```
/**********************************************************************
******
B1:Write a Java program to implement following scheduling algorithms
First Come First Serve (FCFS)(Non-Pre-emptive), Shortest Remaining
Time First (SRTF) (Pre-emptive)

To implement pre-emptive and non-pre-emptive CPU scheduling
algorithms.
**********************************************************************
******/
import java.util.*;
import java.util.Scanner;
class CPUscheduling //creating a cpuscheduling class
{
   void FCFS()
   { //creating a fcfs method of cpu scheduling class
     Scanner sc = new Scanner(System.in);
     System.out.println("Enter the no of processes: ");//accepting the
no. of processes
      int n = sc.nextInt();
      int Pid[] = new int[n]; // process ids
      int AT[] = new int[n]; // arrival times
      int BT[] = new int[n]; // burst or execution times
      int FT[] = new int[n]; // completion times
      int TT[] = new int[n]; // turn around times
      int WT[] = new int[n]; // waiting times
      int temp;
      float totalWT=0,totalTT=0;
      for(int i = 0; i < n; i++)
      {
        System.out.print("Enter Arrival time for process "+(i+1)+" :
");//accepting arrival time for respective process
        AT[i] = sc.nextInt();
        System.out.print("Enter burst time for process "+(i+1)+" :
");//accepting Burst time for respective process
        BT[i] = sc.nextInt();
        Pid[i] = i+1;
       }
       for(int i = 0 ; i <n; i++) //sort according to arrival times
       {
        for(int j=0; j < n-(i+1) ; j++)
        {
          if( AT[j] > AT[j+1] )
          {
            temp = AT[j];
            AT[j] = AT[j+1];
            AT[j+1] = temp;
            temp = BT[j];
            BT[j] = BT[j+1];
            BT[j+1] = temp; temp = Pid[j];
            Pid[j] = Pid[j+1];
            Pid[j+1] = temp;
           }
          }
        }
  for(int i = 0 ; i < n; i++) //to get the completion time
  {
    if( i == 0)
     {
      FT[i] = AT[i] + BT[i];
```

```
     }
    else
     {
       if( AT[i] > FT[i-1])
       {
        FT[i] = AT[i] + BT[i];
       }
       else
       {
          FT[i] = FT[i-1] + BT[i];
        }
      }
TT[i] = FT[i] - AT[i] ; // Turn-around time= completion time- arrival
time
WT[i] = TT[i] - BT[i] ; // Waiting time= Turn-around time- burst time
totalWT += WT[i] ; // Total waiting time
totalTT += TT[i] ; // Total Turn-around time
}
System.out.println("*****************************************************
*****************************");
System.out.println("\nPid\tAT\tBT\tFT\tTT\tWT");
System.out.println("*****************************************************
*****************************");
for(int i = 0 ; i< n; i++)
{
System.out.println(Pid[i] + " \t " + AT[i] + "\t" + BT[i] + "\t" + FT[i]
+ "\t" +
TT[i] + "\t" + WT[i] ) ;
}
System.out.println("*****************************************************
*****************************");
System.out.println("Average Waiting time is : "+ (totalWT/n));
// displaying average waiting time.
System.out.println("Average Turn-around time is :"+(totalTT/n));
// displaying average turn around time. System.out.println("Gantt
Chart:");
//displaying gantt Chart
for(int i=0;i<n;i++){
System.out.print("P" +Pid[i]+"|");
}
}
void SRTF(){
Scanner sc=new Scanner(System.in);
System.out.println ("Enter the no of process:");
int n= sc.nextInt();
ArrayList<Integer> list=new ArrayList<Integer>();
int pid[] = new int[n]; // it takes pid of process
int AT[] = new int[n]; // at means arrival time
int BT[] = new int[n]; // bt means burst time
int FT[] = new int[n]; // ct means complete time
int TT[] = new int[n];// ta means turn around time
int WT[] = new int[n]; // wt means waiting time
int f[] = new int[n]; // checks process is completed or not
int k[]= new int[n]; // it also stores brust time
int i, st=0, total=0;
float totalWT=0, totalTT=0;
for (i=0;i<n;i++)
{
pid[i]= i+1;
```

```
System.out.print("Enter Arrival time for process "+(i+1)+" :
");//accepting arrival time for respective process

AT[i] = sc.nextInt();
System.out.print("Enter burst time for process "+(i+1)+": ");//accepting
Burst time for respective process
BT[i] = sc.nextInt();
k[i]= BT[i];
f[i]= 0;
}
while(true)
{
    int min=100,c=n;
    if (total==n)
