Name: Shardul Vikram Dharmadhikari
Roll no: 8016

ASSIGNMENT NO: 08

AIM: Write a Java program (using OOP features) to implement following scheduling algorithms: FCFS, SJF (Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive)

OBJECTIVES:

- To study the basics of scheduling algorithms and learn concept of Preemptive and Non-Preemptive scheduling.
- To understand aggregation functions.

PRE-REQUISITES:

1. Java basics.
2. Basics of Operating System.

Schedular:-

Schedular is an Operaing System module that selects the next job to be admitted into the system & next process to run.There are major three types of schedular basically as follows:-

1. Short Term Schedular
2. Mid Term Schedular
3. Long Term Schedular

Scheduling :-

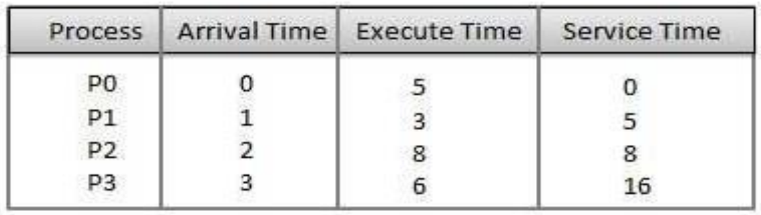
Schduling is the method specified by some means is assigned to resources that complete the work; work may be either virtual computation elements like thread , processes , data flows etc. There is Major two types of scheduling algorithm to solve any sort of operations.

THEORY:

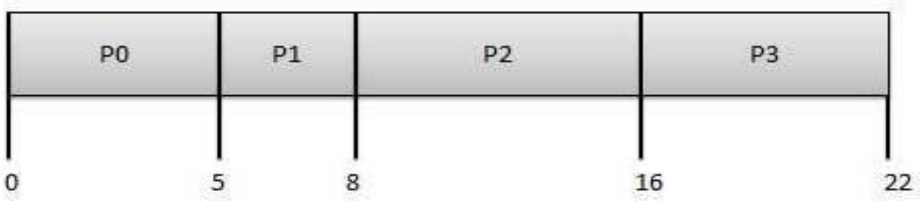
Preemptive Scheduling	Non-Preemptive Scheduling
Processor can be preempted to execute a different process in the middle of execution of any current process. CPU utilization is more compared to Non-Preemptive Scheduling. Waiting time and Response time is less. The preemptive scheduling is prioritized. The highest priority process should always be the process that is currently utilized. If a high priority process frequently arrives in the ready queue, low priority process may starve. Preemptive scheduling is flexible. Ex:- SRT Round Robin, etc. Ex:- FCFS, SJF, Priority	Once Processor starts to execute a process it must finish it before executing the other. It cannot be paused in middle. CPU utilization is less compared to Preemptive Scheduling. Waiting time and Response time is more. When a process enters the state of running, the state of that process is not deleted from the scheduler until it finishes its service time. If a process with long burst time is running CPU, then another process with less CPU burst time may starve. Non-preemptive scheduling is rigid.

A.First Come First Serve

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.



Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16



Wait time of each process is as follows –

Process	Wait Time : Service Time - Arrival Time
P0	0 - 0 =0
P1	5 - 1 =4
P2	8 - 2 =6
P3	16 - 3 =13

Average Wait Time: (0+4+6+13) / 4 = 5.75

B. Shortest Job First (Preemptive)

For SJF scheduling algorithm, read the number of processes/jobs in the system, Their CPU burst times arrange all the jobs in order with respect to their burst times. There may be two jobs in queue with the same execution time, and then FCFS approach is to be performed. Each process will be executed according to the length of its burst time. Then calculate the waiting time and turnaround time of each of the processes accordingly

PROCESS	DURATION	ORDER	ARRIVAL TIME
P1	9	1	0
P2	2	2	2

P0 3 - 0 = 3
P1 0 - 0 = 0
P2 16 - 2 = 14
P3 8 - 3 = 5

Average Wait Time: (3+0+14+5) / 4 = 5.50

C. Priority (Non-Preemptive)

□ Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems. Each process is assigned a priority. Process with the highest priority is to be executed first and so on.

