פרויקט גמר בתכנות מקבילי ומבוזר

ניהול מקבילי של

שדה תעופה

מגישה: תמר שם טוב

ת.ז: 325112332

ביה"ס: סמינר בית יעקב צפת

מנחה: המ' אלישבע כהן

תאריך הגשה: יולי 2022

תוכן עניינים:

מבוא:

בפרויקט זה מימשתי מערכת לניהול המראות ונחיתות בשדה תעופה.

הפרויקט מדמה שדה תעופה שבו מספר מסלולי תעופה, המשמשים להמראה ולנחיתה.

כל אחד ממסלולים אלו הוא משאב משותף שחולקות הטיסות, ויש להקפיד על מקסימום טיסה אחת במסלול בכל רגע נתון, כדי למנוע תאונות חלילה.

בשדה התעופה מתקבלות בקשות מטיסות הרוצות להמריא / לנחות בשדה, ולצורך כך מבקשות שימוש במסלול. בקשה כזו נדרשת לכלול את המידע הרלוונטי – שם הטיסה והכיוון אליו היא יוצאת / ממנו היא מגיעה, כדי להתאים לה את המסלול הקרוב ביותר לכיוון זה. כעת הטיסה ממתינה לאישור שהיא יכולה להשתמש במסלול.

לאחר קבלת הבקשה מהטיסה – המערכת מוצאת את המסלול הקרוב ביותר לכיוון שהתקבל ומגדירה אותו כמסלול אליו הטיסה רוצה להיכנס, ומכניסה את פרטי הטיסה לרשימת הטיסות הממתינות.

כדי להפעיל טיסה מסוימת – נועלת המערכת את המשאב המשותף – מסלול התעופה המשויך לטיסה זו. הוא נשאר נעול במשך כל הזמן שהטיסה שוהה בו (לכל מסלול מוגדר זמן הנסיעה שבו), ולאחר מכן הוא משתחרר והטיסה מוסרת מרשימת ההמתנה.

באופן זה, המטוסים יוכלו להמריא ולנחות בנחת, במקביליות ובלי חשש בעז"ה.

מדריך למשתמש:

יש להריץ את הטופס **AirPortGUI**.

בטופס זה ניתן לראות:

במרכז הטופס: 4 מלבנים המייצגים את מסלולי התעופה, אחד לכל מסלול.

בצד שמאל: תיבת טקסט בה מופיעות טיסות נכנסות הממתינות להיכנס למסלול, כולל תקציר פרטים טכניים לכל טיסה – הכיוון אליו מיועדות פניה / ממנו היא מגיעה, והפעולה לביצוע – המראה או נחיתה.

בצד ימין ניתן לראות את הטיסות שסיימו את השימוש במסלולים.

רשימות אלו מתעדכנות בזמן אמת.

יש להריץ את הממשק ע"י לחיצה על כפתור start.

