```
//A java program to print the flag of Bangladesh using Java Swing and
awt.
//Computer Graphics Lab, CSE3222
//October 4, 2018

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

public class BanFlag extends JFrame {

    public  BanFlag()
    {
        setBackground(Color.DARK_GRAY);
        setTitle("Bangladesh Flag");
        setSize(640,480);
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

    }

    public void paint(Graphics g)
    {
      g.setColor(Color.black);
      g.fillRect(10,30,20,500);
      g.setColor(new Color(46, 139, 89));
      g.fillRect(40,50,300,170);
      g.setColor(Color.red);
      g.fillOval(120,80,120,120);

    }

    public static void main(String[] args)
    {
        BanFlag flag = new BanFlag();
        flag.paint(null);
    }
}
```

```
//A Digital Clock Code using Java


import java.awt.*;
import java.awt.event.*;
import java.awt.geom.*;
import javax.swing.*;
import java.util.*;


public class digitalClock extends JPanel implements Runnable{

      String mode="am";
      int size=7;
      int totSecs;
      digitalNumber h1,h2,m1,m2,s1,s2;
      boolean pulse=false;
      boolean afNoon;
      GregorianCalendar cal;
      Thread th;

      public digitalClock(){
            h1=new digitalNumber(20,100,size);
            h2=new digitalNumber(100,100,size);
            m1=new digitalNumber(200,100,size);
            m2=new digitalNumber(280,100,size);
            s1=new digitalNumber(360,60,size/2);
            s2=new digitalNumber(400,60,size/2);

            setBackground(Color.BLACK);
            setLayout(new BorderLayout());

            JCheckBox modeBox = new JCheckBox("Toggle AM/PM mode");
            modeBox.addItemListener(new ItemListener(){
                  public void itemStateChanged(ItemEvent e){
                        if(e.getStateChange() ==
ItemEvent.DESELECTED)mode="am";
                        if(e.getStateChange() ==
ItemEvent.SELECTED)mode="pm";
                        repaint();
                  }
            });
            add(modeBox,BorderLayout.SOUTH);
            start();
      }
```

```
public void start(){
      if(th==null){
            th=new Thread(this);
            th.start();
      }
}

public void run(){
      while(th!=null){
            try{
                  totSecs=setSecs();
                  showTime();
                  if(pulse)pulse=false;
                  else pulse=true;
                  repaint();
                  Thread.sleep(1000);
            }catch(Exception e){}
      }
}

public void stop(){
      if(th!=null)th=null;
}

public int setSecs(){
      cal=new GregorianCalendar();
      int h,m,s;
      h=cal.get(Calendar.HOUR)*3600;
      if(cal.get(Calendar.AM_PM)==Calendar.PM)h+=3600*12;
      m=cal.get(Calendar.MINUTE)*60;
      s=cal.get(Calendar.SECOND);
      return h+m+s;
}

public int divide(int a, int b){
      int z = 0;
  int i = a;

  while (i>= b)
    {
      i = i - b;
      z++;
    }
  return z;
```

```
    }

      public void showTime(){
            if(totSecs>86399)totSecs=0;

            int hours=divide(totSecs,3600);
            int minutes=divide(totSecs,60)-hours*60;
            int 
seconds=totSecs-hours*3600-60*divide((totSecs-hours*3600),60);

            if(hours<13&&afNoon==true)afNoon=false;
            if(mode=="pm" && hours> 12){
                  hours=hours-12;
                  afNoon=true;
            }
            //set Hours
            if(hours<10){
                  h1.turnOffNumber();
                  h2.setNumber(hours);
            }else if(hours>=10 && hours<20){
                  h1.setNumber(1);
                  h2.setNumber(hours-10);
            }else{
                  h1.setNumber(2);
                  h2.setNumber(hours-20);
            }
            //set Minutes
            int dM=divide(minutes,10);
            if(dM<6)m1.setNumber(dM);
            else m1.setNumber(0);
            m2.setNumber(minutes-dM*10);
            //set Seconds
            int dS=divide(seconds,10);
            if(dS<6)s1.setNumber(dS);
            else s1.setNumber(0);
