**Object-Oriented Programming**

**Lab Session #4**

**I. References**

* Java How to program: [https://github.com/nikhil-vytla/Java-How-to-Program-(Early-Objects)-](https://github.com/nikhil-vytla/Onclass-Coding-Curriculum/blob/master/Java%20How%20to%20Program%20(Early%20Objects)%2010th%20Edition.pdf)

[10th-Edition.pdf](https://github.com/nikhil-vytla/Onclass-Coding-Curriculum/blob/master/Java%20How%20to%20Program%20(Early%20Objects)%2010th%20Edition.pdf)

* + Download Code Examples: <https://github.com/pdeitel/JavaHowToProgram10eEarlyObjectsVersion>
  + Refer to Chapters 3, 4, 5, and 7.
* Swing Tutorial: <https://www.tutorialspoint.com/swing/index.htm>

**II. Exercises**

You are required to implement the following design as well as a main() method in another class to test your implementation:

1. Using JOptionPane dialog called **input dialog** to ask users to input the user’s name and responds with a message dialog containing a greeting and the name that the user entered. (Refer to Chapter 3, section 3.6)



**Fig. 1** | Obtaining user input from a dialog.

**Dialog.java:**

package Lab4.Question1;

import javax.swing.JOptionPane;

public class Dialog {

    public Dialog(){}

    public void display(){

        String name = JOptionPane.showInputDialog("What is your name?");

        String message = String.format("Welcome, %s, to Java Programming!", name);

        JOptionPane.showMessageDialog(null, message);

    }

}

**Test.java:**

package Lab4.Question1;

public class Test {

    public static void main(String[] args) {

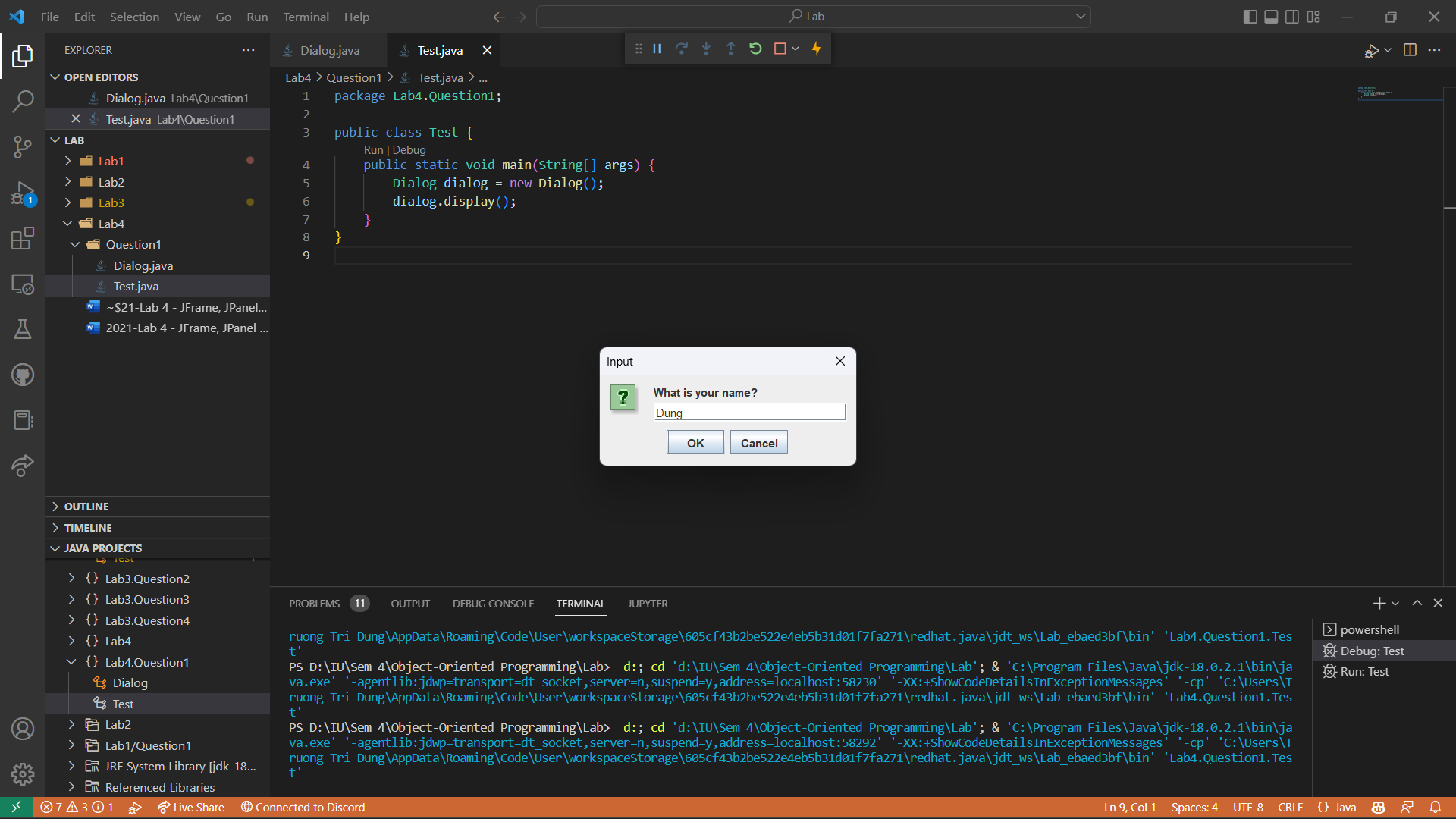
        Dialog dialog = new Dialog();

