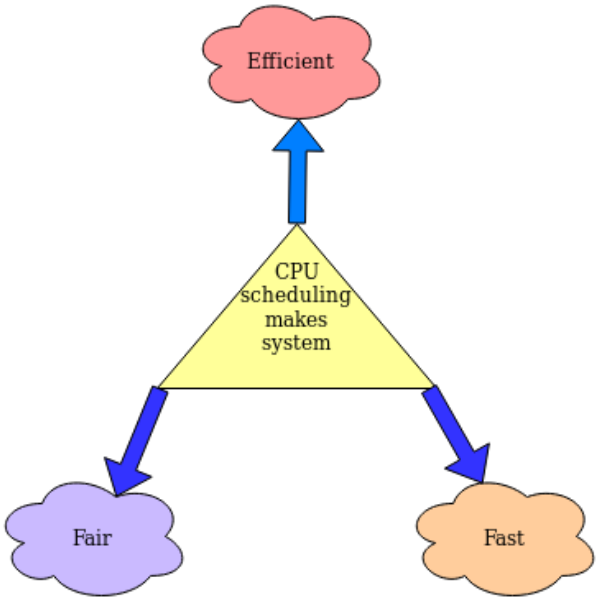
Operating Systems

CPU Scheduling algorithms implementation in Java

Ratnam Parikh(036) ***Devshree Patel(075)***

Introduction

CPU scheduling is a mechanism which allows one process to use the CPU resources while the execution of other processes is on hold or in waiting state due to unavailability of any CPU resource. Thus, making full use of the CPU. Whenever the CPU becomes idle, the operating system must select one of the processes in the **ready queue** to be executed. The selection process is carried out by the short-term scheduler (or CPU scheduler). The scheduler selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.



Types Of CPU Scheduling Algorithms

1. **Preemptive Scheduling:** In this type of Scheduling, the tasks are usually assigned with priorities. At times it is necessary to run a certain task that has a higher priority before another task although it is running. Therefore, the running task is interrupted for some time and resumed later when the priority task has finished its execution. Eg: Round Robin Scheduling Algorithm
2. **Non-Preemptive Scheduling:** Under non-preemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state. Eg: First Come First Serve Scheduling Algorithm.

Types Of Process Scheduler

1. **Long term or job scheduler:** It brings the new process to the 'Ready State'. It controls the Degree *of Multiprogramming*, i.e., number of process present in ready state at any point of time.It is important that the long-term scheduler make a careful selection of both IO and CPU bound process.
2. **Short term or CPU scheduler:** It is responsible for selecting one process from ready state for scheduling it on the running state. ***Dispatcher*** is responsible for loading the process selected by Short-term scheduler on the CPU (Ready to Running State) Context switching is done by dispatcher only. A dispatcher does the following:
    1. Switching context.
    2. Switching to user mode.
    3. Jumping to the proper location in the newly loaded program.
3. **Medium-term scheduler** It is responsible for suspending and resuming the process. It mainly does swapping (moving processes from main memory to disk and vice versa). Swapping may be necessary to improve the process mix or because a change in memory requirements has overcommitted available memory, requiring memory to be freed up.

Round Robin Algorithm

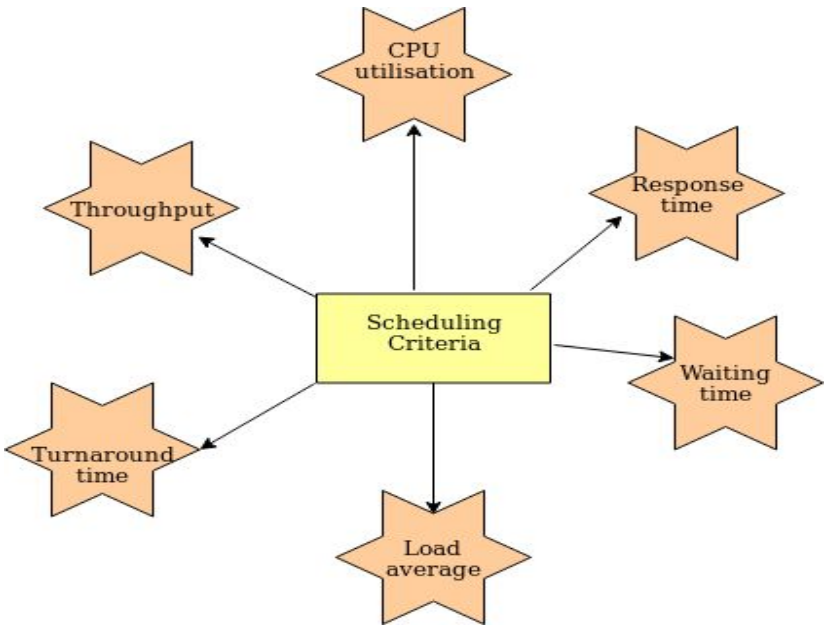
Round Robin is a CPU Scheduling Algorithm where each process is assigned a fixed time slot in a cyclic way.

1. It is simple, easy to implement, and starvation-free as all processes get fair share of CPU.
2. One of the most commonly used techniques in CPU scheduling as a core.
3. It is preemptive as processes are assigned CPU only for a fixed slice of time at most.
4. The disadvantage of it is more overhead of context switching.

First Come First Serve Algorithm

First in, first out (FIFO), also known as a first come, first served (FCFS), is the simplest scheduling algorithm. FIFO simply queues processes in the order that they arrive in the ready queue.In this, the process that comes first will be executed first and next process starts only after the previous gets fully executed.