Processes with the same priority are executed on first come first served basis. Priority can be decided based on memory requirements, time requirements or any other resource requirement.

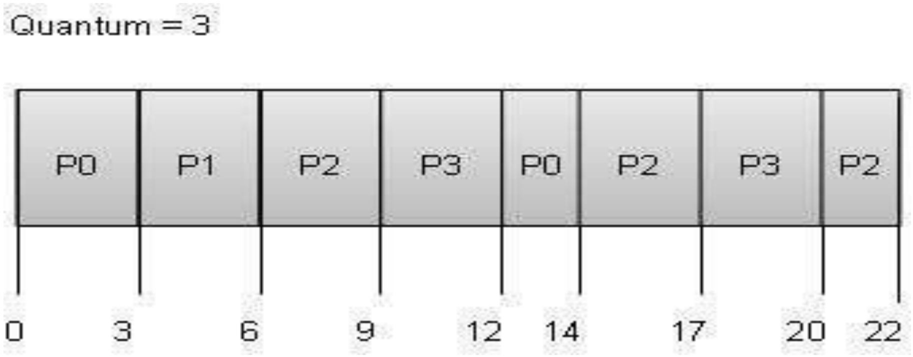
Process	Burst Time	Priority
P1	10	2
P2	5	0
P3	8	1

P1	P2	P3

0 10 18 23

D. Round Robin (Preemptive)

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a quantum.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.



Wait time of each process is as follows −

Process	Wait Time : Service Time - Arrival Time
P0	(0 - 0) + (12 - 3) =9
P1	(3 - 1) =2
P2	(6 - 2) + (14 - 9) + (20 - 17) =12
P3	(9 - 3) + (17 - 12) =11

Average Wait Time: (9+2+12+11) / 4 = 8.5

CONCLUSION:

Thus we learned different Scheduling algorithms.