        dialog.display();

    }

}

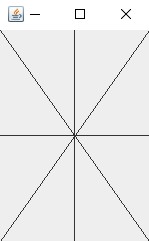
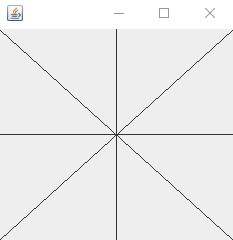
**Output:**



A screenshot of a computer

Description automatically generated

1. Using class Graphics(from package java.awt), which provides various methods for drawing text and shapes onto the screen, and class JPanel(from package javax.swing), which provides an area on which we can draw, to create a simple application that draws four lines as shown in Figure 2. (Refer to Chapter 4, section 4.15)



**Fig. 2** | Creating JFrame to display DrawPanel.

**DrawFourLines.java:**

package Lab4.Question2;

import javax.swing.JPanel;

import java.awt.Graphics;

import java.awt.Color;

public class DrawFourLines extends JPanel {

    public DrawFourLines(){

    }

    @Override

    public void paintComponent(Graphics g) {

        int width = getWidth();

        int height = getHeight();

        super.paintComponent(g);

        try {

            g.setColor(Color.BLACK);

            g.drawLine(0, 0, width, height);

            g.drawLine(0, height, width, 0);

            g.drawLine(0, height / 2, width, height / 2);

            g.drawLine(width / 2, 0, width / 2, height);

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

**Test.java:**

package Lab4.Question2;

import javax.swing.JFrame;

public class Test {

    public static void main(String[] args) {

        DrawFourLines drawFourLines = new DrawFourLines();

        JFrame frame = new JFrame();

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.add(drawFourLines);

        frame.setSize(400,400);

        frame.setVisible(true);

    }

}

**Output:**

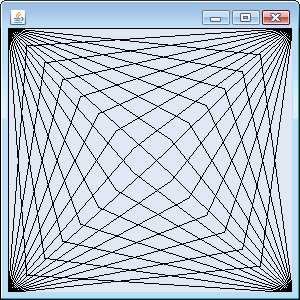
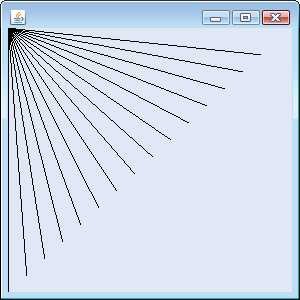
A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