            s2.setNumber(seconds-dS*10);

            //System.out.println(""+hours+" : "+minutes+" . "+seconds);
      }
      public void showDots(Graphics2D g2){
            if(pulse)g2.setColor(Color.RED);
            else g2.setColor(new Color(230,230,230));
            g2.fill(new Rectangle2D.Double(178,65,14,14));
            g2.fill(new Rectangle2D.Double(178,135,14,14));
      }
    public void showMode(Graphics2D g2){
        if(afNoon && mode=="pm"){g2.setColor(Color.RED);
            g2.drawString("PM", 360, 140);
            g2.setColor(new Color(230,230,230));
            g2.drawString("AM", 360, 120);
        }else if(afNoon==false && mode=="pm"){
            g2.setColor(new Color(230,230,230));
            g2.drawString("PM", 360, 140);
            g2.setColor(Color.RED);
            g2.drawString("AM", 360, 120);
        }else if(mode=="am"){
            g2.setColor(new Color(230,230,230));
            g2.drawString("PM", 360, 140);
            g2.drawString("AM", 360, 120);
        }
    }

    public void paint(Graphics g){
        super.paint(g);
        Graphics2D g2 = (Graphics2D)g;
        h1.drawNumber(g2);
        h2.drawNumber(g2);
        m1.drawNumber(g2);
        m2.drawNumber(g2);
        s1.drawNumber(g2);
        s2.drawNumber(g2);
        showDots(g2);
        showMode(g2);
    }

    public static void main(String[] a){
        JFrame f=new JFrame("Digital Clock");
    f.setSize(450,260);
    f.setTitle("My Digital Clock");
    //f.setResizable(false);
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f.setVisible(true);
    f.getContentPane().add(new digitalClock());
    }
}

class digitalNumber{
    int x,y;
    int k;
```

```
led[] leds;

public digitalNumber(int x, int y, int k){
      this.x=x;
      this.y=y;
      this.k=k;
      leds = new led[7];
      leds[0] = new led(x,y,"vert");
      leds[1] = new led(x,y+10*k,"vert");
      leds[2] = new led(x+8*k,y,"vert");
      leds[3] = new led(x+8*k,y+10*k,"vert");
      leds[4] = new led(x+2*k,y-9*k,"horiz");
      leds[5] = new led(x+2*k,y+k,"horiz");
      leds[6] = new led(x+2*k,y+11*k,"horiz");
}

public void setNumber(int num){
      if(num==0){
            leds[0].setState(true);
            leds[1].setState(true);
            leds[2].setState(true);
            leds[3].setState(true);
            leds[4].setState(true);
            leds[5].setState(false);
            leds[6].setState(true);
      }else if(num==1){
            leds[0].setState(false);
            leds[1].setState(false);
            leds[2].setState(true);
            leds[3].setState(true);
            leds[4].setState(false);
            leds[5].setState(false);
            leds[6].setState(false);
      }else if(num==2){
            leds[0].setState(false);
            leds[1].setState(true);
            leds[2].setState(true);
            leds[3].setState(false);
            leds[4].setState(true);
            leds[5].setState(true);
            leds[6].setState(true);
      }else if(num==3){
            leds[0].setState(false);
            leds[1].setState(false);
            leds[2].setState(true);
```

```
        leds[3].setState(true);
        leds[4].setState(true);
        leds[5].setState(true);
        leds[6].setState(true);
    }else if(num==4){
        leds[0].setState(true);
        leds[1].setState(false);
        leds[2].setState(true);
        leds[3].setState(true);
        leds[4].setState(false);
        leds[5].setState(true);
        leds[6].setState(false);
    }else if(num==5){
        leds[0].setState(true);
        leds[1].setState(false);
        leds[2].setState(false);
        leds[3].setState(true);
        leds[4].setState(true);
        leds[5].setState(true);
        leds[6].setState(true);
    }else if(num==6){
        leds[0].setState(true);
        leds[1].setState(true);
        leds[2].setState(false);
        leds[3].setState(true);
        leds[4].setState(true);
        leds[5].setState(true);
        leds[6].setState(true);
    }else if(num==7){
        leds[0].setState(false);
        leds[1].setState(false);
        leds[2].setState(true);