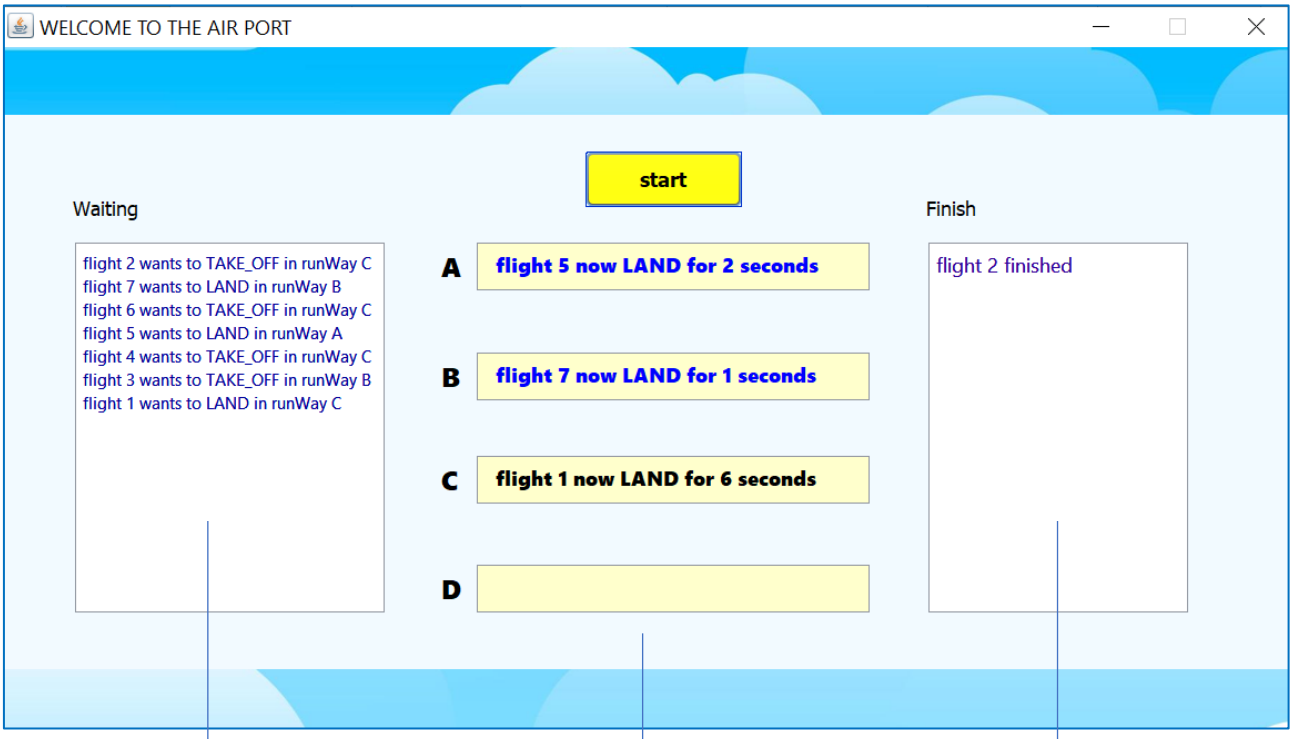
כעת ניתן לראות את רשימות הטיסות הממתינות ואלו שסיימו, ואת פרטי הטיסה הפעילה שבכל מסלול, כולל משך הזמן שנשאר לה עד לסיומו.

כאשר טיסה מסיימת את המסלול – פרטיה יופיעו מיד ברשימת הטיסות שגמרו, והטיסה הרלוונטית הבאה תיכנס למסלול. כדי להדגיש את החלפת הטיסות – מתחלף צבע הכתב שבכל מסלול עם החלפת טיסה, מצבע כחול לשחור ולהפך.

טופס זה ממחיש את ניהול פעולתם המקבילית של מסלולי התעופה.

תיאור מסכים

המסך הראשי לאחר לחיצה על כפתור start המפעיל את שדה התעופה:



הטיסות שממתינות להכנס למסלולי התעופה

הטיסות שסיימו להשתמש במסלולי התעופה

מסלולי התעופה והטיסה הפעילה בכל אחד מהם ברגע זה. מס' השניות שנותרו לכל טיסה במסלול יורד בכל שניה, וכשמגיע ל 0 – מתחלפת הטיסה במסלול זה וצבע הכתב משתנה.

פירוט המחלקות

Enum

Action: מתאר את הפעולה לביצוע – המראה או נחיתה

ערכים אפשריים:

TAKE_OFF	המראה
LAND	נחיתה

Direction: מתאר את 8 הכיוונים האפשריים של הטיסות

ערכים אפשריים:

NORTH	צפון
SOUTH	דרום
EAST	מזרח
WEST	מערב
NORTH_WEST	צפון-מערב
NORTH_EAST	צפון-מזרח
SOUTH_WEST	דרום-מערב
SOUTH_EAST	דרום-מזרח

מחלקות

מחלקת CustomOutputStream יורשת מ OutputStream – מחלקה המשנה את מיקום הדפסת הפלט – מהקונסול לתיבת טקסט נבחרת.

