Assignment #3 - CPU Schedulers Simulator

Scheduling is a fundamental operating-system function. Almost all computer resources are scheduled before use. The CPU is, of course, one of the primary computer resources. Thus, its scheduling is central to operating-system design. CPU scheduling determines which processes run when there are multiple run-able processes. CPU scheduling is important because it can have a big effect on resource utilization and the overall performance of the system.

Write a java program to simulate the following schedulers:

- **Non-preemptive** Priority Scheduling using context switching
- **Non-Preemptive** Shortest- Job First (SJF)

  (Solve starvation problem if exist in this schedule)
- Shortest- Remaining Time First (SRTF) Scheduling using context switching -

  (Solve starvation problem if exist in this schedule)
- FCAI Scheduling :
  a. Traditional CPU scheduling algorithms, like Round Robin (RR) or Priority Scheduling, often suffer from **starvation** or inefficiency when handling a mix of short- and long-burst processes with varying priorities. To address these limitations, we introduce **FCAI Scheduling**, an adaptive scheduling algorithm that combines **priority**, **arrival time**, and **remaining burst time** into a single **FCAI Factor** to dynamically manage the execution order and quantum allocation for processes.

Key Components

- **Dynamic FCAI Factor**:
  - A composite metric calculated for each process, considering:
    - Priority (P)
    - Arrival time (AT)
    - Remaining burst time (RBT)

  FCAI Factor = (10−Priority) + (Arrival Time/V1) + (Remaining Burst Time/V2)

  Where:

    - **V1** = last arrival time of all processes/10
    - **V2** = max burst time of all processes/10

- **Quantum Allocation Rules**:
  - Each process starts with a **unique quantum**.
  - When processes are preempted or added back to the queue, their quantum is updated dynamically:
    - Q= Q + 2 (if process completes its quantum and still has remaining work)
    - Q=Q + unused quantum (if process is preempted)

- **Non-Preemptive and Preemptive Execution**:
  - A process executes non-preemptively for the first 40% of its quantum.
  - After 40% execution, preemption is allowed.

Example of AG Schedule:

Processes	Burst time	Arrival time	Priority	Quantum
P1	17	0	4	4
P2	6	3	9	3
P3	10	4	3	5
P4	4	29	8	2

Answer:

- Initial Calculations
  - V1 = 2.9, V2 = 1.7

Processes	Burst time	Arrival time	Priority	Quantum	Initial FCAI Factors
P1	17	0	4	4	6+[0/2.9] +[17/1.7]=16
P2	6	3	9	3	1+[3/2.9]+[6/1.7]=7
P3	10	4	3	5	7+[4/2.9]+[10/1.7]=15
P4	4	29	10	2	0+[29/2.9]+[4/1.7]=13

Detailed Execution Timeline:

Time	Process	Executed Time	Remaining Burst Time	Updated Quantum	Priority	FCAI Factor	Action - Details
0–3	P1	3	14	4 → 5	4	16 → 15 6+[0/2.9] +[17/1.7] → 6+[0/2.9] +[14/1.7]	P1 starts execution, runs for 3 units, remaining burst = 14.
3–6	P2	3	3	3 → 5	9	7 → 5 1+[3/2.9] +[6/1.7] → 1+[3/2.9] +[3/1.7]	P2 preempts P1, runs for 3 units, remaining burst = 3.
6–8	P1	2	12	5 → 8	4	15 → 14 6+[0/2.9] +[14/1.7] → 6+[0/2.9] +[12/1.7]	P1 runs for 2 more units, remaining burst = 12