CODE:

```
/* Problem Statement: Write a JAVA program (using oop features) to
implement following
1. FCFS
2. SJF(Preemptive)
3. Priority(Non-  Preemptive)
4. Round Robin(Preemptive)

            1.FCFS
*/
import java.io.*;
import java.util.Scanner;
public class FCFS
{
      public static void main(String args[])
      {
            int i,no_p,burst_time[],TT[],WT[];
            float avg_wait=0,avg_TT=0;
            burst_time=new int[50];
            TT=new int[50];
            WT=new int[50];
            WT[0]=0;
```

```
           Scanner s=new Scanner(System.in);
           System.out.println("Enter the number of process: ");
           no_p=s.nextInt();
           System.out.println("\nEnter Burst Time for processes:");
           for(i=0;i<no_p;i++)
           {
                System.out.print("\tP"+(i+1)+":  ");
                burst_time[i]=s.nextInt();
           }

           for(i=1;i<no_p;i++)
           {
                WT[i]=WT[i-1]+burst_time[i-1];
                avg_wait+=WT[i];
           }
           avg_wait/=no_p;

           for(i=0;i<no_p;i++)
           {
                TT[i]=WT[i]+burst_time[i];
                avg_TT+=TT[i];
           }
           avg_TT/=no_p;

     System.out.println("\n********************************************
******************");
           System.out.println("\tProcesses:");

     System.out.println("**********************************************
*****************");
           System.out.println("    Process\tBurst Time\tWaiting
Time\tTurn Around Time");
           for(i=0;i<no_p;i++)
           {
                System.out.println("\tP"+(i+1)+"\t
"+burst_time[i]+"\t\t  "+WT[i]+"\t\t "+TT[i]);

           }
           System.out.println("\n-----------------------------------------
-----------------------");
           System.out.println("\nAverage waiting time : "+avg_wait);
           System.out.println("\nAverage Turn Around time :
"+avg_TT+"\n");
      }
}

/*Output:
Enter the number of process:
3

Enter Burst Time for processes:
     P1:  24
     P2:  3
     P3:  3

*****************************************************************
     Processes:
*****************************************************************
    Process     Burst Time Waiting Time     Turn Around Time
     P1     24         0         24
```

```
      P2      3          24         27
      P3      3          27         30

----------------------------------------------------------------
Average waiting time : 17.0
Average Turn Around time : 27.0  */


/*Round Robin(Preemptive)*/
import java.util.*;
import java.io.*;
class RoundR
{
      public static void main(String args[])
      {
            int Process[]=new int[10];
            int a[]=new int[10];
            int Arrival_time[]=new int[10];
            int Burst_time[]=new int[10];
            int WT[]=new int[10];
            int TAT[]=new int[10];
            int Pno,sum=0;;
            int TimeQuantum;

System.out.println("\nEnter the no. of Process::");
            Scanner sc=new Scanner(System.in);
            Pno=sc.nextInt();
            System.out.println("\nEnter each process::");
            for(int i=0;i<Pno;i++)
            {
                  Process[i]=sc.nextInt();
            }

System.out.println("\nEnter the Burst Time of each process::");
            for(int i=0;i<Pno;i++)
            {
                  Burst_time[i]=sc.nextInt();
            }
System.out.println("\nEnter the Time Quantum::");
TimeQuantum=sc.nextInt();
            do{
            for(int i=0;i<Pno;i++)
            {
                  if(Burst_time[i]>TimeQuantum)
                  {
                        Burst_time[i]-=TimeQuantum;
                              for(int j=0;j<Pno;j++)
                              {
                                    if((j!=i)&&(Burst_time[j]!=0))
                              WT[j]+=TimeQuantum;
                        }
                  }
                        else
                        {
                              for(int j=0;j<Pno;j++)
                              {
                                    if((j!=i)&&(Burst_time[j]!=0))
                                    WT[j]+=Burst_time[i];
                              }
                              Burst_time[i]=0;
```

```
                      }
             }
                     sum=0;
                     for(int k=0;k<Pno;k++)
                      sum=sum+Burst_time[k];
          } while(sum!=0);

                   for(int i=0;i<Pno;i++)
                       TAT[i]=WT[i]+a[i];
                     System.out.println("process\t\tBT\tWT\tTAT");
                     for(int i=0;i<Pno;i++)
