**Overview:**

**✅ PHASE 1 – Core Process and CPU Simulation**

🔑 Goal: Get basic scheduling, process execution, and CPU behavior simulated.

**Focus:**

* Basic **CPU execution logic**
* **Process scheduling algorithms** (FCFS, SJF, RR)
* Process queue management
* CPU burst simulation, CLI interaction

**✅ PHASE 2 – Memory and Storage Simulation**

🔑 Goal: Integrate memory allocation and paging with your scheduler.

**Focus:**

* Memory allocation strategies (First Fit, Best Fit)
* Simulate RAM size, fragmentation
* Paging logic: logical to physical address mapping
* Optional: simulate swap space or limited RAM causing page faults

**✅ PHASE 3 – Advanced Features & Robustness**

🔑 Goal: Add complexity like deadlock detection, disk I/O, and simulate full OS behavior.

**Focus:**

* **Deadlock handling** via Banker's algorithm
* **Disk scheduling**: FCFS, SSTF, SCAN
* Introduce **interrupt simulation** and **error handling**
* Integrate with Main.java to unify all modules

**PHASE 1:**

✅ FCFS  
✅ SJF (preemptive & non-preemptive)  
✅ Priority (preemptive & non-preemptive)  
✅ Round Robin

is a solid and realistic plan. Here's how your code and logic will flow — visualized as a **conceptual flowchart with data flow explanation**.

**📦 Data Flow Description**

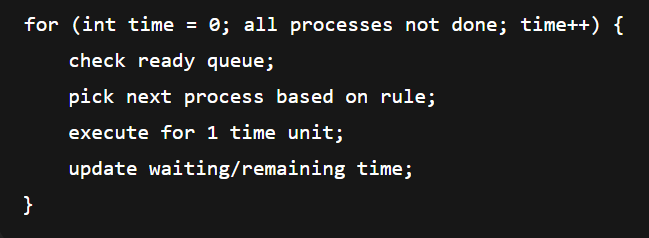
|  |  |
| --- | --- |
| **Step** | **What Happens** |
| **User Input** | Enter number of processes + their attributes |
| **Process Object Creation** | Each input is wrapped into a Process object |
| **Algorithm Selection** | User chooses which scheduling class to invoke |
| **Pass to Scheduler** | List<Process> passed to simulate() method in the class |
| **Scheduler Logic** | Time-driven or priority-driven execution |
| **Update Process States** | Each Process updated with final timings |
| **Print Results** | Results shown via System.out.println() |



**🔄 Internal Time-Driven Schedulers For:**

* SJF (Preemptive)
* Priority (Preemptive)
* Round Robin

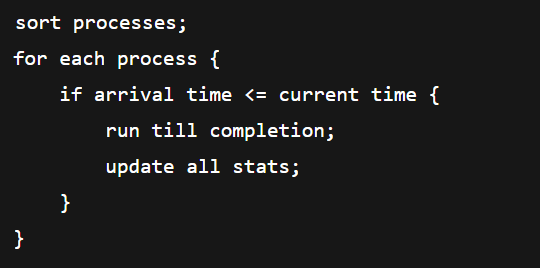
They simulate **time unit by unit** like this:



**⚖️ Non-Time-Driven Schedulers**

* FCFS
* SJF (Non-Preemptive)
* Priority (Non-Preemptive)

They simulate in one go per process:



**PHASE 2:**

✅ Allocation (Fixed + Dynamic)

✅ Paging & Page Tables

✅ Page Replacement (FIFO, LRU)

✅ Segmentation

✅ Page Faults + Virtual Memory (partial)

✅ Fragmentation Analysis

|  |  |  |
| --- | --- | --- |
| Feature | Can We Simulate? | Notes |
| Fixed Partition Allocation | ✅ | Use arrays for fixed slots |
| Dynamic Allocation (First Fit, Best Fit, Worst Fit) | ✅ | Variable-sized partitions |
| Paging | ✅ | Model page tables and frame allocation |
| Segmentation | ✅ | Simulate segment mapping like paging |
| Page Faults | ✅ | When accessing non-resident pages |
| Page Replacement | ✅ | FIFO, LRU via queues/lists |
| Virtual Memory (basic) | ✅ | Simulate partial loading and swapping |
| Fragmentation stats | ✅ | Count internal/external unused blocks |



**✅ Part 1: Partition Allocation**

* Fixed partition allocation simulation
* Dynamic partition allocation (First Fit, Best Fit, Worst Fit)
* Internal and external fragmentation stats

**✅ Part 2: Paging**

* Load processes into pages
* Simulate page tables and frames
* Visual CLI layout of memory
* Handle page faults

**✅ Part 3: Page Replacement**

* FIFO, LRU algorithms
* Track page hits, faults
* Time-simulated access stream

**✅ Part 4: Segmentation**

* Segment table per process
* Simulate segment loading and access
* Compare with paging