**Object-Oriented Programming**

**Lab Session #4**

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.

import javax.swing.JOptionPane;

public class Question1 { //NameDialog

public static void main(String[] args) {

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

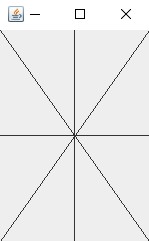
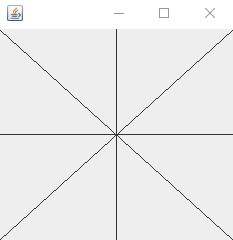
String message = String.*format*("Welcome, %s, to Java Programming ", askname);

JOptionPane.*showMessageDialog*(null, message);

}

}

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.

import java.awt.Graphics;

import javax.swing.JFrame;

import javax.swing.JPanel;

public class Question2 extends JPanel{

public static void main(String[] args) {

Question2 panel = new Question2();

JFrame window = new JFrame("Question 2");

window.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

window.add(panel);

window.setSize(350, 350);

window.setVisible(true);

}

public void paint(Graphics g) {

super.paint(g);

int x = getWidth();

int y = getHeight();

g.drawLine(0, 0, x, y);

g.drawLine(0, y, x, 0);

g.drawLine(x/2, 0, x/2 ,y);

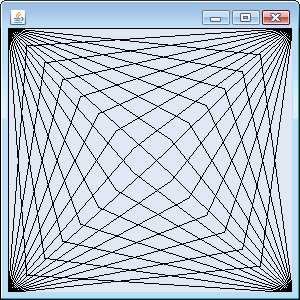
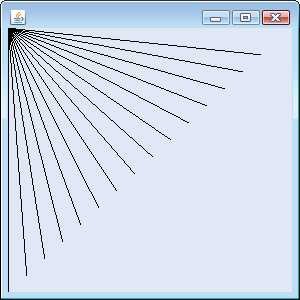
g.drawLine(0, y/2, x, y/2);

}

}

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.



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

1. 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.

import java.awt.Graphics;

import javax.swing.JPanel;

import javax.swing.JFrame;

public class Question3b extends JPanel {

public static void main(String args[]){

Question3b panel = new Question3b();

JFrame window = new JFrame("Question 3b");

window.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

window.add(panel);

window.setSize(350, 350);

window.setVisible(true);

}

public void paintComponent(Graphics g){

super.paintComponent(g);

int width = getWidth();

int height = getHeight();

for(int i = 0; i<=350; i+=30) {

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

}

for(int i = 0; i<=350; i+=30) {

g.drawLine(width , height, i, ((-height/width)\*i+height));

}

for(int i = 0; i<=350; i+=30) {

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

}

for(int i = 0; i<=350; i+=30) {

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

}

}

}

import java.awt.Graphics;

import javax.swing.JPanel;

import javax.swing.JFrame;

public class Question3a extends JPanel{

public static void main(String args[]) {

Question3a panel = new Question3a();

JFrame window = new JFrame("Question 3a");

window.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

window.add(panel);

window.setSize(350, 350);

window.setVisible(true);

}

public void paintComponent(Graphics g){

super.paintComponent(g);

int width = getWidth();

int height = getHeight();

for(int i = 0; i<=300; i+=20) {

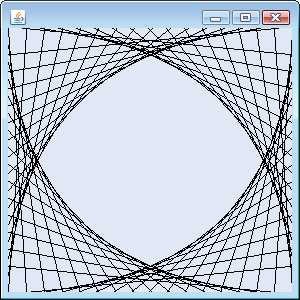
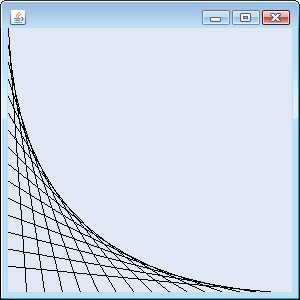
g.drawLine(0 , 0, i, ((-height/width)\*i+height));

}

}

}

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.

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.geom.Line2D;

import javax.swing.JFrame;

import javax.swing.JPanel;

public class Question4a extends JPanel{

public static void main(String []arg){

Question4a panel = new Question4a();

JFrame window = new JFrame("Question 4a");

window.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

window.add(panel);

window.setSize(300, 300);

window.setVisible(true);

}

public void paintComponent(Graphics g) {

Graphics2D l = (Graphics2D)g;

int width = getWidth();

int height = getHeight();

double counter = 0;

while(counter <= 20) {

double tyle = counter / 20;

Line2D line1 = new Line2D.Double(0, tyle \* height, tyle \* width, height);

l.draw(line1);

counter++;

}

}

}

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.geom.Line2D;

import javax.swing.JFrame;

import javax.swing.JPanel;

public class Question4b extends JPanel{

public static void main(String []arg){

Question4b panel = new Question4b();

JFrame window = new JFrame("Question 4b");

window.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

window.add(panel);

window.setSize(300, 300);

window.setVisible(true);

}

public void paintComponent(Graphics g){

Graphics2D g2 = (Graphics2D)g;

int width = getWidth();

int height = getHeight();

double counter = 0;

while(counter < 20) {

double tyle = counter / 20;

double tyle2 = (20-counter-1) / 20;

Line2D line1 = new Line2D.Double( 0, tyle \* height, tyle \* width, height );

Line2D line2 = new Line2D.Double( tyle\*width, 0, width, tyle\*height );

Line2D line3 = new Line2D.Double( width, tyle\*height, tyle2\*width, height );

