1 Title:

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

2 Objectives :

- To understand OS & SCHEDULLING Concepts
- To implement Scheduling FCFS, SJF, RR & Priority algorithms
- To study about Scheduling and scheduler

3 Problem Statement :

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

4 Outcomes:

After completion of this assignment students will be able to:

- Knowledge Scheduling policies
- Compare different scheduling algorithms

5 Software Requirements:

JDK/Eclipse

6 Hardware Requirement:

- M/C Lenovo Think center M700 Ci3,6100,6th Gen. H81, 4GB RAM ,500GB HDD

7 Theory Concepts:

What is scheduling?

Scheduling is the process of planning, organizing, and assigning specific times for tasks, events, or actions to be completed. It involves determining when and how resources will be used to achieve certain goals or complete projects efficiently.

What is scheduler?

1. Scheduler in an OS module that selects the next job to be admitted into the system and the next process to run.
2. Primary objective of the scheduler is to optimize system performance in accordance with the criteria deemed by the system designers. In short, scheduler is that module of OS which schedules the programs in an efficient manner.

Necessity of scheduling

• Scheduling is required when no. of jobs are to be performed by CPU.

• Scheduling provides mechanism to give order to each work to be done.

• Primary objective of scheduling is to optimize system performance.

• Scheduling provides the ease to CPU to execute the processes in efficient manner.

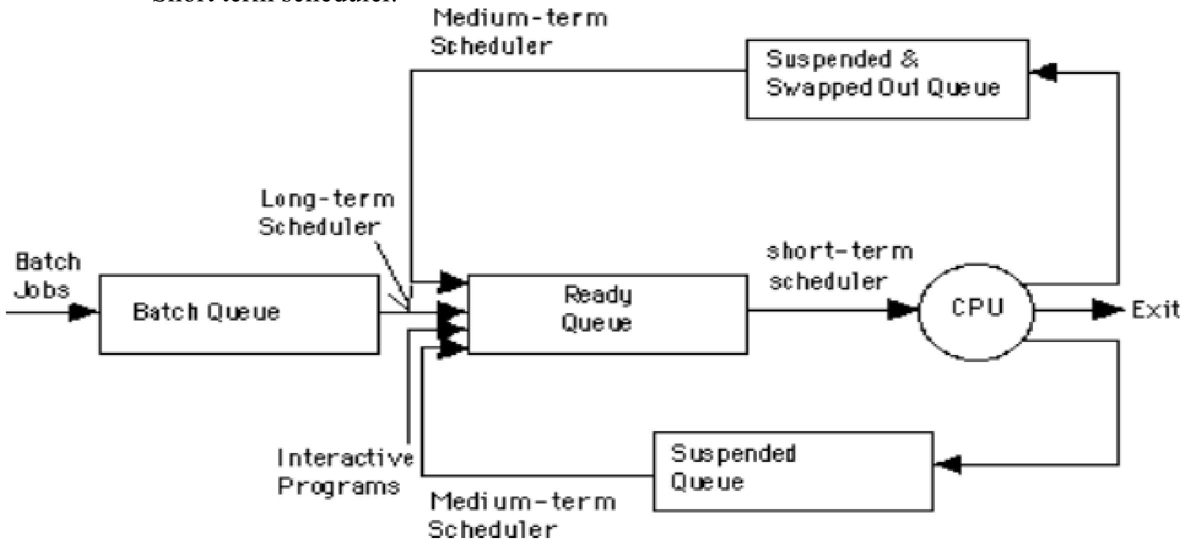
Types of schedulers

In general, there are three different types of schedulers which may co-exist in a complex operating system.

• Long term scheduler

• Medium term scheduler

• Short term scheduler.



``

Long Term Scheduler

• The long term scheduler, when present works with the batch queue and selects the next batch job to beexecuted.

• Batch is usually reserved for resource intensive (processor time, memory, special I/O devices) lowpriority programs that may be used fillers of low activity of interactive jobs.

• Batch jobs usually also contains programmer-assigned or system-assigned estimates of their resourceneeds such as memory size, expected execution time and device requirements.

• Primary goal of long term scheduler is to provide a balanced mix of jobs.

Medium Term Scheduler

• After executing for a while, a running process may because suspended by making an I/O request or byissuing a system call.

• When number of processes becomes suspended, the remaining supply of ready processes in systems where all suspended processes remains resident in memory may become reduced to a level that impairsfunctioning of schedulers.

• The medium term scheduler is in charge of handling the swapped out processes.

• It has little to do while a process is remained as suspended.

Short Term Scheduler

• The short term scheduler allocates the processor among the pool of ready processes resident in thememory.