פירוט	שדה
תיבת טקסט אליה יודפס הפלט, במקום לקונסול	private JTextArea textArea;

פירוט	פונקציה
פעולה בונה המקבלת את תיבת הטקסט אליה יש להדפיס	public CustomOutputStream(JTextArea textArea)
פעולה הכותבת לתיבת הטקסט במקום לקונסול	@Override public void write(int b)

מחלקת RunWays – מחלקה המייצגת מסלול תעופה:

פירוט	שדה
מזהה המסלול	private String runwayID;
זמן הנסיעה במסלול	private long lengthInMinutes;
הטיסה הפעילה במסלול זה כרגע	public Flights currentFlight;
תיבת הטקסט בטופס הראשי המייצגת את מסלול זה	public JTextArea textArea;
מילון המתאר את מרחק הקצה החיצוני של המסלול מכל אחד מ 8 הכיוונים	public Map<String,String> distancesDictionary=new HashMap;()<>

פירוט	פונקציה
פעולה בונה ריקה	public RunWays()
פעולה בונה המאתחלת את משתני המחלקה	public RunWays(String runwayID,long lengthInMinutes,JTextArea textArea)
פונקציה המוסיפה ערך נוסף למילון המרחקים	public void AddItemToDistancesDictionay(Directions direction,long distance)
פונקציה המחזירה את מזהה המסלול	public String getRunwayID()
פונקציה המחזירה את זמן הנסיעה במסלול	public long getLengthInMinutes()
פונקציה המחזירה את מילון המרחקים של המסלול	public Map<String, String> getDistancesDictionary()
פונקציה המקבלת טיסה ומגדירה אותה כטיסה הנוכחית במסלול זה	public void SetCurrentFlight(Flights currentFlight)
פונקציה המחזירה את הטיסה הנוכחית שבמסלול	public Flights getCurrentFlight()

מחלקת Flights – מתארת טיסה. המחלקה מממשת את הממשק Runnable.

פירוט	שדה
שם טיסה	private String flightName;

private AirPort airPort;	שדה התעופה אליו מקושרת טיסה זו (יפורט בהמשך)
private Directions direction;	כיוון הטיסה – לאן היא יוצאת / מהיכן היא נכנסת
private Action action;	הפעולה אותה יש לבצע – המראה / נחיתה
private RunWays ClosestRunWay;	מסלול התעופה הקרוב ביותר לטיסה זו, מחושב ע"י פונקציה שתפורט בהמשך
private int priority;	עדיפות הטיסה במספרים, כאשר נכנסת בקשה של טיסה חדשה – מועלית העדיפות של כל יתר הטיסות. מקסימום עדיפות – 10.
static PrintStream waitingFlightsTextArea;	משתנה סטטי המצביע לתיבת הטקסט אליה יש להדפיס את הטיסות הממתינות
static PrintStream finishedFlightsTextArea;	משתנה סטטי המצביע לתיבת הטקסט אליה יש להדפיס את הטיסות שסיימו

פונקציה	תיאור
public Flights(String flightName, Directions direction, Action action,AirPort airPort, int priority)	פעולה בונה המאתחלת את משתני המחלקה
public String getFlightName()	פונקציה המחזירה את שם הטיסה
public void run()	פונקציה המפעילה את הטיסה: מדפיסה את פרטי הטיסה בתיבת הטקסט של הטיסות הממתינות, מגדירה את המסלול הקרוב לטיסה, מכניסה את הטיסה למסלול זה באופן מסונכרן, מדפיסה את הפרטים לאחר הסיום, ומסירה את הטיסה מרשימת הטיסות שבשדה התעופה.

מחלקת AirPort – מחלקה המייצגת שדה תעופה.

שדה	פירוט
private String name;	שם שדה התעופה
public ArrayList<RunWays> runWays;	רשימת מסלולי התעופה שבשדה זה

רשימת הטיסות שבשדה זה. טיסה שגמרה להמריא / לנחות – מוסרת מהרשימה.	private ArrayList<Flights> flights;

פירוט	פונקציה
פעולה בונה המאתחלת את המשתנים	public AirPort(String name)
פונקציה המקבלת מסלול תעופה חדש ומוסיפה אותו לרשימת המסלולים בשדה	public void AddRunWay(RunWays newRunWay)
פונקציה המקבלת טיסה חדשה ומוסיפה אותה לרשימת הטיסות שבשדה	public void AddFlight(Flights newFlight)
פונקציה המקבלת כיוון כלשהו ומחזירה את המסלול הקרוב ביותר לכיוון זה	public RunWays FindClosestRunWay(Directions direction)
פונקציה המחזירה את הטיסות שבשדה	public ArrayList<Flights> getFlights()
פונקציה המקבלת טיסה ומסירה אותה מרשימת הטיסות	public void removeFlight(Flights f)

