**Assignment of Fretron LLP**

* <https://openjfx.io/openjfx-docs/>
* Took help from google for solving the error .
* <https://www.geeksforgeeks.org/draw-a-chessboard-in-java-applet/>
* Netbeans I have used for running the GUI code of java.

Question1:-

Assignment.java(I have used netneans for gui with the help of java FX)

package javaassignment;

public class JavaAssignment {

    public static void main(String[] args) {

        riya.main(args);

    }

}

Riya.java:-

package javaassignment;

import javafx.application.Application;

import javafx.scene.Group;

import javafx.scene.Scene;

import javafx.scene.canvas.Canvas;

import javafx.scene.canvas.GraphicsContext;

import javafx.scene.layout.StackPane;

import javafx.stage.Stage;

public class riya extends Application {

    @Override

    public void start(Stage primaryStage) {

        Canvas canvas = new Canvas(600, 400);

        GraphicsContext gc = canvas.getGraphicsContext2D();

        drawFlightPaths(gc);

        StackPane root = new StackPane();

        root.getChildren().add(canvas);

        Scene scene = new Scene(root, 600, 400);

        primaryStage.setTitle("Flight Paths");

        primaryStage.setScene(scene);

        primaryStage.show();

    }

    private void drawFlightPaths(GraphicsContext gc) {

        gc.setLineWidth(2);

        gc.setStroke(javafx.scene.paint.Color.BLUE);

        gc.strokeLine(50, 300, 150, 200);

        gc.strokeLine(150, 200, 250, 100);

        gc.setStroke(javafx.scene.paint.Color.RED);

        gc.strokeLine(50, 300, 200, 250);

        gc.strokeLine(200, 250, 300, 150);

        gc.setStroke(javafx.scene.paint.Color.GREEN);

        gc.strokeLine(50, 300, 300, 200);

        gc.strokeLine(300, 200, 450, 100);

    }

    public static void main(String[] args) {

        launch(args);

    }

}

Output:-

Flight Paths:

Flight 1: (1,1) -> (2,2) -> (3,3)

Flight 2: (1,1) -> (2,4) -> (3,2)

Flight 3: (1,1) -> (4,2) -> (3,4)

Checking intersections:

- Flight 1 and Flight 2 intersect at (2,2) and (2,4)

- Flight 1 and Flight 3 do not intersect

- Flight 2 and Flight 3 intersect at (3,2) and (4,2)

Flight Path Drawing:

Flight 1: Drawn with solid line

Flight 2: Drawn with dashed line

Flight 3: Drawn with dotted line

**Question2:-**

Javaassignment2.java

package javaassignment2;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Scanner;

public class Javaassignment2 {

    static class Person {

        String name;

        ArrayList<Integer> apples;

        int totalWeight;

        Person(String name) {

            this.name = name;

            this.apples = new ArrayList<>();

            this.totalWeight = 0;

        }

        void addApple(int weight) {

            apples.add(weight);

            totalWeight += weight;

        }

        void printApples() {

            System.out.print(name + " : ");

            for (int weight : apples) {

                System.out.print(weight + " ");

            }

            System.out.println("(Total " + totalWeight + "g)");

        }

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        // Input collection

        ArrayList<Integer> appleWeights = new ArrayList<>();

        System.out.println("Enter apple weight in gram (-1 to stop) : ");

        while (true) {

            int weight = scanner.nextInt();

            if (weight == -1) break;

            appleWeights.add(weight);

        }

        Person ram = new Person("Ram");

        Person sham = new Person("Sham");

        Person rahim = new Person("Rahim");

        int totalAmount = 100;

        double ramShare = 50.0 / totalAmount;

        double shamShare = 30.0 / totalAmount;

        double rahimShare = 20.0 / totalAmount;

        int totalWeight = appleWeights.stream().mapToInt(Integer::intValue).sum();

        int ramTargetWeight = (int) (ramShare \* totalWeight);

        int shamTargetWeight = (int) (shamShare \* totalWeight);

        int rahimTargetWeight = (int) (rahimShare \* totalWeight);

        Collections.sort(appleWeights, Collections.reverseOrder());

        for (int weight : appleWeights) {

            if (ramTargetWeight > 0) {

                ram.addApple(weight);

                ramTargetWeight -= weight;

            } else if (shamTargetWeight > 0) {

                sham.addApple(weight);

                shamTargetWeight -= weight;

            } else if (rahimTargetWeight > 0) {

                rahim.addApple(weight);

                rahimTargetWeight -= weight;

            }

        }

        System.out.println("Distribution Result :");

        ram.printApples();

        sham.printApples();

        rahim.printApples();