• Its main objective is to maximize system performance in accordance with the chosen set of criteria.

• Some of the events introduced thus for that cause rescheduling by virtue of their ability to change theglobal system state are:

• Clock ticks

• Interrupt and I/O completions

• Most operational OS calls

• Sending and receiving of signals

• Activation of interactive programs.

• Whenever one of these events occurs ,the OS involves the short term scheduler.

Scheduling Criteria :

- **CPU Utilization:**
  Keep the CPU as busy as possible. It range from 0 to 100%. In practice, it range from 40 to 90%.
- **Throughput:**
  Throughput is the rate at which processes are completed per unit of time.
- **Turnaround time:**

  This is the how long a process takes to execute a process. It is calculated as the time gap between thesubmission of a process and its completion.
- **Waiting time:**

  Waiting time is the sum of the time periods spent in waiting in the ready queue.
- **Response time:**

  Response time is the time it takes to start responding from submission time. It is calculated as theamount of time it takes from when a request was submitted until the first response is produced.

Non-preemptive Scheduling :

In non-preemptive mode, once if a process enters into running state, it continues to execute until it terminates or blocks itself to wait for Input/Output or by requesting some operating system service.

Preemptive Scheduling :

In preemptive mode, currently running process may be interrupted and moved to the ready State by the operating system.

When a new process arrives or when an interrupt occurs, preemptive policies may incur greateroverhead than non-preemptive version but preemptive version may provide better service.

It is desirable to maximize CPU utilization and throughput, and to minimize turnaround time, waitingtime and response time.

Types of scheduling Algorithms
• In general, scheduling disciplines may be pre-emptive or non-pre-emptive .

• In batch, non-pre-emptive implies that once scheduled, a selected job turns to completion.There are different types of scheduling algorithms such as:

- FCFS(First Come First Serve)
- SJF(Short Job First)
- Priority scheduling
- Round Robin Scheduling algorithm

First Come First Serve Algorithm
• FCFS is working on the simplest scheduling discipline.

• The workload is simply processed in an order of their arrival, with no pre-emption.

• FCFS scheduling may result into poor performance.

• Since there is no discrimination on the basis of required services, short jobs may considerable in turnaround delay and waiting time.

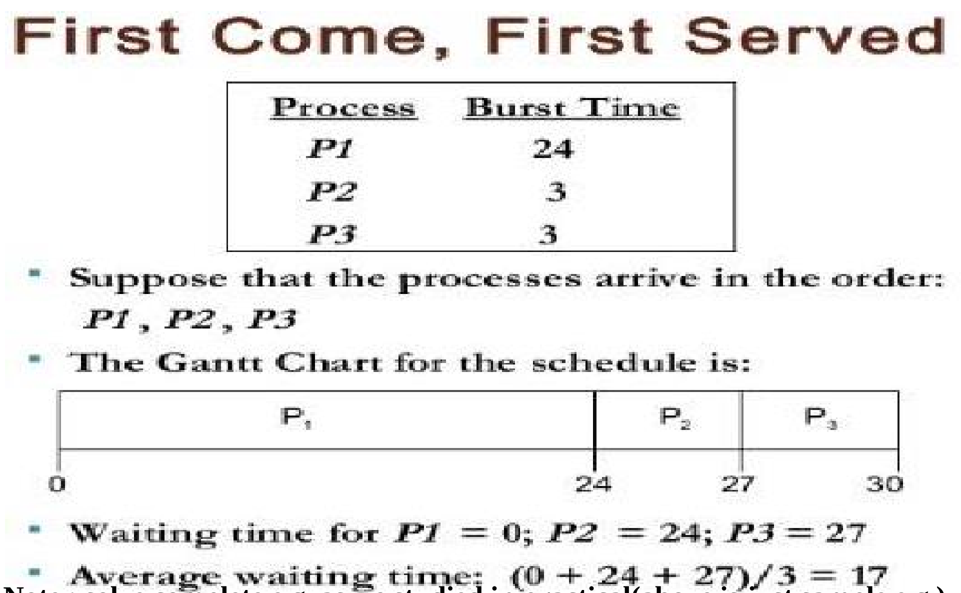
``

Advantages

- Better for long processes
- Simple method (i.e., minimum overhead on processor)
- No starvation

Disadvantages

- Convoy effect occurs. Even very small process should wait for its turn to come to utilize the CPU.Short process behind long process results in lower CPU utilization.
- Throughput is not emphasized.