1. Using loops and control statements to draw lines can lead to many interesting designs.
2. Create the design in the left screen capture of Fig. 3. This design draws lines from the top-left corner, fanning them out until they cover the upper-left half of the panel. One approach is to divide the width and height into an equal number of steps (we found 15 steps worked well). The first endpoint of a line will always be in the top-left corner (0, 0). The second endpoint can be found by starting at the bottom-left corner and moving up one vertical step and right one horizontal step. Draw a line between the two endpoints. Continue moving up and to the right one step to find each successive endpoint. The figure should scale accordingly as you resize the window.
3. Modify part (a) to have lines fan out from all four corners, as shown in the right screen capture of Fig. 3. Lines from opposite corners should intersect along the middle.



**Fig. 3** | Lines fanning from a corner.

**DrawTopLeft.java:**

package Lab4.Question3;

import javax.swing.JPanel;

import java.awt.Graphics;

import java.awt.Color;

public class DrawTopLeft extends JPanel{

    public DrawTopLeft(){

    }

    @Override

    public void paintComponent(Graphics g) {

        int width = getWidth()/15;

        int height = getHeight()/15;

        super.paintComponent(g);

        try {

            g.setColor(Color.BLACK);

            for (int i=0;i<15;i++){

                g.drawLine(0, 0, width\*i, height\*(15-i));

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

**DrawAllCorners.java:**

package Lab4.Question3;

import javax.swing.JPanel;

import java.awt.Graphics;

import java.awt.Color;

public class DrawAllCorners extends JPanel{

    public DrawAllCorners(){

    }

    @Override

    public void paintComponent(Graphics g) {

        int width = getWidth()/15;

        int height = getHeight()/15;

        super.paintComponent(g);

        try {

            g.setColor(Color.BLACK);

            for (int i=0;i<15;i++){

                g.drawLine(0, 0, width\*i, height\*(15-i));

            }

            for (int i=0;i<15;i++){

                g.drawLine(0, getHeight(), width\*(15-i), height\*(15-i));

            }

            for (int i=0;i<15;i++){

                g.drawLine(getWidth(), 0, width\*i, height\*i);

            }

            for (int i=0;i<15;i++){

                g.drawLine(getWidth(), getHeight(), width\*(15-i), height\*i);

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

**Test.java:**

package Lab4.Question3;

import javax.swing.JFrame;

public class Test {

    public static void main(String[] args) {

        //DrawTopLeft panel = new DrawTopLeft();

        DrawAllCorners panel = new DrawAllCorners();

        JFrame frame = new JFrame();

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.add(panel);