מחלקת AirPortGUI – המסך הראשי:

פירוט	פונקציה
פונקציית לחיצה על כפתור start – הפונקציה מוסיפה מס' טיסות לשדה התעופה ומפעילה אותן	private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)

קוד המחלקות

מחלקת CustomOutputStream

```
public class CustomOutputStream extends OutputStream{
    private JTextArea textArea;

    public CustomOutputStream(JTextArea textArea) {
        this.textArea = textArea;
    }
```

```
    @Override
    public void write(int b) throws IOException {
        // redirects data to the text area
        textArea.append(String.valueOf((char)b));
        // scrolls the text area to the end of data
        textArea.setCaretPosition(textArea.getDocument().getLength());
        // keeps the textArea up to date
        textArea.update(textArea.getGraphics());
    }
}
```

מחלקת RunWays:

```
public class RunWays {
    private String runwayID;
    private long lengthInMinutes;
    private Flights currentFlight;
    private Map<String,String> distancesDictionary=new HashMap<>();

    public JTextArea textArea;

    public RunWays() { }

    public RunWays(String runwayID,long lengthInMinutes,JTextArea textArea) {
        this.runwayID = runwayID;
        this.lengthInMinutes=lengthInMinutes;
        this.textArea=textArea;
    }

    public void AddItemToDistancesDictionay(Directions direction,long distance){
        this.distancesDictionary.put(direction.toString(),
String.valueOf(distance));
    }

    public String getRunwayID() {
        return runwayID;
    }

    public long getLengthInMinutes() {
```

```
        return lengthInMinutes;
    }

    public Map<String, String> getDistancesDictionary() {
        return distancesDictionary;
    }

    public void SetCurrentFlight(Flights currentFlight){
        this.currentFlight=currentFlight;
    }

    public Flights getCurrentFlight() {
        return currentFlight;
    }

    public JTextArea getTextArea() {
        return textArea;
    }

    public void setCurrentFlight(Flights currentFlight) {
        this.currentFlight = currentFlight;
    }

    public void setTextArea(JTextArea textArea) {
        this.textArea = textArea;
    }

    public void setDistancesDictionary(Map<String, String> distancesDictionary) {
        this.distancesDictionary = distancesDictionary;
    }
}
```

מחלקת Flights:

```
public class Flights extends Thread{
    private String flightName;
    private AirPort airPort;
    private Directions direction;
    private Action action;
```

```
    private RunWays ClosestRunWay;
    private int priority;

    static PrintStream waitingFlightsTextArea;
    static PrintStream finishedFlightsTextArea;


    public Flights(String flightName, Directions direction, Action action,AirPort
airPort, int priority) {
        this.flightName=flightName;
        this.direction = direction;
        this.action = action;
        this.airPort=airPort;
        this.ClosestRunWay=airPort.FindClosestRunWay(direction);
        this.priority=priority;
    }

    public String getFlightName() {
        return flightName;
    }

    public void run(){
        try {
             //מדפיס הודעה שהטיסה ממתינה בתיבת הטקסט של טיסות ממתינות
           System.setOut(waitingFlightsTextArea);
           System.out.println("flight "+flightName+" wants to "+ action +" in runWay
"+ClosestRunWay.getRunwayID());

            this.ClosestRunWay=airPort.FindClosestRunWay(direction);
        //נועל את המסלול הנדרש ומדפיס בו את פרטי הטיסה הנוכחית
        synchronized(this.ClosestRunWay){
            if (ClosestRunWay.textArea.getForeground()==Color.BLUE){
                ClosestRunWay.textArea.setForeground(Color.BLACK);
            }
            else{
                ClosestRunWay.textArea.setForeground(Color.BLUE);
            }
```

```
            long length=ClosestRunWay.getLengthInMinutes();
            //מוריד את מספר השניות שנותרו לטיסה במסלול זה באחד כל שניה, ומדפיס
            while(length>0)
            {
                ClosestRunWay.textArea.setText("  flight "+flightName +" now
"+action+"for"+TimeUnit.MILLISECONDS.toSeconds(length)+"seconds");

                sleep(1000);
                length-=1000;
            }
            ClosestRunWay.textArea.setText("");
            this.airPort.removeFlight(this);
        }
        //מדפיס את פרטי הטיסה שהסתימה כרגע בתיבת הטקסט של טיסות שסיימו
        System.setOut(finishedFlightsTextArea);
        System.out.println("flight "+flightName+" finished");
        }

        catch (InterruptedException ex) {
            Logger.getLogger(Flights.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

מחלקת AirPort:

