```
/*********************************************************************
 Name - Sejal Girish Mahajan
 Div - B-05
 Problem Statement :-
   Write a program to simulate CPU scheduling algorithms
   FCFS, SJF, Piority, and Round Robin
 *********************************************************************/

import java.util.Scanner;
public class FCFS
{
   static class Process
   {
      int id = 0;
      int arrivalTime = 0;
      int burstTime = 0;
      int completionTime = 0;
      int turnaroundTime = 0;
      int waitingTime = 0;
      int priority = 0;

      Process(int id, int arrivalTime, int burstTime,int priority)
      {
         this.id = id;
         this.arrivalTime = arrivalTime;
         this.burstTime = burstTime;
         this.priority = priority;
      }

      Process(int id,int burstTime,int priority)
      {
```

```
        this.id = id;
        this.arrivalTime = arrivalTime;
        this.burstTime = burstTime;
        this.priority = priority;
    }
}

private static void calculateFCFS(Process[] processes)
{
    int currentTime = 0;

    for (Process p : processes)
    {
        // Ensure the current time is at least the arrival time of the process
        if (currentTime < p.arrivalTime)
            currentTime = p.arrivalTime;



        p.completionTime = currentTime + p.burstTime;


        p.turnaroundTime = p.completionTime - p.arrivalTime;


        p.waitingTime = p.turnaroundTime - p.burstTime;


        currentTime = p.completionTime;
    }
}

private static void calculatePriority(Process[] processes)
{
    int n = processes.length;
```

```
int currentTime = 0;
int completed = 0;

boolean[] isCompleted = new boolean[n];

while (completed < n)
{
   int idx = -1;
   int highestPriority = Integer.MAX_VALUE;
   int earliestArrival = Integer.MAX_VALUE;

   for (int i = 0; i < n; i++)
   {
      if (processes[i].arrivalTime <= currentTime && !isCompleted[i])
      {
         if (processes[i].priority < highestPriority)
         {
            highestPriority = processes[i].priority;
            earliestArrival = processes[i].arrivalTime;
            idx = i;
         }
         else if (processes[i].priority == highestPriority)
         {
            if (processes[i].arrivalTime < earliestArrival)
            {
               earliestArrival = processes[i].arrivalTime;
               idx = i;
            }
            else if (processes[i].arrivalTime == earliestArrival)
               if (processes[i].id < processes[idx].id)
                  idx = i;
```

```
            }
        }
    }
    if (idx != -1)
    {
        Process p = processes[idx];

        if (currentTime < p.arrivalTime)
        {
            currentTime = p.arrivalTime;
        }

        p.completionTime = currentTime + p.burstTime;

        p.turnaroundTime = p.completionTime - p.arrivalTime;

        p.waitingTime = p.turnaroundTime - p.burstTime;

        currentTime = p.completionTime;

        isCompleted[idx] = true;
        completed++;
    }
    else
        currentTime++;

  }
}
private static void calculateSJF(Process[] processes)
{
  int n = processes.length, completed = 0;
```

```
int currentTime = 0; // Start at time 0
boolean[] comp = new boolean[n];

while (completed < n)
{
   int min = 9999;
   int md = -1;

   for (int i = 0; i < n; i++)
   {
      if (!comp[i] && processes[i].arrivalTime <= currentTime && processes[i].burstTime < min)
      {
         min = processes[i].burstTime;
         md = i;
      }
   }

   if (md == -1) {
      currentTime++;
      continue;
   }

   processes[md].completionTime = currentTime + processes[md].burstTime;
   processes[md].turnaroundTime = processes[md].completionTime - processes[md].arrivalTime;
   processes[md].waitingTime = processes[md].turnaroundTime - processes[md].burstTime;

   currentTime = processes[md].completionTime;

   comp[md] = true;
   completed++;
}
```

```
}
private static void calculateRR(Process[] processes)
{
   int n = processes.length;
   Scanner sc = new Scanner(System.in);
   System.out.print("\nEnter Time Quantum: ");
   int tq = sc.nextInt();

   int[] rbt = new int[n];
   for (int i = 0; i < n; i++)
   {
      rbt[i] = processes[i].burstTime;  // Copy burst time
   }

   int currentTime = 0;
   int completed = 0;
   boolean[] comp = new boolean[n];

   while (completed < n)
   {
      boolean didProcessRun = false;

      for (int i = 0; i < n; i++)
      {
         if (!comp[i])
         {
            didProcessRun = true;
            if (rbt[i] > tq)
            {
               currentTime += tq;
               rbt[i] -= tq;
```

```
                }
                else
                {
                    currentTime += rbt[i];
                    rbt[i] = 0;

                    comp[i] = true;
                    completed++;

                    processes[i].completionTime = currentTime;
                    processes[i].turnaroundTime = currentTime - processes[i].arrivalTime;
                    processes[i].waitingTime = processes[i].turnaroundTime - processes[i].burstTime;