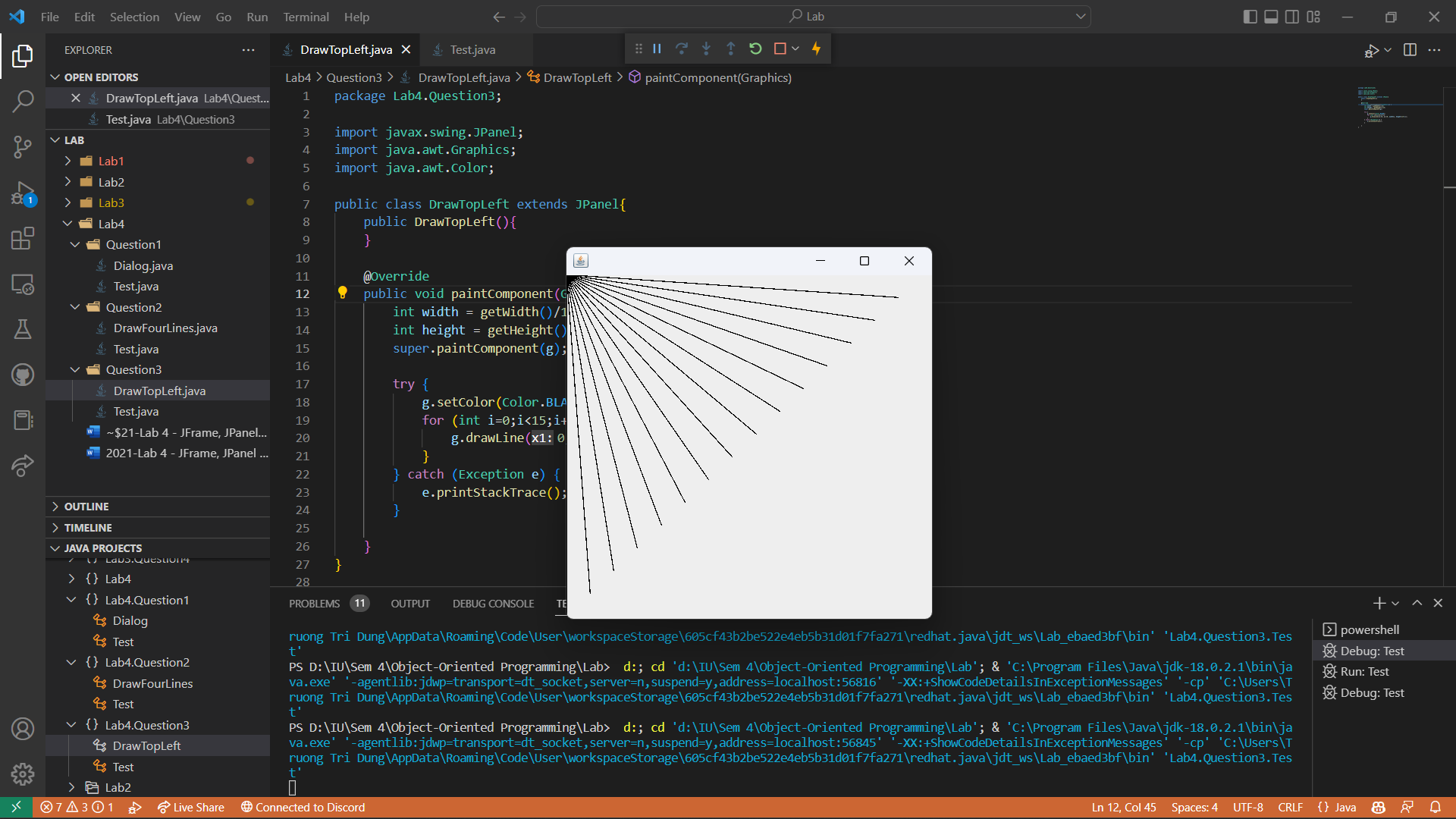
        frame.setSize(400,400);

        frame.setVisible(true);

    }

}

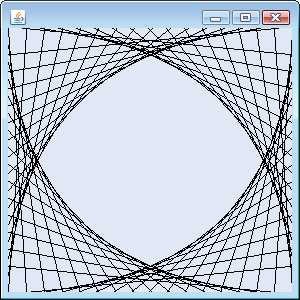
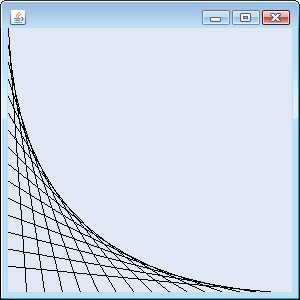
**Output:**



Graphical user interface

Description automatically generated

1. Figure 4 displays two additional designs created using while loops and drawLine.
   1. Create the design in the left screen capture of Fig. 4. Begin by dividing each edge into an equal number of increments (we chose 15 again). The first line starts in the top-left corner and ends one step right on the bottom edge. For each successive line, move down one increment on the left edge and right one increment on the bottom edge. Continue drawing lines until you reach the bottom-right corner. The figure should scale as you resize the window so that the endpoints always touch the edges.
   2. Modify your answer in part (a) to mirror the design in all four corners, as shown in the right screen capture of Fig. 4.