        leds[3].setState(true);
        leds[4].setState(true);
        leds[5].setState(false);
        leds[6].setState(false);
    }else if(num==8 ){
        leds[0].setState(true);
        leds[1].setState(true);
        leds[2].setState(true);
        leds[3].setState(true);
        leds[4].setState(true);
        leds[5].setState(true);
        leds[6].setState(true);
    }else if(num==9){
        leds[0].setState(true);
        leds[1].setState(false);
        leds[2].setState(true);
        leds[3].setState(true);
        leds[4].setState(true);
        leds[5].setState(true);
        leds[6].setState(true);
    }
}

public void turnOffNumber(){
    for(int i=0;i<7;i++){
        leds[i].setState(false);
    }
}

public void drawNumber(Graphics2D g2){
    for(int i=0; i<7; i++){
        leds[i].render(g2);
    }
}

class led{
    int x, y;
    Polygon p;
    String type;
    boolean lightOn=false;

    public led(int x, int y, String type){
        this.x=x;
        this.y=y;
        this.type=type;

        p = new Polygon();

        if(type=="vert"){
            p.addPoint(x,y);
            p.addPoint(x+k,y+k);
            p.addPoint(x+2*k,y);
            p.addPoint(x+2*k,y-8*k);
            p.addPoint(x+k,y-9*k);
            p.addPoint(x,y-8*k);
        }

        if(type=="horiz"){
```

```
                    p.addPoint(x,y);
                    p.addPoint(x+k,y+k);
                    p.addPoint(x+5*k,y+k);
                    p.addPoint(x+6*k,y);
                    p.addPoint(x+5*k,y-k);
                    p.addPoint(x+k,y-k);
              }
        }

        public void render(Graphics2D g2){
              g2.setColor(new Color(230,230,230));
              if(lightOn)g2.setColor(Color.RED);
              g2.fillPolygon(p);
        }

        public void setState(boolean s){
              lightOn=s;
        }
  }
}
```

Legend Ruhul :P

```
// A Java Code to translate a rectangle


import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Graphics;
import java.awt.Graphics2D;
import javax.swing.JFrame;
import javax.swing.JPanel;

class Surface extends JPanel {

    private void doDrawing(Graphics g) {

        Graphics2D g2d = (Graphics2D) g.create();

        g2d.setPaint(new Color(150, 150, 150));
        g2d.fillRect(20, 20, 80, 50);

        //Here we are setting value to translate
      g2d.translate(150, 50);
        g2d.fillRect(20, 20, 80, 50);

        g2d.dispose();
    }

    @Override
    public void paintComponent(Graphics g) {

        super.paintComponent(g);
        doDrawing(g);
    }
}

public class Translation extends JFrame {

    public Translation() {

        initUI();
    }

    private void initUI() {

        add(new Surface());
        setTitle("Translation");
        setSize(600, 480);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {

                Translation translation = new Translation();
                translation.setVisible(true);
            }
        });
    }
}
```

```
// A Java Code to rotate a rectangle

import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Graphics;
import java.awt.Graphics2D;
import javax.swing.JFrame;
import javax.swing.JPanel;

class Surface extends JPanel {

    private void doDrawing(Graphics g) {

        Graphics2D g2d = (Graphics2D) g.create();

        g2d.setPaint(new Color(150, 150, 150));
        g2d.fillRect(20, 20, 80, 50);
        g2d.translate(180, -50);

        //set value to rotate
        g2d.rotate(Math.PI/4);
        g2d.fillRect(80, 80, 80, 50);

        g2d.dispose();
    }

    @Override
    public void paintComponent(Graphics g) {

        super.paintComponent(g);
        doDrawing(g);
    }
}

public class Rotation extends JFrame {

    public Rotation() {

        initUI();
    }

    private void initUI() {

        setTitle("Rotation");
        add(new Surface());

        setSize(600, 480);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {

                Rotation rotation = new Rotation();
                rotation.setVisible(true);
            }
        });
    }
}
```

Legend Ruhul :P

```