                }
            }
        }

        if (!didProcessRun)
        {
            currentTime++;
        }
    }
}


    private static void display(Process[] processes)
    {
        System.out.println("\nProcess ID | Arrival Time | Priority | Burst Time | Completion Time | Turnaround Time | Waiting Time");
        for (Process p : processes)
            System.out.printf("%9d | %12d | %9d | %9d | %15d | %14d | %11d\n",
                p.id, p.arrivalTime,p.priority, p.burstTime, p.completionTime, p.turnaroundTime, p.waitingTime);
```

```
}
public static void main(String[] args)
{
   Scanner scanner = new Scanner(System.in);

   System.out.print("Enter the number of processes: ");
   int n = scanner.nextInt();

   Process[] processes = new Process[n];
   int ch = 0; // for priority scheduling

   do
   {
      System.out.println("\n-------MENU------");
      System.out.println("1.FCFS");
      System.out.println("2.SJF");
      System.out.println("3.Priority");
      System.out.println("4.RR");
      System.out.println("5.Exit");
      System.out.print("Enter your choice : ");
      ch = scanner.nextInt();
      switch(ch)
      {
         case 1:
            for(int i = 0; i < n; i++)
            {
               System.out.print("Enter arrival time for process " + (i + 1) + ": ");
               int arrivalTime = scanner.nextInt();
               System.out.print("Enter burst time for process " + (i + 1) + ": ");
               int burstTime = scanner.nextInt();
```

```
            processes[i] = new Process(i + 1, arrivalTime, burstTime,0);
        }
        for(int i = 0; i < n - 1; i++)
        {
            for(int j = i + 1; j < n; j++)
            {
                if(processes[i].arrivalTime > processes[j].arrivalTime)
                {
                    Process temp = processes[i];
                    processes[i] = processes[j];
                    processes[j] = temp;
                }
            }

        }
        calculateFCFS(processes);
        display(processes);
        break;

    case 2:
        for(int i = 0; i < n; i++)
        {
            System.out.print("Enter arrival time for process " + (i + 1) + ": ");
            int arrivalTime = scanner.nextInt();
            System.out.print("Enter burst time for process " + (i + 1) + ": ");
            int burstTime = scanner.nextInt();
            processes[i] = new Process(i + 1, arrivalTime, burstTime,0);
        }
        for(int i = 0; i < n - 1; i++)
        {
            for(int j = i + 1; j < n; j++)
```

```
        {
            if(processes[i].arrivalTime > processes[j].arrivalTime)
            {
                Process temp = processes[i];
                processes[i] = processes[j];
                processes[j] = temp;
            }
        }

    }
    calculateSJF(processes);
    display(processes);
    break;
case 3:
    for(int i = 0; i < n; i++)
    {
        System.out.print("Enter arrival time for process " + (i + 1) + ": ");
        int arrivalTime = scanner.nextInt();
        System.out.print("Enter burst time for process " + (i + 1) + ": ");
        int burstTime = scanner.nextInt();
        System.out.print("Enter priority for process " + (i + 1) + ": ");
        int priority = scanner.nextInt();
        processes[i] = new Process(i + 1, arrivalTime, burstTime,priority);
    }
    for(int i = 0; i < n - 1; i++)
    {
        for(int j = i + 1; j < n; j++)
        {
            if(processes[i].arrivalTime > processes[j].arrivalTime)
            {
                Process temp = processes[i];
```

```
                processes[i] = processes[j];
                processes[j] = temp;
            }
        }

    }
    calculatePriority(processes);
    display(processes);
    break;

case 4:
    for(int i = 0; i < n; i++)
    {
        System.out.print("Enter burst time for process " + (i + 1) + ": ");
        int burstTime = scanner.nextInt();
        processes[i] = new Process(i + 1, burstTime,0);
    }
    for(int i = 0; i < n - 1; i++)
    {
        for(int j = i + 1; j < n; j++)
        {
            if(processes[i].arrivalTime > processes[j].arrivalTime)
            {
                Process temp = processes[i];
                processes[i] = processes[j];
                processes[j] = temp;
            }
        }

    }
    calculateRR(processes);
```

```
                display(processes);
                break;
            }

        }while(ch != 5);

        scanner.close();
    }
}

/* OUTPUT :-

C:\Users\athar\OneDrive\Desktop\Java\SPOS>java FCFS
Enter the number of processes: 6

-------MENU------
1.FCFS
2.SJF
3.Priority
4.RR
5.Exit
Enter your choice : 1
Enter arrival time for process 1: 0
Enter burst time for process 1: 9
Enter arrival time for process 2: 1
Enter burst time for process 2: 3
Enter arrival time for process 3: 1
Enter burst time for process 3: 2
Enter arrival time for process 4: 1
Enter burst time for process 4: 4
Enter arrival time for process 5: 2
```