**Fig. 4** | Line art with loops and drawLine.

**DrawTopLeft.java:**

package Lab4.Question4;

import javax.swing.JPanel;

import java.awt.Graphics;

import java.awt.Color;

public class DrawTopLeft extends JPanel{

    public DrawTopLeft(){

    }

    @Override

    public void paintComponent(Graphics g) {

        int width = getWidth()/15;

        int height = getHeight()/15;

        super.paintComponent(g);

        try {

            g.setColor(Color.BLACK);

            int i=0;

            while (i<15) {

                g.drawLine(0, height\*i, width\*(i+1), height\*15);

                i++;

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

**DrawAllCorners.java:**

package Lab4.Question4;

import javax.swing.JPanel;

import java.awt.Graphics;

import java.awt.Color;

public class DrawAllCorners extends JPanel{

    public DrawAllCorners(){

    }

    @Override

    public void paintComponent(Graphics g) {

        int width = getWidth()/15;

        int height = getHeight()/15;

        super.paintComponent(g);

        try {

            g.setColor(Color.BLACK);

            int i=0;

            while (i<15) {

                g.drawLine(0, height\*i, width\*(i+1), getHeight());

                i++;

            }

            i=0;

            while (i<15) {

                g.drawLine(width\*(i+1), getHeight(), getWidth(), getHeight()-height\*(i+1));

                i++;

            }

            i=0;

            while (i<15) {

                g.drawLine(getWidth(), getHeight()-height\*(i+1), getWidth()-width\*(i+1), 0);

                i++;

            }

            i=0;

            while (i<15) {

                g.drawLine(getWidth()-width\*(i+1), 0, 0, height\*(i+1));

                i++;

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

**Test.java:**

package Lab4.Question4;

import javax.swing.JFrame;

public class Test {

    public static void main(String[] args) {

        //DrawTopLeft panel = new DrawTopLeft();

        DrawAllCorners panel = new DrawAllCorners();

        JFrame frame = new JFrame();

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.add(panel);

        frame.setSize(400,400);

        frame.setVisible(true);

    }

}

**Output:**

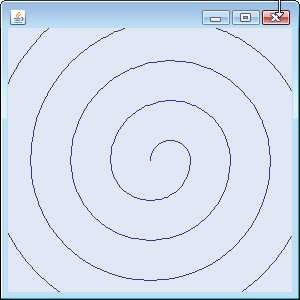
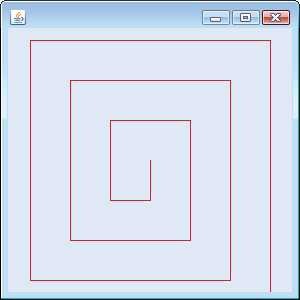
Graphical user interface

Description automatically generated

A picture containing graphical user interface

Description automatically generated

1. ***(Drawing Spirals)*** In this exercise, you will draw spirals with methods drawLine and drawArc.
   1. Draw a square-shaped spiral (as in the left screen capture of Fig. 5), centered on the panel, using method drawLine. One technique is to use a loop that increases the line length after drawing every second line. The direction in which to draw the next line should follow a distinct pattern, such as down, left, up, right.
   2. Draw a circular spiral (as in the right screen capture of Fig. 5), using method drawArc to draw one semicircle at a time. Each successive semicircle should have a larger radius (as specified by the bounding rectangle’s width) and should continue drawing where the previous semicircle finished.



**Fig. 5** | Drawing a spiral using drawLine (left) and drawArc (right).

**DrawLineSpirals.java:**

package Lab4.Question5;

import javax.swing.JPanel;

import java.awt.Graphics;

import java.awt.Color;

public class DrawLineSpirals extends JPanel{

    public DrawLineSpirals(){

    }

    @Override

    public void paintComponent(Graphics g) {

        int width = getWidth();

        int height = getHeight();

        int centerX = getWidth()/2;

        int centerY = getHeight()/2;

        int x = centerX;

        int y = centerY;

        int frame = 50;

        super.paintComponent(g);

        try {

            g.setColor(Color.BLACK);

            int state = 0; //0: down, 1: up

            while (x<width && y<height && x>0 && y>0) {

                if (state == 0) {

                    g.drawLine(x, y, x, y+frame);

                    y += frame;

                    g.drawLine(x, y, x-frame, y);

                    x -= frame;

                    state = 1;