First Come, First Served

Process	Burst Time
P1	24
P2	3
P3	3

- Suppose that the processes arrive in the order: *P1* , *P2* , *P3*
- The Gantt Chart for the schedule is:

P_1	P_2	P_3

0 24 27 30

- Waiting time for *P1* = 0; *P2* = 24; *P3* = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

Note : solve complete e.g. as we studied in practical(above is just sample e.g.). you can take anye.g.

Shortest Job First Algorithm :

- his is also known as **shortest job first**, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processer should know in advance how much time process will take.

Advantages

- It gives superior turnaround time performance to shortest process next because a short job is givenimmediate preference to a running longer job.
- Throughput is high.

Disadvantages

- Elapsed time (i.e., execution-completed-time) must be recorded, it results an additional overhead onthe processor.
- Starvation may be possible for the longer processes.

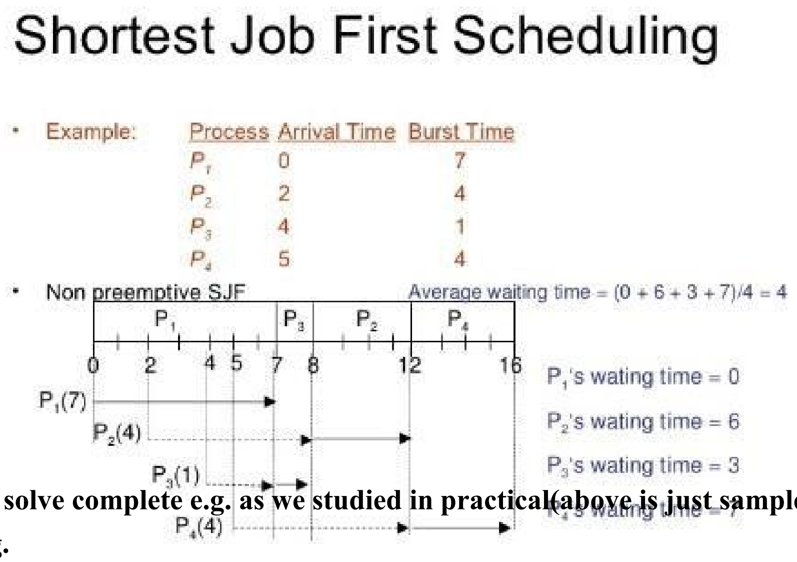
This algorithm is divided into two types:

• Pre-emptive SJF

• Non-pre-emptive SJF

• Pre-emptive SJF Algorithm:

In this type of SJF, the shortest job is executed 1st. the job having least arrival time is taken first for execution. It is executed till the next job arrival is reached.



Shortest Job First Scheduling

• Example:

Process	Arrival Time	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

• Non preemptive SJF

Average waiting time = (0 + 6 + 3 + 7)/4 = 4

P_1's wating time = 0

P_2's wating time = 6

P_3's wating time = 3

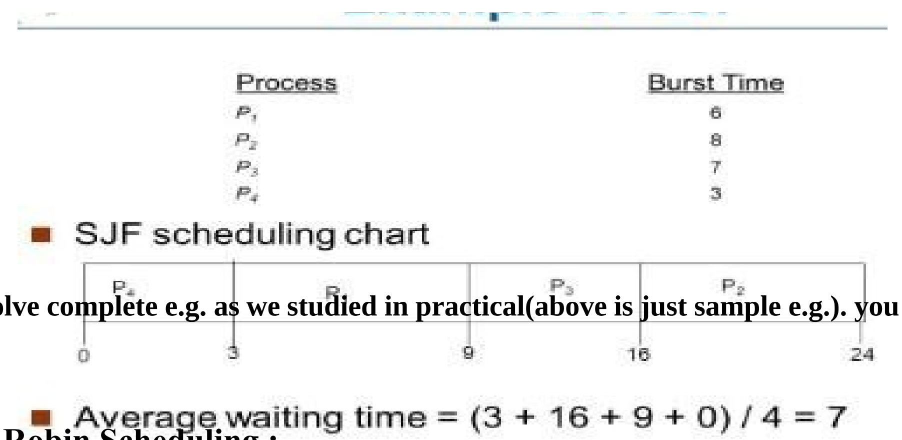
P_4's wating time = 7

Note : solve complete e.g. as we studied in practical(above is just sample e.g.). you can take anye.g.

Non-pre-emptive SJF Algorithm:

In this algorithm, job having less burst time is selected 1st for execution. It is executed for its total burst time and then the next job having least burst time is selected.

Example of SJF



Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

- SJF scheduling chart

- Average waiting time = (3 + 16 + 9 + 0) / 4 = 7

Note : solve complete e.g. as we studied in practical(above is just sample e.g.). you can take anye.g.

