import java.awt.Color;

import java.awt.Graphics;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import javax.swing.JFrame;

public class Main

{

                static int minimumArrivalTime,sumCPUBurstTime;

                static int lengthOfEachBlock;

                static final int SCREEN\_WIDTH=700,SCREEN\_HEIGHT=280;

                static final int rectangleUpperPadding=50,rectangleHeight=100;

                static int numberOfProcesses;

                static int CPUBurstTime[],arrivalTime[];

                static BufferedReader br;

                static Main obj;

                Main()

                {

                                this.obj=this;

                }

                public static void main(String[] args) throws NumberFormatException, IOException

                {

                                br=new BufferedReader(new InputStreamReader(System.in));

                                System.out.println("Enter the number of processes : ");

                                numberOfProcesses=Integer.parseInt(br.readLine());

                                CPUBurstTime=new int[numberOfProcesses];

                                arrivalTime=new int[numberOfProcesses];

                                for(int i=0;i<numberOfProcesses;i++)

                                {

                                                System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

                                                System.out.println("Enter the data for the process "+(i+1));

                                                System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

                                                System.out.print("Enter the arrival Time : ");

                                                arrivalTime[i]=Integer.parseInt(br.readLine());

                                                System.out.print("Enter the CPU Burst Time : ");

                                                CPUBurstTime[i]=Integer.parseInt(br.readLine());

                                }

                    drawGanttChart();

                }

                public static void drawGanttChart() throws NumberFormatException, IOException

                {

                    int choice;

                    sumCPUBurstTime=0;

                    /\* calculating the sum of all cpu burst time \*/

                    for(int i=0;i<numberOfProcesses;i++)

                    {

                        sumCPUBurstTime+=CPUBurstTime[i];

                    }

                    /\* now the total width of the screen is to be divided into sumCPUBurstTime equal parts \*/

                    lengthOfEachBlock=SCREEN\_WIDTH/sumCPUBurstTime;

                    /\*

                     \* claculating the minimum arrival time

                     \*/

                    minimumArrivalTime=Integer.MAX\_VALUE;

                    for(int i=0;i<numberOfProcesses;i++)

                    {

                                if(minimumArrivalTime>arrivalTime[i])

                                                minimumArrivalTime=arrivalTime[i];

                    }

                    /\* asking the user which gantt chart do you want \*/

                    System.out.println("YOU HAVE THE FOLLOWING CHOICES : \n");

                    System.out.println("1. Draw Gantt Chart for FCFS Algorithm");

                    System.out.println("2. Draw Gantt Chart for Non-Preemptive SJF Algorithm");

                    System.out.println("3. Draw Gantt Chart for Preemptive SJF Algorithm");

                    System.out.println("4. Exit");

                                while(true)

                    {

                        System.out.println("ENTER YOUR CHOICE : ");

                        choice=Integer.parseInt(br.readLine());

                        switch(choice)

                        {

                            case 1:

                                drawGanttChartForFCFS();

                                break;

                            case 2:

                                drawGanttChartForNonPreemptiveSJF();

                                break;

                            case 3:

                                drawGanttChartForPreemptiveSJF();

                                break;

                            case 4:

                                System.exit(0);

                            default:

                                System.out.println("You Entered a wrong Choice\nPlease fill in the choice again...");

                        }

                    }

                }

                public static void drawGanttChartForFCFS()

                {

                                new FrameForFCFS(obj);

                }

                public static void drawGanttChartForNonPreemptiveSJF()

                {

                                new FrameForNonPreemptiveSJF(obj);

                }

                public static void drawGanttChartForPreemptiveSJF()

                {

                                new FrameForPreemptiveSJF(obj);

                }

}

class FrameForFCFS extends JFrame

{

                int arrivalTime[];

                Main obj;

                FrameForFCFS(Main obj)

                {

                                super("FCFS");

                                this.obj=obj;

                                //this.setResizable(false);

                                this.setVisible(true);

                                this.setSize(obj.SCREEN\_WIDTH+100, obj.SCREEN\_HEIGHT);

                                arrivalTime=obj.arrivalTime.clone();

                }

                @Override

                public void paint(Graphics g)

                {

                                super.paint(g);

                                this.getContentPane().setBackground(Color.white);

                                int currentTime=obj.minimumArrivalTime;

                                arrivalTime=obj.arrivalTime.clone();

                                int i,j,min,mini = 0;

                    int leftStart=50;

                    g=this.getContentPane().getGraphics();

                    g.drawString(""+obj.minimumArrivalTime,leftStart,obj.rectangleUpperPadding+obj.rectangleHeight+20);

                    for(j=0;j<obj.numberOfProcesses;j++)

                    {

                        min=Integer.MAX\_VALUE;

                        for(i=0;i<obj.numberOfProcesses;i++)

                        {

                            if(min>arrivalTime[i])

                            {

                                min=arrivalTime[i];

                                mini=i;

                            }

                        }

                        arrivalTime[mini]=Integer.MAX\_VALUE;

                        g=this.getContentPane().getGraphics();

                        g.drawRect(leftStart,obj.rectangleUpperPadding,obj.lengthOfEachBlock\*obj.CPUBurstTime[mini],obj.rectangleHeight);