     break;
    for( i=0;i<n;i++)
    {
      if((AT[i]<=st) && (f[i]==0) && (BT[i]<min))
      {
         min=BT[i];
         c=i;
      }
    }
   if (c==n)
   {
     st++;
   }
   else
    {
     list.add(c);
     BT[c]--;
     st++;
     if (BT[c]==0)
     {
      FT[c]= st;
      f[c]=1;
      total++;
     }
    }
}
for(i=0;i<n;i++)
{
TT[i] = FT[i] - AT[i];
WT[i] = TT[i] - k[i];
totalWT+= WT[i];
totalTT+= TT[i];
}
System.out.println("**********************************************************
*********************************");
System.out.println("Pid\tAT\tBT\tFT\tTT\tWT");
for(i=0;i<n;i++)
{
System.out.println(pid[i] +"\t"+ AT[i]+"\t"+ k[i] +"\t"+ FT[i] +"\t"+
TT[i]
+"\t"+ WT[i]);
}
System.out.println("**********************************************************
*********************************");
System.out.println("\nAverage Turn-aroundTime is: "+ (float)(totalTT/n));
System.out.println("Average WaitingTime is: "+ (float)(totalWT/n));
```

```
System.out.println("Gantt Chart:"); //displaying gantt Chart
for(i=0;i<list.size();i++)
{
System.out.print(" P"+(list.get(i)+1)+" |");
}
}
}
public class Main
{
    //main class
    public static void main(String[] args)
   {
    int choice;
    CPUscheduling c=new CPUscheduling(); //creating object of
CPUscheduling class
    do
    {

System.out.println("\n*****************************MENU****************
*************************************");
    System.out.println("1.FCFS ");
    System.out.println("2.SRTF");
    System.out.println("3.EXIT");

System.out.println("\n************************************************
*************************************");
    System.out.println("Enter your choice: ");
     Scanner s=new Scanner(System.in);
     choice=s.nextInt();
     switch(choice)
     {
        case 1:c.FCFS(); //calling the fcfs method
               break;
        case 2:c.SRTF();
               break;
        case 3:break;
        default:System.out.println("Invalid choice !!");
     }
   }while(choice!=3);
}
}
/*


*****************************MENU*****************************************
****************

1.FCFS

2.SRTF

3.EXIT



************************************************************************
****************

Enter your choice:
```

```
1

Enter the no of processes:

1   5

Enter Arrival time for process 1 : 0

Enter burst time for process 1 : 22

Enter Arrival time for process 2 : 4

Enter burst time for process 2 : 8

Enter Arrival time for process 3 : 10

Enter burst time for process 3 : 20

Enter Arrival time for process 4 : 14

Enter burst time for process 4 : 10

Enter Arrival time for process 5 : 28

Enter burst time for process 5 : 4

**********


Pid   AT    BT    FT    TT    WT

**********

1      0    22    22    22    0

2      4    8     30    26    18

3      10   20    50    40    20

4      14   10    60    46    36

5      28   4     64    36    32

**********

Average Waiting time is : 21.2

Average Turn-around time is :34.0

P1|P2|P3|P4|P5|

**********MENU**********

1.FCFS
```

```
2.SRTF

3.EXIT


********************************************************************************
*****************

Enter your choice:

2

Enter the no of process:

4

Enter Arrival time for process 1 : 0

Enter burst time for process 1: 6

Enter Arrival time for process 2 : 1

Enter burst time for process 2: 3

Enter Arrival time for process 3 : 2

Enter burst time for process 3: 5

Enter Arrival time for process 4 : 3

Enter burst time for process 4: 2

********************************************************************************
*************

Pid   AT    BT    FT    TT    WT

1     0     6     11    11    5

2     1     3     4     3     0

3     2     5     16    14    9

4     3     2     6     3     1

********************************************************************************
**************


Average Turn-aroundTime is: 7.75

Average WaitingTime is: 3.75

Gantt Chart:

 P1 | P2 | P2 | P2 | P4 | P4 | P1 | P1 | P1 | P1 | P1 | P3 | P3 | P3 | P3
| P3 |
```

```
******************************MENU*************************************
*****************

1.FCFS

2.SRTF

3.EXIT



*********************************************************************
*****************

Enter your choice:

3
*/
```