                } else {

                    g.drawLine(x, y, x, y-frame);

                    y -= frame;

                    g.drawLine(x, y, x+frame, y);

                    x += frame;

                    state = 0;

                }

                frame += 50;

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

**DrawArcSpirals.java:**

package Lab4.Question5;

import javax.swing.JPanel;

import java.awt.Graphics;

import java.awt.Color;

import java.lang.Math;

public class DrawArcSpirals extends JPanel{

    public DrawArcSpirals(){

    }

    @Override

    public void paintComponent(Graphics g) {

        int width = getWidth();

        int height = getHeight();

        int centerX = getWidth()/2;

        int centerY = getHeight()/2;

        int frame = 50;

        int x = centerX;

        int y = centerY-frame/2;

        super.paintComponent(g);

        try {

            g.setColor(Color.BLACK);

            int state = 1; //0: down, 1: up

            while (frame/2<width/Math.sqrt(2) || frame/2<height/Math.sqrt(2)) {

                if (state == 1) {

                    g.drawArc(x, y, frame, frame, 0, 180);

                    x -= 50;

                    y -= 50/2;

                    state = 0;

                } else {

                    g.drawArc(x, y, frame, frame, 180, 180);

                    y -= 50/2;

                    state = 1;

                }

                frame += 50;

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

**Test.java:**

package Lab4.Question5;

import javax.swing.JFrame;

public class Test {

    public static void main(String[] args) {

        //DrawLineSpirals panel = new DrawLineSpirals();

        DrawArcSpirals panel = new DrawArcSpirals();

        JFrame frame = new JFrame();

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.add(panel);

        frame.setSize(400,400);

        frame.setVisible(true);

    }

}

**Output:**

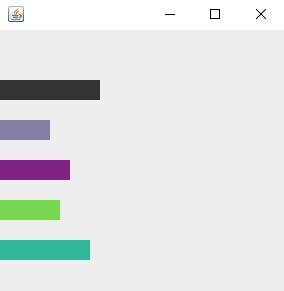
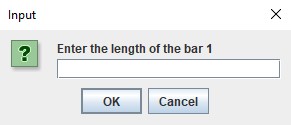
Graphical user interface

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

1. ***(Bar Chart Printing Program)*** Write an application that reads five integer numbers using dialogs. For each number that is read, your program should display the bar chart using rectangles of varying lengths. Display the rectangles *after* you read all five numbers.



**Fig. 6** | Obtaining user input and creating a JFrame to display Bar Chart.

**BarChart.java:**

package Lab4.Question6;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import java.awt.Graphics;

import java.awt.Color;

public class BarChart extends JPanel{

    private static int[] a = new int[6];

    public BarChart(){

    }

    public void display(){

        for (int i=1;i<=5;i++){

            String s = JOptionPane.showInputDialog("Enter the length of the bar "+i);

            int x = Integer.parseInt(s);

            this.a[i] = x;

        }

    }

    @Override

    public void paintComponent(Graphics g) {

        int width = getWidth();

        int height = getHeight();

        int frame = height/13;

        super.paintComponent(g);

        try {

            g.setColor(Color.BLACK);

            g.fillRect(0, 2\*frame, frame\*this.a[1], frame);

            g.setColor(Color.RED);

            g.fillRect(0, 4\*frame, frame\*this.a[2], frame);

            g.setColor(Color.BLUE);

            g.fillRect(0, 6\*frame, frame\*this.a[3], frame);

            g.setColor(Color.GREEN);

            g.fillRect(0, 8\*frame, frame\*this.a[4], frame);

            g.setColor(Color.YELLOW);

            g.fillRect(0, 10\*frame, frame\*this.a[5], frame);

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

**Test.java:**

package Lab4.Question6;

import javax.swing.JFrame;

public class Test {

    public static void main(String[] args) {

        BarChart dialog = new BarChart();

        dialog.display();

        BarChart panel = new BarChart();

        JFrame frame = new JFrame();

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.add(panel);

        frame.setSize(400,400);

        frame.setVisible(true);

    }

}

**Output:**