```
public class AirPort{

    private String name;
    public ArrayList<RunWays> runWays;
    private ArrayList<Flights> flights;

    public AirPort(String name) {
        this.name = name;
        this.flights=new ArrayList<>();
        this.runWays=new ArrayList<>();
    }

    public void AddRunWay(RunWays newRunWay){
```

```
        this.runWays.add(newRunWay);
    }

    public void AddFlight(Flights newFlight){
        for (Flights flight : flights) {
            flight.setPriority(Math.min(flight.getPriority()+1,10));
        }
        this.flights.add(newFlight);
        newFlight.start();
    }

    public RunWays FindClosestRunWay(Directions direction){
        long minDistance=runWays.get(0).getLengthInMinutes(),currentDistance;
        String a;
        RunWays minDistanceRunWay=new RunWays();

        for (RunWays runWay : runWays) {
            if(runWay.getCurrentFlight()==null){
                a=runWay.getDistancesDictionary().get(direction.toString());
                currentDistance=Long.parseUnsignedLong(a);
                minDistance=Math.min(minDistance, currentDistance);
                if(minDistance==currentDistance)
                    minDistanceRunWay=runWay;
            }
        }

        if(minDistanceRunWay==null){
            for (RunWays runWay : runWays) {
                a=runWay.getDistancesDictionary().get(direction.toString());
                currentDistance=Long.parseUnsignedLong(a);
                minDistance=Math.min(minDistance, currentDistance);
                if(minDistance==currentDistance)
                    minDistanceRunWay=runWay;
            }
        }
        return minDistanceRunWay;
    }
```