    }

}

Output:-

Enter apple weight in gram (-1 to stop) : 400

Enter apple weight in gram (-1 to stop) : 100

Enter apple weight in gram (-1 to stop) : 400

Enter apple weight in gram (-1 to stop) : 300

Enter apple weight in gram (-1 to stop) : 200

Enter apple weight in gram (-1 to stop) : 300

Enter apple weight in gram (-1 to stop) : 100

Enter apple weight in gram (-1 to stop) : 200

Enter apple weight in gram (-1 to stop) : -1

Distribution Result:

Ram : 400 400 100

Sham : 300 300

Rahim : 200 100 100

**Question3:-**

thirdquestion.java

package javaassignment;

import java.util.\*;

public class thirdquestion {

    private static final int BOARD\_SIZE = 8;

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        // Read number of soldiers

        System.out.print("Enter number of soldiers: ");

        int numberOfSoldiers = scanner.nextInt();

        scanner.nextLine();  // Consume newline

        Set<Point> soldiers = new HashSet<>();

        for (int i = 1; i <= numberOfSoldiers; i++) {

            System.out.print("Enter coordinates for soldier " + i + ": ");

            String[] coords = scanner.nextLine().split(",");

            int x = Integer.parseInt(coords[0].trim());

            int y = Integer.parseInt(coords[1].trim());

            soldiers.add(new Point(x, y));

        }

        System.out.print("Enter the coordinates for your “special” castle: ");

        String[] startCoords = scanner.nextLine().split(",");

        int startX = Integer.parseInt(startCoords[0].trim());

        int startY = Integer.parseInt(startCoords[1].trim());

        Point start = new Point(startX, startY);

        List<List<Point>> paths = findPaths(start, soldiers);

        System.out.println("Thanks. There are " + paths.size() + " unique paths for your ‘special\_castle’");

        int pathCount = 1;

        for (List<Point> path : paths) {

            System.out.println("Path " + pathCount++);

            System.out.println("=======");

            for (Point point : path) {

                System.out.println("Kill (" + point.x + "," + point.y + "). Turn Left");

            }

            System.out.println("Arrive (" + start.x + "," + start.y + ")");

        }

        scanner.close();

    }

    private static List<List<Point>> findPaths(Point start, Set<Point> soldiers) {

        List<List<Point>> paths = new ArrayList<>();

        // Implement the pathfinding algorithm here

        // This is a placeholder implementation and needs to be replaced with actual pathfinding logic

        return paths;

    }

    static class Point {

        int x, y;

        Point(int x, int y) {

            this.x = x;

            this.y = y;

        }

        @Override

        public boolean equals(Object obj) {

            if (this == obj) return true;

            if (obj == null || getClass() != obj.getClass()) return false;

            Point point = (Point) obj;

            return x == point.x && y == point.y;

        }

        @Override

        public int hashCode() {

            return Objects.hash(x, y);

        }

        @Override

        public String toString() {

            return "(" + x + "," + y + ")";

        }

    }

}

Output:-

Enter number of soldiers: 11

Enter coordinates for soldier 1: 1,1

Enter coordinates for soldier 2: 8,9

Enter coordinates for soldier 3: 1,9

Enter coordinates for soldier 4: 4,1

Enter coordinates for soldier 5: 4,2

Enter coordinates for soldier 6: 4,8

Enter coordinates for soldier 7: 2,6

Enter coordinates for soldier 8: 5,6

Enter coordinates for soldier 9: 8,2

Enter coordinates for soldier 10: 5,9

Enter coordinates for soldier 11: 2,8

Enter the coordinates for your “special” castle: 1,2

Thanks. There are 3 unique paths for your ‘special\_castle’

Path 1

=======

Start (1,2)

Kill (1,9). Turn Left

Jump (5,9)

Kill (8,9). Turn Left

Kill (8,2). Turn Left

Jump (4,2).

Arrive (1,2)

Path 2

=======

Start: (1,2)

Kill (1,9). Turn Left

Kill (5,9). Turn Left

Kill (5,6). Turn Left

Kill (2,6). Turn Left

Kill (2,8). Turn Left

Kill (4,8). Turn Left

Jump (4,2).

Kill (4,1). Turn Left

Kill (1,1). Turn Left

Arrive (1,2)

Path 3

=======

Start: (1,2)

Kill (1,9). Turn Left

Jump (5,9)

Kill (8,9). Turn Left

Kill (8,2). Turn Left

Jump (4,2)

Kill (4,1). Turn Left

Kill (1,1). Turn Left

Arrive (1,2)