// A Java Code to scale a rectangle

import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import javax.swing.JFrame;
import javax.swing.JPanel;

class Surface extends JPanel {

    private void doDrawing(Graphics g) {

        Graphics2D g2d = (Graphics2D) g.create();

        g2d.setColor(new Color(150, 150, 150));
        g2d.fillRect(20, 20, 80, 50);

        AffineTransform tx1 = new AffineTransform();
        tx1.translate(110, 22);
        tx1.scale(0.5, 0.5);

        g2d.setTransform(tx1);
        g2d.fillRect(0, 0, 80, 50);

        AffineTransform tx2 = new AffineTransform();

        //here we are again translating and scaling
      tx2.translate(170, 20);
        tx2.scale(1.5, 1.5);

        g2d.setTransform(tx2);
        g2d.fillRect(0, 0, 80, 50);

        g2d.dispose();
    }

    @Override
    public void paintComponent(Graphics g) {

        super.paintComponent(g);
        doDrawing(g);
    }
}

public class Scaling extends JFrame {

    public Scaling() {

        initUI();
    }

    private void initUI() {

        add(new Surface());

        setTitle("Scaling");
        setSize(330, 160);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {

                Scaling scaling = new Scaling();
                scaling.setVisible(true);
            }
        });
    }
}
```

```java
// A java code for self square Fractals

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

public class squareFractal {
      public static void main(String[] args)
      {
            FractalFrame frame = new FractalFrame();
            frame.setTitle("Square Fractal Generator 1.0");
            frame.setVisible(true);
      }
}

class FractalFrame extends JFrame
{
      private JPanel panel;
    private int width, height;
    private int x = 200;
    private int limit = 15;

      public FractalFrame()
      {
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            final int defaultWidth  = 800;
            final int defaultHeight = 800;
            setSize(defaultWidth, defaultHeight);
            setLocation(300,50);

            panel = new JPanel();
            Container contentPane = getContentPane();
            contentPane.add(panel, "Center");
      }

      public void paint(Graphics g) {
        super.paint(g);
        Graphics2D g2 = (Graphics2D) g;
        //Gets the size of the frame
        width  = getWidth();
        height = getHeight();
        g.setColor(Color.BLACK);
        //Creates the starting position of the first square to fit
EXACTLY center to the frame.
        int w = (width/2)-(x/2);
        int h = (height/2)-(x/2);
        //Draws the first square
        g.fillRect(w, h, x, x);
        //Starts the recursive fractal generation if the complexity is
greater than 0.
        if(limit > 0) drawSquare(w,h,g);
        //When complete, notifies the user.
        JOptionPane.showMessageDialog(null, "Generating Fractal =
Complete!");
    }

    // Draws the first level of squares and then recursively draws the
remaining amount (Stops when N == Limit).
    // Each concurrent square will be exactly half that of its parent.
Thus:
    // --> fillRect( [X-Coordinate], [Y-Coordinate], [Width of Square],
[Height of Square])
    private void drawSquare(int w, int h, Graphics g) {
        // N will be used as a checker to count how many levels of
recursion have occured.
        int n = 0;
        //Draws Pos 0 (Top-Left)
        g.fillRect(w-(x/2), h-(x/2), x/2, x/2);
        drawSquare(w-(x/2), h-(x/2), g, n, 0, x/2);
        //Draws Pos 1 (Top-Right)
        g.fillRect(w+x, h-(x/2), x/2, x/2);
        drawSquare(w+x, h-(x/2), g, n, 1, x/2);
        //Draws pos 2 (Bottom-Right)
        g.fillRect(w+x, h+x, x/2, x/2);
        drawSquare(w+x, h+x, g, n, 2, x/2);
        //Draws pos 3 (Bottom-Left)
        g.fillRect(w-(x/2), h+x, x/2, x/2);
        drawSquare(w-(x/2), h+x, g, n, 3, x/2);
    }

    // Recursive function to draw squares
    private void drawSquare(int w, int h, Graphics g, int n, int origin,
int size) {
        // Stops the recursion loop when it has reached its complexity
limit.