                        g.drawString("P"+(mini+1),leftStart+5,obj.rectangleUpperPadding+50);

                        leftStart+=obj.lengthOfEachBlock\*obj.CPUBurstTime[mini];

                        currentTime+=obj.CPUBurstTime[mini];

                        g.drawString(""+currentTime,leftStart,obj.rectangleUpperPadding+obj.rectangleHeight+20);

                    }

                }

}

class FrameForNonPreemptiveSJF extends JFrame

{

                int CPUBurstTime[];

                Main obj;

                FrameForNonPreemptiveSJF(Main obj)

                {

                                super("Non preemptive SJF");

                                System.out.println("hello lavish kothari");

                                this.obj=obj;

                                //this.setResizable(false);

                                this.setVisible(true);

                                this.setSize(obj.SCREEN\_WIDTH+100, obj.SCREEN\_HEIGHT);

                                CPUBurstTime=obj.CPUBurstTime.clone();

                }

                @Override

                public void paint(Graphics g)

                {

                                super.paint(g);

                                this.getContentPane().setBackground(Color.white);

                                int currentTime=obj.minimumArrivalTime;

                                CPUBurstTime=obj.CPUBurstTime.clone();

                                int i,j,min,mini = 0;

                    int leftStart=50;

                    g=this.getContentPane().getGraphics();

                    g.drawString(""+obj.minimumArrivalTime,leftStart,obj.rectangleUpperPadding+obj.rectangleHeight+20);

                    for(j=0;j<obj.numberOfProcesses;j++)

                    {

                        min=Integer.MAX\_VALUE;

                        for(i=0;i<obj.numberOfProcesses;i++)

                        {

                            if(min>CPUBurstTime[i] && obj.arrivalTime[i]<=currentTime)

                            {

                                min=CPUBurstTime[i];

                                mini=i;

                            }

                        }

                        g=this.getContentPane().getGraphics();

                        g.drawRect(leftStart,obj.rectangleUpperPadding,obj.lengthOfEachBlock\*obj.CPUBurstTime[mini],obj.rectangleHeight);

                        g.drawString("P"+(mini+1),leftStart+5,obj.rectangleUpperPadding+50);

                        leftStart+=obj.lengthOfEachBlock\*obj.CPUBurstTime[mini];

                        currentTime+=obj.CPUBurstTime[mini];

                        g.drawString(""+currentTime,leftStart,obj.rectangleUpperPadding+obj.rectangleHeight+20);

                        CPUBurstTime[mini]=Integer.MAX\_VALUE;

                    }

                }

}

class FrameForPreemptiveSJF extends JFrame

{

                int CPUBurstTime[];

                Main obj;

                FrameForPreemptiveSJF(Main obj)

                {

                                super("Preemptive SJF");

                                System.out.println("hello lavish kothari");

                                this.obj=obj;

                                //this.setResizable(false);

                                this.setVisible(true);

                                this.setSize(obj.SCREEN\_WIDTH+100, obj.SCREEN\_HEIGHT);

                                CPUBurstTime=obj.CPUBurstTime.clone();

                }

                @Override

                public void paint(Graphics g)

                {

                                super.paint(g);

                                CPUBurstTime=obj.CPUBurstTime.clone();

                                System.out.println("paint called");

                                this.getContentPane().setBackground(Color.white);

                                int currentTime=obj.minimumArrivalTime;

                                int min,mini=0,prevmini = 0;

                                int leftStart=50;

                                g=this.getContentPane().getGraphics();

                    g.drawString(""+obj.minimumArrivalTime,leftStart,obj.rectangleUpperPadding+obj.rectangleHeight+20);

                                for(int j=0;j<obj.sumCPUBurstTime;j++)

                                {

                                                min=Integer.MAX\_VALUE;

                        for(int i=0;i<obj.numberOfProcesses;i++)

                        {

                            if(min>CPUBurstTime[i] && obj.arrivalTime[i]<=currentTime && CPUBurstTime[i]!=0)

                            {

                                min=CPUBurstTime[i];

                                mini=i;

                            }

                        }

                        if(j==0)

                prevmini=mini;

                        if(prevmini!=mini || j==obj.sumCPUBurstTime-1)

                        {

                                        g=this.getContentPane().getGraphics();

                                        if(j==obj.sumCPUBurstTime-1)

                                                        g.drawRect(leftStart,obj.rectangleUpperPadding,obj.lengthOfEachBlock\*(currentTime+1),obj.rectangleHeight);

                                        else

                                                        g.drawRect(leftStart,obj.rectangleUpperPadding,obj.lengthOfEachBlock\*(currentTime),obj.rectangleHeight);

                                        g.drawString("P"+(prevmini+1),leftStart+5,obj.rectangleUpperPadding+50);

                                        leftStart+=obj.lengthOfEachBlock\*currentTime;

                                        if(j==obj.sumCPUBurstTime-1)

                                                        g.drawString(""+(currentTime+1),leftStart+obj.lengthOfEachBlock,obj.rectangleUpperPadding+obj.rectangleHeight+20);

                                        else

                                                g.drawString(""+(currentTime),leftStart,obj.rectangleUpperPadding+obj.rectangleHeight+20);

                        }

                        currentTime++;

                        CPUBurstTime[mini]--;

                        prevmini=mini;

                                }

                }

}