Enter burst time for process 5: 3

Enter arrival time for process 6: 3

Enter burst time for process 6: 2

Process ID | Arrival Time | Priority | Burst Time | Completion Time | Turnaround Time | Waiting Time

1	0	0	9	9	9	0
2	1	0	3	12	11	8
3	1	0	2	14	13	11
4	1	0	4	18	17	13
5	2	0	3	21	19	16
6	3	0	2	23	20	18

-------MENU------

1.FCFS

2.SJF

3.Priority

4.RR

5.Exit

Enter your choice : 2

Enter arrival time for process 1: 1

Enter burst time for process 1: 7

Enter arrival time for process 2: 3

Enter burst time for process 2: 3

Enter arrival time for process 3: 6

Enter burst time for process 3: 2

Enter arrival time for process 4: 7

Enter burst time for process 4: 10

Enter arrival time for process 5: 9

Enter burst time for process 5: 8

Enter arrival time for process 6: 14

Enter burst time for process 6: 3

Process ID | Arrival Time | Priority | Burst Time | Completion Time | Turnaround Time | Waiting Time

1	1	0	7	8	7	0
2	3	0	3	13	10	7
3	6	0	2	10	4	2
4	7	0	10	34	27	17
5	9	0	8	21	12	4
6	14	0	3	24	10	7

-------MENU------

1.FCFS

2.SJF

3.Priority

4.RR

5.Exit

Enter your choice : 3

Enter arrival time for process 1: 0

Enter burst time for process 1: 1

Enter priority for process 1: 6

Enter arrival time for process 2: 1

Enter burst time for process 2: 7

Enter priority for process 2: 4

Enter arrival time for process 3: 2

Enter burst time for process 3: 3

Enter priority for process 3: 3

Enter arrival time for process 4: 3

Enter burst time for process 4: 6

Enter priority for process 4: 5

Enter arrival time for process 5: 4

Enter burst time for process 5: 5

Enter priority for process 5: 1

Enter arrival time for process 6: 5

Enter burst time for process 6: 15

Enter priority for process 6: 2

Process ID | Arrival Time | Priority | Burst Time | Completion Time | Turnaround Time | Waiting Time

1	0	6	1	1	1	0
2	1	4	7	8	7	0
3	2	3	3	31	29	26
4	3	5	6	37	34	28
5	4	1	5	13	9	4
6	5	2	15	28	23	8

-------MENU------

1.FCFS

2.SJF

3.Priority

4.RR

5.Exit

Enter your choice : 4

Enter burst time for process 1: 10

Enter burst time for process 2: 5

Enter burst time for process 3: 8

Enter burst time for process 4: 3

Enter burst time for process 5: 6

Enter Time Quantum: 3

Process ID | Arrival Time | Priority | Burst Time | Completion Time | Turnaround Time | Waiting Time

1	0	0	10	32	32	22
2	0	0	5	20	20	15
3	0	0	8	31	31	23
4	0	0	3	12	12	9
5	0	0	6	26	26	20

-------MENU------

1.FCFS

2.SJF

3.Priority

4.RR

5.Exit

Enter your choice : 5

*/