        if(n == limit) return;
        n++;
        // This bit of math is key. It prevents the program from drawing
a square at the location of its parent.
        origin = (origin+2)%4;
        //Draws Pos 0
        if(origin != 0) {
            g.fillRect(w-(size/2), h-(size/2), size/2, size/2);
            //Recursive Call
            drawSquare(w-(size/2), h-(size/2),g,n,0,size/2);
        }
        //Draws Pos 1
        if(origin != 1) {
            g.fillRect(w+size, h-(size/2), size/2, size/2);
            //Recursive Call
            drawSquare(w+size, h-(size/2),g,n,1,size/2);
        }
        //Draws pos 2
        if(origin != 2) {
            g.fillRect(w+size, h+size, size/2, size/2);
            //Recursive Call
            drawSquare(w+size, h+size,g,n,2,size/2);
        }
        //Draws pos 3
        if(origin != 3) {
            g.fillRect(w-(size/2), h+size, size/2, size/2);
            //Recursive Call
            drawSquare(w-(size/2), h+size,g,n,3,size/2);
        }
    }
}
```

```
// A java Code to implement  Julia Set

import java.awt.*;
import java.awt.image.BufferedImage;
import javax.swing.*;

public class JuliaSet extends JPanel {
    private final int maxIter = 300;
    private final double zoom = 1;
    private double cY, cX;

    public JuliaSet() {
        setPreferredSize(new Dimension(1080, 720));
        setBackground(Color.WHITE);
    }

    void drawJuliaSet(Graphics2D g) {
        int w = getWidth();
        int h = getHeight();
        BufferedImage image = new BufferedImage(w, h,
                BufferedImage.TYPE_INT_RGB);

        cX = -0.7;
        cY = 0.27015;
        double moveX = 0, moveY = 0;
        double zx, zy;

        for (int x = 0; x < w; x++) {
            for (int y = 0; y < h; y++) {
                zx = 1.5 * (x - w / 2) / (0.5 * zoom * w) + moveX;
                zy = (y - h / 2) / (0.5 * zoom * h) + moveY;
                float i = maxIter;
                while (zx * zx + zy * zy < 4 && i > 0) {
                    double tmp = zx * zx - zy * zy + cX;
                    zy = 2.0 * zx * zy + cY;
                    zx = tmp;
                    i--;
                }
                int c = Color.HSBtoRGB((maxIter / i) % 1, 1, i > 0 ? 1 :
0);
                image.setRGB(x, y, c);
            }
        }
        g.drawImage(image, 0, 0, null);
    }
    @Override
    public void paintComponent(Graphics gg) {
        super.paintComponent(gg);
        Graphics2D g = (Graphics2D) gg;
        g.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                RenderingHints.VALUE_ANTIALIAS_ON);
        drawJuliaSet(g);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            JFrame f = new JFrame();
            f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            f.setTitle("Julia Set");
            f.setResizable(false);
            f.add(new JuliaSet(), BorderLayout.CENTER);
            f.pack();
            f.setLocationRelativeTo(null);
            f.setVisible(true);
        });
    }
}
```

```java
// A Java Code to implement Line Drawing Algorithm

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.SwingUtilities;
import javax.swing.WindowConstants;

public class Bresenham {

    public static void main(String[] args) {
        SwingUtilities.invokeLater(Bresenham::run);
    }

    private static void run() {
        JFrame f = new JFrame();
        f.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        f.setTitle("Bresenham");

        f.getContentPane().add(new BresenhamPanel());
        f.pack();

        f.setLocationRelativeTo(null);
        f.setVisible(true);
    }
}

class BresenhamPanel extends JPanel {

    private final int pixelSize = 10;

    BresenhamPanel() {
        setPreferredSize(new Dimension(600, 500));
        setBackground(Color.WHITE);
    }

    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);

        int w = (getWidth() - 1) / pixelSize;