Round Robin Scheduling :

- 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 agiven time period.
- Context switching is used to save states of preempted processe

Advantages

- Round-robin is effective in a general-purpose, times-sharing system or transaction-processingsystem.
- Fair treatment for all the processes.
- Overhead on processor is low.
- Overhead on processor is low.
- Good response time for short processes.

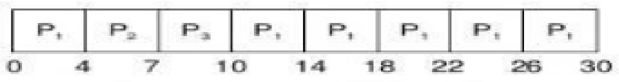
Disadvantages

- Care must be taken in choosing quantum value.
- Processing overhead is there in handling clock interrupt.
- Throughput is low if time quantum is too small.

Round Robin

Process	Burst Time
P1	24
P2	3
P3	3

- Quantum time = 4 milliseconds
- The Gantt chart is:



- Average waiting time = {[0+(10-4)]+4+7}/3 = 5.6

Note : solve complete e.g. as we studied in practical(above is just sample e.g.). you can take anye.g.

Priority Scheduling :

- Priority scheduling is a non-preemptive algorithm and one of the most common schedulingalgorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resourcerequirement.

Advantage

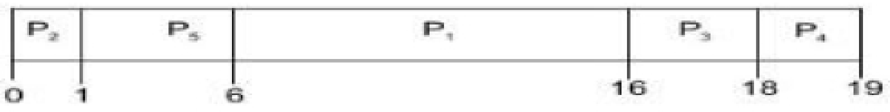
- Good response for the highest priority processes.

Disadvantage

Priority

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

- Gantt Chart



- Average waiting time = (6 + 0 + 16 + 18 + 1)/5 = 8.2

- Starvation may be possible for the lowest priority processes.

Note : solve complete e.g. as we studied in practical(above is just sample e.g.). you can take any e.g.

Algorithms(procedure)

:FCFS :

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst timeStep 4: Set the waiting of the first process as ‘0’ and its burst time as its turn around time Step 5: for each process in the Ready Q calculate

a Waiting time for process(n)= waiting time of process (n-1) + Burst time of process(n-1)

b Turn around time for Process(n)= waiting time of Process(n)+ Burst time for process(n)

Step 6: Calculate

a Average waiting time = Total waiting Time / Number of process

b Average Turnaround time = Total Turnaround Time / Number of processStep 7: Stop the process

SJF :

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst timeStep 4: Start the Ready Q according the shortest Burst time by sorting according to lowest tohighest burst time.

Step 5: Set the waiting time of the first process as ‘0’ and its turnaround time as its burst time.

Step 6: For each process in the ready queue, calculate

c Waiting time for process(n)= waiting time of process (n-1) + Burst time of process(n-1)

d Turn around time for Process(n)= waiting time of Process(n)+ Burst time for process(n)

Step 6: Calculate

c Average waiting time = Total waiting Time / Number of process

d Average Turnaround time = Total Turnaround Time / Number of processStep 7: Stop the process

RR :

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue and time quantum (or) time slice Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time Step 4: Calculate the no. of time slices for each process where

No. of time slice for process(n) = burst time process(n)/time slice

Step 5: If the burst time is less than the time slice then the no. of time slices =1.

Step 6: Consider the ready queue is a circular Q, calculate

a Waiting time for process(n) = waiting time of process(n-1)+ burst time of process(n-1) +the time difference in getting the CPU from process(n-1)

b Turn around time for process(n) = waiting time of process(n) + burst time of process(n)+the time difference in getting CPU from process(n).

Step 7: Calculate

e Average waiting time = Total waiting Time / Number of process

f Average Turnaround time = Total Turnaround Time / Number of process Step 8: Stop the process.

Priority

Scheduling :

Algorithms :

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time, priority Step 4: Start the Ready Q according the priority by sorting according to lowest to highest burst time and process.

Step 5: Set the waiting time of the first process as ‘0’ and its turnaround time as its burst time.

``

Step 6: For each process in the ready queue, calculate

e Waiting time for process(n)= waiting time of process (n-1) + Burst time of process(n-1)

f Turn around time for Process(n)= waiting time of Process(n)+ Burst time for process(n)

Step 6: Calculate

g Average waiting time = Total waiting Time / Number of process

h Average Turnaround time = Total Turnaround Time / Number of process Step 7: Stop the process

Note: you can write algorithm & procedure as per your program/concepts

9 Flowchart :

Note: you should draw flowchart as per algorithm/procedure as above

10 Conclusion:

Hence we have studied and implemented that-

- CPU scheduling concepts like context switching, types of schedulers, different timing parameter like waiting time, turnaround time, burst time, etc.
- Different CPU scheduling algorithms like FIFO, SJF, Priority and Round Robin Etc.