8–11	P2	3	0	Completed	9	Completed	P2 preempts P1, runs for 3 units, P2 completes execution.
11–13	P3	2	8	5 → 8	3	15 → 14 7+[4/2.9]+[10/1.7] → 7+[4/2.9]+[8/1.7]	P3 starts execution, runs for 2 units, remaining burst = 8.
13–21	P1	8	4	8 → 10	4	14 → 9 6+[0/2.9]+[12/1.7] → 6+[0/2.9]+[4/1.7]	P1 preempts P3, runs for 8 units, remaining burst = 4.
21–25	P3	4	4	8 → 12	3	14 → 12 7+[4/2.9]+[8/1.7] → 7+[4/2.9]+[4/1.7]	P3 runs for 4 more units, remaining burst = 4
25–29	P1	4	0	Completed	4	Completed	P1 preempts P3, runs for 4 units, P1 completes execution.
29–33	P3	4	0	Completed	3	Completed	P3 runs for 4 more units, P3 completes execution.
33–37	P4	4	0	Completed	10	Completed	P4 starts execution, runs for 2 units, then another 2 units

Note:

1. All calculations are performed using the **ceil** function
2. A queue is used for process ordering. If a process executes 40% and is preempted by another process with a better factor, the preempted process is re-added to the queue. If the process is not preempted, the next process in the queue will execute.

Program Input:

- Number of processes
- Round Robin Time Quantum
- context switching

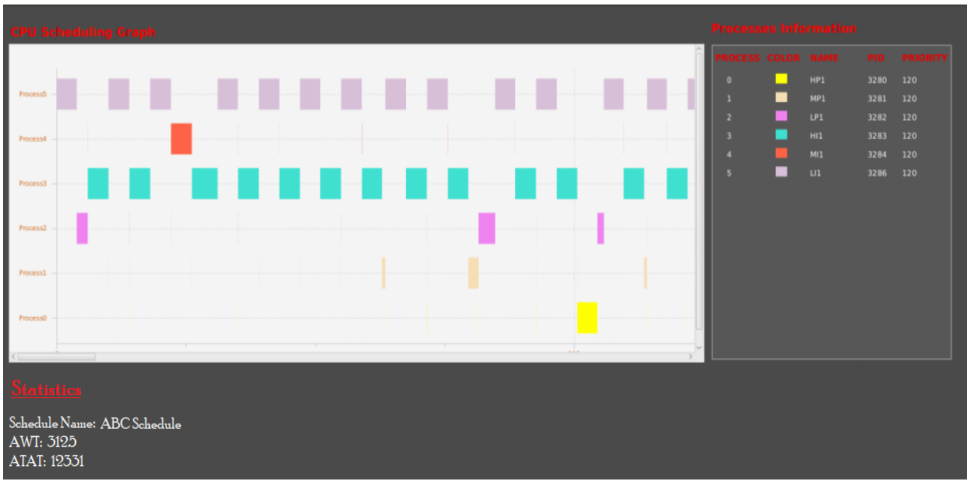
For Each Process you need to receive the following parameters from the user:

- Process Name
- Process Color(Graphical Representation)
- Process Arrival Time
- Process Burst Time
- Process Priority Number

Program Output:

For each scheduler output the following:

- Processes execution order
- Waiting Time for each process
- Turnaround Time for each process
- Average Waiting Time
- Average Turnaround Time
- Print all history update of quantum time for each process (**AG Scheduling**)
- **BOUNS:** graphical representation of Processes execution order
  (Example of Graphical representation)



Guidelines:

- **Language:** Java.
- **Submission Deadline:** 6^{st} December, 2024 2:59 PM.
- **Team Members:** The assignment is submitted in group of maximum 5 students.
- **Submission Format:** You must submit only one ".zip" file containing the source code, and the submitted file name must follow this format: ID1_ID2_ID3 _Group for example 20190000_20190001_20190002_DS1.
- **Late submission is not allowed.**

Grading Criteria
BONUS (2 grades)

	Shortest- Job First (SJF) Scheduling	SRTF Scheduling	Priority Scheduling	FCAI Scheduling	Grade
Grade	1 + (0.5 Bonus)	1.5 + (0.5 Bonus)	1 + (0.5 Bonus)	2.5 + (0.5 Bonus)	6 + 2