        int h = (getHeight() - 1) / pixelSize;
        int maxX = (w - 1) / 2;
        int maxY = (h - 1) / 2;
        int x1 = -maxX, x2 = maxX * -2 / 3, x3 = maxX * 2 / 3, x4 =
maxX;
        int y1 = -maxY, y2 = maxY * -2 / 3, y3 = maxY * 2 / 3, y4 =
maxY;

        drawLine(g, 0, 0, x3, y1); // NNE
        //drawLine(g, 0, 0, x4, y2); // ENE
        //drawLine(g, 0, 0, x4, y3); // ESE
        //drawLine(g, 0, 0, x3, y4); // SSE
        //drawLine(g, 0, 0, x2, y4); // SSW
        //drawLine(g, 0, 0, x1, y3); // WSW
        //drawLine(g, 0, 0, x1, y2); // WNW
        //drawLine(g, 0, 0, x2, y1); // NNW
    }

    private void plot(Graphics g, int x, int y) {
        int w = (getWidth() - 1) / pixelSize;
        int h = (getHeight() - 1) / pixelSize;
        int maxX = (w - 1) / 2;
        int maxY = (h - 1) / 2;

        int borderX = getWidth() - ((2 * maxX + 1) * pixelSize + 1);
        int borderY = getHeight() - ((2 * maxY + 1) * pixelSize + 1);
        int left = (x + maxX) * pixelSize + borderX / 2;
        int top = (y + maxY) * pixelSize + borderY / 2;

        g.setColor(Color.black);
        g.drawOval(left, top, pixelSize, pixelSize);
    }

    private void drawLine(Graphics g, int x1, int y1, int x2, int y2) {
        // delta of exact value and rounded value of the dependent
variable
        int d = 0;

        int dx = Math.abs(x2 - x1);
        int dy = Math.abs(y2 - y1);

        int dx2 = 2 * dx; // slope scaling factors to
        int dy2 = 2 * dy; // avoid floating point

        int ix = x1 < x2 ? 1 : -1; // increment direction
        int iy = y1 < y2 ? 1 : -1;
        int x = x1;
        int y = y1;

        if (dx >= dy) {
            while (true) {
                plot(g, x, y);
                if (x == x2)
                    break;
                x += ix;
                d += dy2;
                if (d > dx) {
                    y += iy;
                    d -= dx2;
                }
            }
        } else {
            while (true) {
                plot(g, x, y);
                if (y == y2)
                    break;
                y += iy;
                d += dx2;
                if (d > dy) {
                    x += ix;
                    d -= dy2;
                }
            }
        }
    }
}
```

```
// A Java Code to Implement Midpoint Circle Drawing  Algorithm

import java.io.*;
import java.util.*;
import java.math.*;
import javax.swing.*;
import java.awt.*;
import java.awt.geom.*;
import java.awt.*;


public class MidpointCircle {

public static void main(String[] args) {
   SwingUtilities.invokeLater(MidpointCircle::run);
  }

  private static void run() {
        JFrame f = new JFrame();
        f.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        f.setTitle("Midpoint Circle");

        f.getContentPane().add(new CirclePanel());
        f.pack();

        f.setLocationRelativeTo(null);
        f.setVisible(true);
    }
}

class CirclePanel extends JPanel {

    private final int pixelSize = 10;

    CirclePanel() {
        setPreferredSize(new Dimension(600, 500));
        setBackground(Color.WHITE);
    }

    @Override
    public void paint(Graphics g) {
    int r = 150;
    int d = (5 / 4) * r;
    int x = 0;
    int y = r;
    do {
      g.setColor(Color.red);
      g.drawLine(y + 200, x + 200, y + 200, x + 200);
      g.drawLine(x + 200, y + 200, x + 200, y + 200);
      g.drawLine(x + 200, -y + 200, x + 200, -y + 200);
      g.drawLine(y + 200, -x + 200, y + 200, -x + 200);
      g.drawLine(-y + 200, -x + 200, -y + 200, -x + 200);
      g.drawLine(-x + 200, -y + 200, -x + 200, -y + 200);
      g.drawLine(-x + 200, y + 200, -x + 200, y + 200);
      g.drawLine(-y + 200, x + 200, -y + 200, x + 200);

      if (d < 0) {
        d = d + 2 * x + 3;
      } else {
        d = d + 2 * (x - y) + 5;
        y = y - 1;
      }
      x = x + 1;
    }
    while (x < y);

  }
}
```