                     {

System.out.println("process"+(i+1)+"\t"+a[i]+"\t"+WT[i]+"\t"+TAT[i]);
                     }
                           float avg_wt=0;
                     float avg_tat=0;
                     for(int j=0;j<Pno;j++)
                     {
                             avg_wt+=WT[j];
                     }
                     for(int j=0;j<Pno;j++)
                     {
                             avg_tat+=TAT[j];
                     }
                       System.out.println("average waiting time
"+(avg_wt/Pno)+"\n Average turn around time"+(avg_tat/Pno));
      }
}

/*OUTPUT::
unix@unix-HP-280-G1-
MT:~/TEA33$ java RoundR
Enter the no. of Process::
5
Enter each process::
1
2
3
4
5

Enter the Burst Time of each process::
2
1
8
4
5
Enter the Time Quantum::
2
process         BT    WT    TAT
process1   0    0     0
process2   0    2     2
process3   0    12    12
process4   0    9     9
process5   0    13    13
average waiting time 7.2
Average turn around time7.2       */
```

```
/*              2. SJF(Non-Preemptive)            */
import java.util.Scanner;
class SJF1{
public static void main(String args[]){
int burst_time[],process[],waiting_time[],tat[],i,j,n,total=0,pos,temp;
float wait_avg,TAT_avg;
Scanner s = new Scanner(System.in);

System.out.print("Enter number of process: ");
n = s.nextInt();

process = new int[n];
burst_time = new int[n];
waiting_time = new int[n];
tat = new int[n];

System.out.println("\nEnter Burst time:");
for(i=0;i<n;i++)
{
System.out.print("\nProcess["+(i+1)+"]: ");
burst_time[i] = s.nextInt();;
process[i]=i+1; //Process Number
}

//Sorting
for(i=0;i<n;i++)
{
pos=i;
for(j=i+1;j<n;j++)
{
if(burst_time[j]<burst_time[pos])
pos=j;
}

temp=burst_time[i];
burst_time[i]=burst_time[pos];
burst_time[pos]=temp;

temp=process[i];
process[i]=process[pos];
process[pos]=temp;
}
//First process has 0 waiting time
waiting_time[0]=0;
//calculate waiting time
for(i=1;i<n;i++)
{
waiting_time[i]=0;
for(j=0;j<i;j++)
waiting_time[i]+=burst_time[j];
total+=waiting_time[i];
}

//Calculating Average waiting time
wait_avg=(float)total/n;
total=0;

System.out.println("\nProcess\t Burst Time \tWaiting Time\tTurnaround
Time");
for(i=0;i<n;i++)
{
tat[i]=burst_time[i]+waiting_time[i]; //Calculating Turnaround Time
total+=tat[i];
System.out.println("\n p"+process[i]+"\t\t "+burst_time[i]+"\t\t
"+waiting_time[i]+"\t\t "+tat[i]);
}

//Calculation of Average Turnaround Time
TAT_avg=(float)total/n;
System.out.println("\n\nAverage Waiting Time: "+wait_avg);
System.out.println("\nAverage Turnaround Time: "+TAT_avg);

}
}
```

```
/* 2. SJF(Preemptive)*/
import java.util.Scanner;

class sjf_swap1{
public static void main(String args[])

{
int
burst_time[],process[],waiting_time[],tat[],arr_time[],completion_time[],
i,j,n,total=0,total_comp=0,pos,temp;
float wait_avg,TAT_avg;
Scanner s = new Scanner(System.in);
 System.out.print("Enter number of process: ");
n = s.nextInt();
 process = new int[n];
burst_time = new int[n];
waiting_time = new int[n];
arr_time=new int[n];
tat = new int[n];
```

```
completion_time=new int[n];

//burst time
System.out.println("\nEnter Burst time:");
for(i=0;i<n;i++)
{
System.out.print("\nProcess["+(i+1)+"]: ");
burst_time[i] = s.nextInt();;
process[i]=i+1; //Process Number
}

//arrival time
System.out.println("\nEnter arrival time:");
for(i=0;i<n;i++)
{
System.out.print("\nProcess["+(i+1)+"]: ");
arr_time[i] = s.nextInt();;
process[i]=i+1; //Process Number
}

//Sorting
for(i=0;i<n;i++)
{
pos=i;
for(j=i+1;j<n;j++)
{
if(burst_time[j]<burst_time[pos])
pos=j;
}

temp=burst_time[i];
burst_time[i]=burst_time[pos];
burst_time[pos]=temp;

temp=process[i];
process[i]=process[pos];
process[pos]=temp;

System.out.println("process"+process[i]);
}
//completion
time new
for(i=1;i<n;i++)
{
completion_time[i]=0;
for(j=0;j<i;j++)
completion_time[i]+=burst_time[j];
 total_comp+=completion_time[i];
}

//First process has 0 waiting
time
waiting_time[0]=0;
//calculate

waiting time
for(i=1;i<n;i++)
{
waiting_time[i]=0;
for(j=0;j<i;j++)
waiting_time[i]+=burst_time[j];
total+=waiting_time[i];
}


//Calculating Average waiting time
wait_avg=(float)total/n;
total=0;

System.out.println("\nPro_number\t Burst Time \tcompletion_time\tWaiting
Time\tTurnaround Time");
for(i=0;i<n;i++)
{
tat[i]=burst_time[i]+waiting_time[i];
 //Calculating Turnaround Time
total+=tat[i];
System.out.println("\n"+process[i]+"\t\t "+burst_time[i]+"\t\t
"+completion_time[i]+"\t\t"+waiting_time[i]+"\t\t "+tat[i]);
}

//Calculation of Average Turnaround Time
TAT_avg=(float)total/n;
System.out.println("\n\nAWT: "+wait_avg);
System.out.println("\nATAT: "+TAT_avg);

}
}
```