```
    public ArrayList<Flights> getFlights() {
        return flights;
    }

    public void removeFlight(Flights f)
    {
        this.flights.remove(f);
    }
}
```

מחלקת AirPortGUI:

לחיצה על כפתור START:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        RunWays runWay1=new RunWays("A", 9990,routeA_TextArea);
        RunWays runWay2=new RunWays("B", 8787,routeB_TextArea);
        RunWays runWay3=new RunWays("C", 6787,routeC_TextArea);
        RunWays runWay4=new RunWays("D", 10000,routeD_TextArea);

        runWay1.AddItemToDistancesDictionay(Directions.NORTH, 13);
        runWay1.AddItemToDistancesDictionay(Directions.WEST, 35);
        runWay1.AddItemToDistancesDictionay(Directions.EAST, 12);
        runWay1.AddItemToDistancesDictionay(Directions.SOUTH, 13);
        runWay1.AddItemToDistancesDictionay(Directions.NORTH_EAST, 6);
        runWay1.AddItemToDistancesDictionay(Directions.NORTH_WEST, 3);
        runWay1.AddItemToDistancesDictionay(Directions.SOUTH_EAST, 23);
        runWay1.AddItemToDistancesDictionay(Directions.SOUTH_WEST, 65);

        runWay2.AddItemToDistancesDictionay(Directions.NORTH, 13);
        runWay2.AddItemToDistancesDictionay(Directions.WEST, 56);
        runWay2.AddItemToDistancesDictionay(Directions.EAST, 12);
        runWay2.AddItemToDistancesDictionay(Directions.SOUTH, 43);
        runWay2.AddItemToDistancesDictionay(Directions.NORTH_EAST, 65);
        runWay2.AddItemToDistancesDictionay(Directions.NORTH_WEST, 2);
        runWay2.AddItemToDistancesDictionay(Directions.SOUTH_EAST, 3);
        runWay2.AddItemToDistancesDictionay(Directions.SOUTH_WEST, 5);
```

```
        runWay3.AddItemToDistancesDictionay(Directions.NORTH, 3);
        runWay3.AddItemToDistancesDictionay(Directions.WEST, 54);
        runWay3.AddItemToDistancesDictionay(Directions.EAST, 2);
        runWay3.AddItemToDistancesDictionay(Directions.SOUTH, 3);
        runWay3.AddItemToDistancesDictionay(Directions.NORTH_EAST, 78);
        runWay3.AddItemToDistancesDictionay(Directions.NORTH_WEST, 20);
        runWay3.AddItemToDistancesDictionay(Directions.SOUTH_EAST, 13);
        runWay3.AddItemToDistancesDictionay(Directions.SOUTH_WEST, 25);

        runWay4.AddItemToDistancesDictionay(Directions.NORTH, 13);
        runWay4.AddItemToDistancesDictionay(Directions.WEST, 56);
        runWay4.AddItemToDistancesDictionay(Directions.EAST, 56);
        runWay4.AddItemToDistancesDictionay(Directions.SOUTH, 14);
        runWay4.AddItemToDistancesDictionay(Directions.NORTH_EAST, 6);
        runWay4.AddItemToDistancesDictionay(Directions.NORTH_WEST, 9);
        runWay4.AddItemToDistancesDictionay(Directions.SOUTH_EAST, 1);
        runWay4.AddItemToDistancesDictionay(Directions.SOUTH_WEST, 8);

        airPort.AddRunWay(runWay1);
        airPort.AddRunWay(runWay2);
        airPort.AddRunWay(runWay3);
        airPort.AddRunWay(runWay4);

        Flights.waitingFlightsTextArea=new PrintStream(new
CustomOutputStream(FlightNo));
        Flights.finishedFlightsTextArea=new PrintStream(new
CustomOutputStream(finishFlights));

        Flights flight1=new Flights("1", Directions.NORTH, Action.LAND, airPort,0);
        Flights flight2=new Flights("2", Directions.EAST, Action.TAKE_OFF,
airPort,0);
        Flights flight3=new Flights("3",Directions.SOUTH_WEST, Action.TAKE_OFF,
airPort,0);
        Flights flight4=new Flights("4",Directions.NORTH, Action.TAKE_OFF,
airPort,0);
        Flights flight5=new Flights("5", Directions.WEST, Action.LAND, airPort,0);
        Flights flight6=new Flights("6",Directions.SOUTH, Action.TAKE_OFF,
airPort,0);
```

```
        Flights flight7=new Flights("7",Directions.NORTH_WEST, Action.LAND,
airPort,0);

        airPort.AddFlight(flight1);
        airPort.AddFlight(flight2);
        airPort.AddFlight(flight3);
        airPort.AddFlight(flight4);
        airPort.AddFlight(flight5);
        airPort.AddFlight(flight6);
        airPort.AddFlight(flight7);
    }

//שדה התעופה בו נמצאים
AirPort airPort=new AirPort("PARALLEL_AIRPORT");
```

פעולה ראשית הבונה את הטופס:

```
    public static void main(String args[]) {
        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable()
        {
            public void run()
            {
               AirPortGUI airPortGUI=new AirPortGUI();
               airPortGUI.setVisible(true);
               airPortGUI.pack();
            }
        });
    }
```

עמדת פיתוח:

WINDOWS 10

NetBeans

ביבליוגרפיה

← Threads

https://en.wikipedia.org/wiki/Thread_(computing)

/https://www.geeksforgeeks.org/java-threads

← DeadLock

/https://www.geeksforgeeks.org/introduction-of-deadlock-in-operating-system

https://en.wikipedia.org/wiki/Deadlock

https://www.hamichlol.org.il/%D7%A7%D7%99%D7%A4%D7%90%D7%95%D7%9F_(%D7%9E%D7%93%D7%A2%D7%99_%D7%94%D7%9E%D7%97%D7%A9%D7%91)

← פתרון שגיאות

/https://stackoverflow.com

← בניית התצוגה

/https://www.geeksforgeeks.org/java-swing-jpanel-with-examples

/https://github.com

https://docs.oracle.com/javase/tutorial/uiswing/components/panel.html

https://www.javatpoint.com/java-jpanel

תודה רבה!