Line2D line4 = new Line2D.Double( tyle2\*width, 0, 0, tyle\*height );

g2.draw(line1);

g2.draw(line2);

g2.draw(line3);

g2.draw(line4);

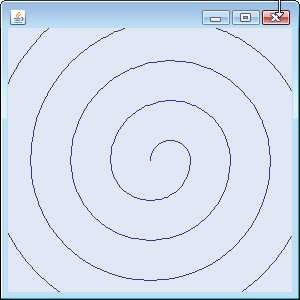
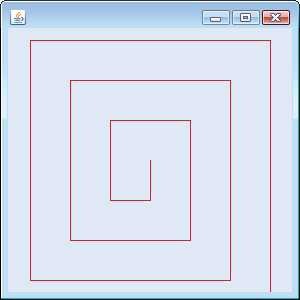
counter++;

}

}

}

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).

import java.awt.Graphics;

import javax.swing.JPanel;

import javax.swing.JFrame;

public class Question5b extends JPanel{

public static void main(String args[]){

Question5b panel = new Question5b();

JFrame window = new JFrame();

window.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

window.add(panel);

window.setSize(350, 350);

window.setVisible(true);

}

public void paintComponent(Graphics g) {

int w = getSize().width /2-10;

int h = getSize().height/2-10;

int width = 30;

int height = 30;

int counter = 0;

while(counter <= 10) {

if (counter % 2 == 0) {

h = h-10;

width = width+2\*10;

height = height+2\*10;

g.drawArc(w, h, width, height, 0, -200);

} else {

w = w-2\*10;

h = h-10;

width =width+2\*10;

height = height+2\*10;

g.drawArc(w, h, width, height, 0, 200);

}

counter++;

}

}

}

import java.awt.Color;

import java.awt.Graphics;

import javax.swing.JPanel;

import javax.swing.JFrame;

public class Question5a extends JPanel{

public static void main(String args[]){

Question5a panel = new Question5a();

JFrame window = new JFrame();

window.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

window.add(panel);

window.setSize(350, 350);

window.setVisible(true);

}

public void paintComponent(Graphics g)

{

int width = getSize().width;

int height = getSize().height;

int widthCenter = width / 2;

int heightCenter = height / 2;

int counter = 0;

while(counter <= 5 ) {

g.setColor(Color.***RED***);

g.drawLine(widthCenter + (30\*counter), heightCenter - (30\*counter), widthCenter + (30\*counter), heightCenter + 30 + (30\*counter));

g.drawLine(widthCenter + (30\*counter), heightCenter + 30 + (30\*counter), widthCenter - 30 - (30\*counter), heightCenter + 30 + (30\*counter));

g.drawLine(widthCenter - 30 - (30\*counter), heightCenter + 30 + (30\*counter), widthCenter - 30 - (30\*counter), heightCenter - 30 - (30\*counter));

g.drawLine(widthCenter - 30 - (30\*counter), heightCenter - 30 - (30\*counter), widthCenter + 30 + (30\*counter), heightCenter - 30 - (30\*counter));

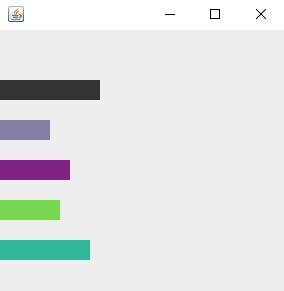
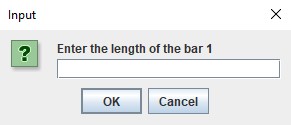
counter++;

}

}

}

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.

import java.awt.Color;

import java.awt.Graphics;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

public class Question6 extends JPanel{

private static int *length1*;

private static int *length2*;

private static int *length3*;

private static int *length4*;

private static int *length5*;

public static void main(String[] args) {

String input1 = JOptionPane.*showInputDialog*("Enter the length of the bar 1");

*length1* = Integer.*parseInt*(input1);

String input2 = JOptionPane.*showInputDialog*("Enter the length of the bar 2");

*length2* = Integer.*parseInt*(input2);

String input3 = JOptionPane.*showInputDialog*("Enter the length of the bar 3");

*length3* = Integer.*parseInt*(input3);

String input4 = JOptionPane.*showInputDialog*("Enter the length of the bar 4");

*length4* = Integer.*parseInt*(input4);

String input5 = JOptionPane.*showInputDialog*("Enter the length of the bar 5");

*length5* = Integer.*parseInt*(input5);

Question6 panel = new Question6();

JFrame window = new JFrame("Bar chart");

window.setDefaultCloseOperation( JFrame.***EXIT\_ON\_CLOSE*** );

window.add( panel );

window.setSize( 400, 400 );

window.setVisible( true );

}

public void paintComponent(Graphics g) {

g.setColor(Color.***black***);

g.fillRect(0, 50, *length1*, 20);

g.drawRect (0, 50 , *length1*, 20);

g.setColor(Color.***LIGHT\_GRAY***);

g.fillRect(0, 100, *length2*, 20);

g.drawRect (0, 100, *length2*, 20);

g.setColor(Color.***magenta***);

g.fillRect(0, 150, *length3*, 20);

g.drawRect (0, 150, *length3*, 20);

g.setColor(Color.***green***);

g.fillRect(0, 200, *length4*, 20);

g.drawRect (0, 200, *length4*, 20);

g.setColor(Color.***BLUE***);

g.fillRect(0, 250, *length5*, 20);

g.drawRect (0, 250, *length5*, 20);

}

}