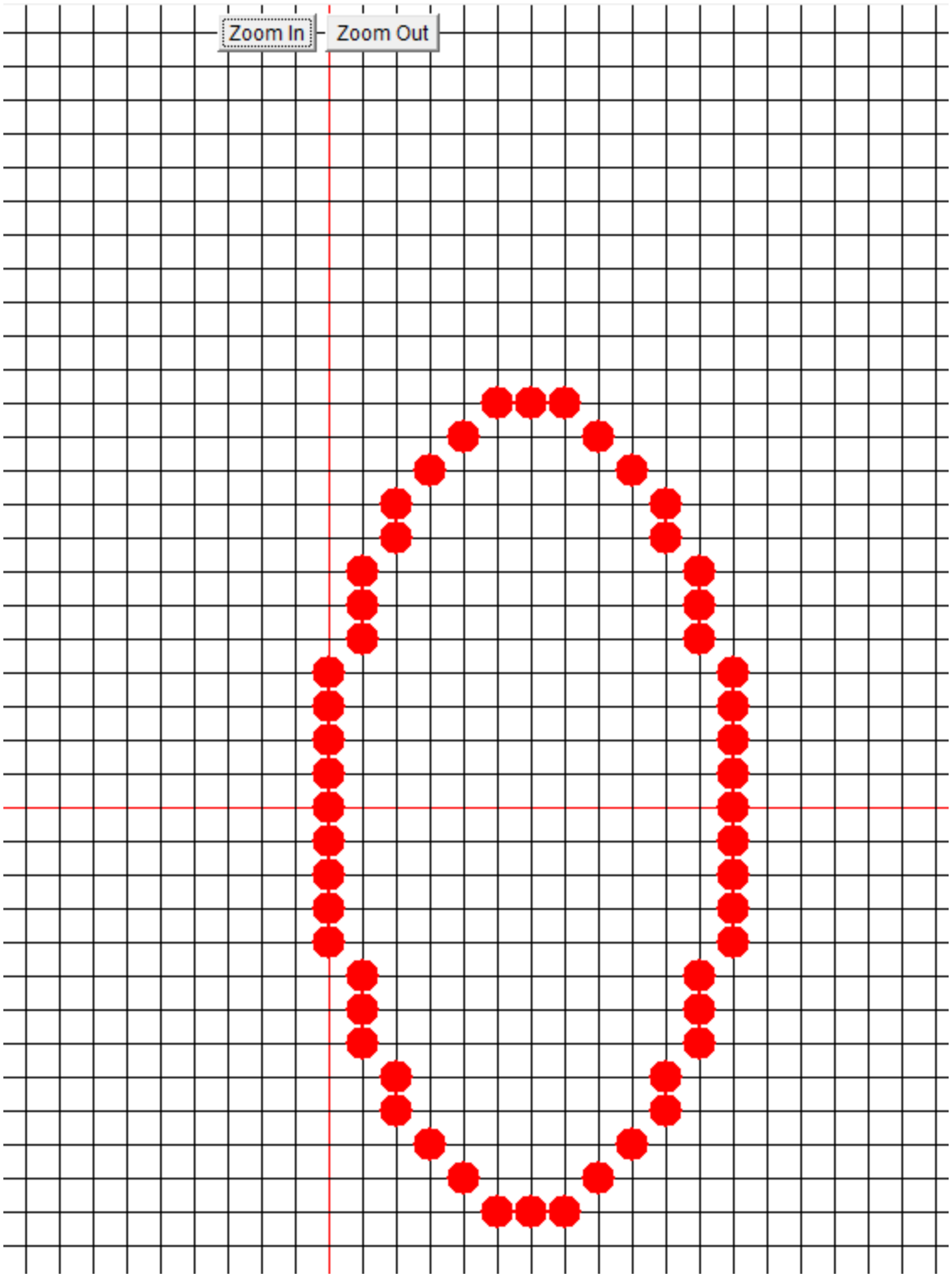
```
            plotpoint(g, x_origin + x, y_origin + y, c);
            plotpoint(g, x_origin - x, y_origin + y, c);
            plotpoint(g, x_origin + x, y_origin - y, c);
            plotpoint(g, x_origin - x, y_origin - y, c);

            y--;
            dy = dy - (2 * radius_x * radius_x);

            if (d2 > 0) {
                d2 = d2 + (radius_x * radius_x) - dy;
            } else {
                x++;
                dx = dx + (2 * radius_y * radius_y);
                d2 = d2 + dx - dy + (radius_x * radius_x);
            }
        }
    }

    public void paint(Graphics g) {
        // shift the origin and put the coordinates in new variables
        originX = (getX() + getWidth()) / 2;
        originY = (getY() + getHeight()) / 2;

        // drawing coordinates lines
        g.setColor(Color.red);
        g.drawLine(originX, 0, originX, getHeight());
        g.drawLine(0, originY, getWidth(), originY);

        // drawing Grid
        // vertical lines
        g.setColor(Color.black);
        // right half vertical lines
        for (int i = originX + scale; i < getWidth(); i += scale) {
            g.drawLine(i, 0, i, getHeight());
        }
        // left half vertical lines
        for (int i = scale; originX - i >= 0; i += scale) {
            g.drawLine(originX - i, 0, originX - i, getHeight());
        }
```

```
        // horizontal lines
        // right half horizontal lines
        for (int i = originY + scale; i < getHeight(); i += scale) {
            g.drawLine(0, i, getWidth(), i);
        }
        // left half horizontal lines
        for (int i = scale; originY - i >= 0; i += scale) {
            g.drawLine(0, originY - i, getWidth(), originY - i);
        }

        midPointEllipseDrawingAlgorithm(g, -2, -1, 8, 6, Color.red);
    }

    public void actionPerformed(ActionEvent e) {
        String st = e.getActionCommand();
        if (st.equals("Zoom In"))
            scale += 4;
        else
            scale -= 4;
        repaint();
    }
}
```

Output:



For, x_origin = -2, y_origin = -1, radius_x = 8, radius_y = 6



For, x_origin = 6, y_origin = 0, radius_x = 6, radius_y = 12