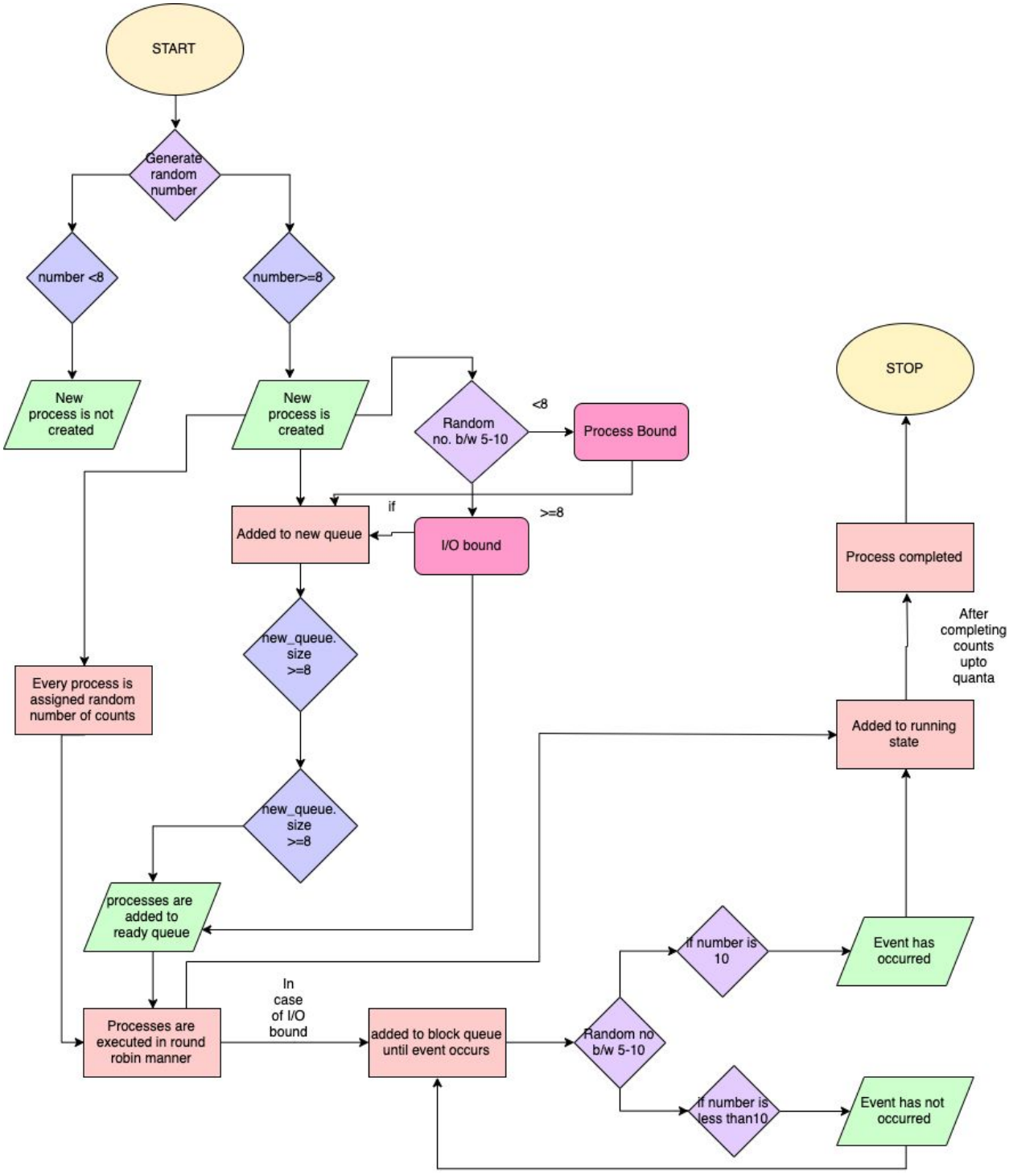
CPU Scheduling Criteria



Working Flow

1. First a thread of process creation runs and dynamically creates process according to the random number generated between 5 to 10. Each value generated between 8 to 10 will lead to process creation. This work is done independently by a thread which helps to portray a real-time situation of randomly creating processes.
2. Now, the new process is placed in a “new queue” which stores all the new generated processes.
3. The long term scheduler is now invoked as new processes join the new queue. Long term scheduler shifts a process from new queue and places them in ready queue for execution.
4. The execution is done in a Round-Robin fashion. Where every process is in execution for the time equal to time quantum specified by the user.
5. The process is assigned count at the beginning of creation which is decremented every time clock. This way when the count of the process is zero that means the process has completed its execution.
6. Processes that are label I/O bound are treated in a different manner. Here, we consider that when an I/O bound process is brought into execution, after it is served for the time quantum the process is blocked waiting for an I/O event to occur.
7. The medium term scheduler is than invoked when an event occurs and the process is then brought back to running state directly.
8. The occurrence of an event is also portrayed as a decision of random number generated. A random number between 5 to 10 is generated from which at the occurrence of number 10 a blocked process is shifted to running state.
9. At the end of execution of all processes an analysis of the processes executed is done and presented to the user.

Flow Chart


START
Generate random number
number <8
number>=8
New process is not created
New process is created
Random no. b/w 5-10
<8
Process Bound
if
>=8
I/O bound
Added to new queue
new_queue. size >=8
Every process is assigned random number of counts
new_queue. size >=8
processes are added to ready queue
Processes are executed in round robin manner
In case of I/O bound
added to block queue until event occurs
Random no b/w 5-10
if number is 10
Event has occurred
if number is less than10
Event has not occurred
Added to running state
After completing counts upto quanta
Process completed
STOP

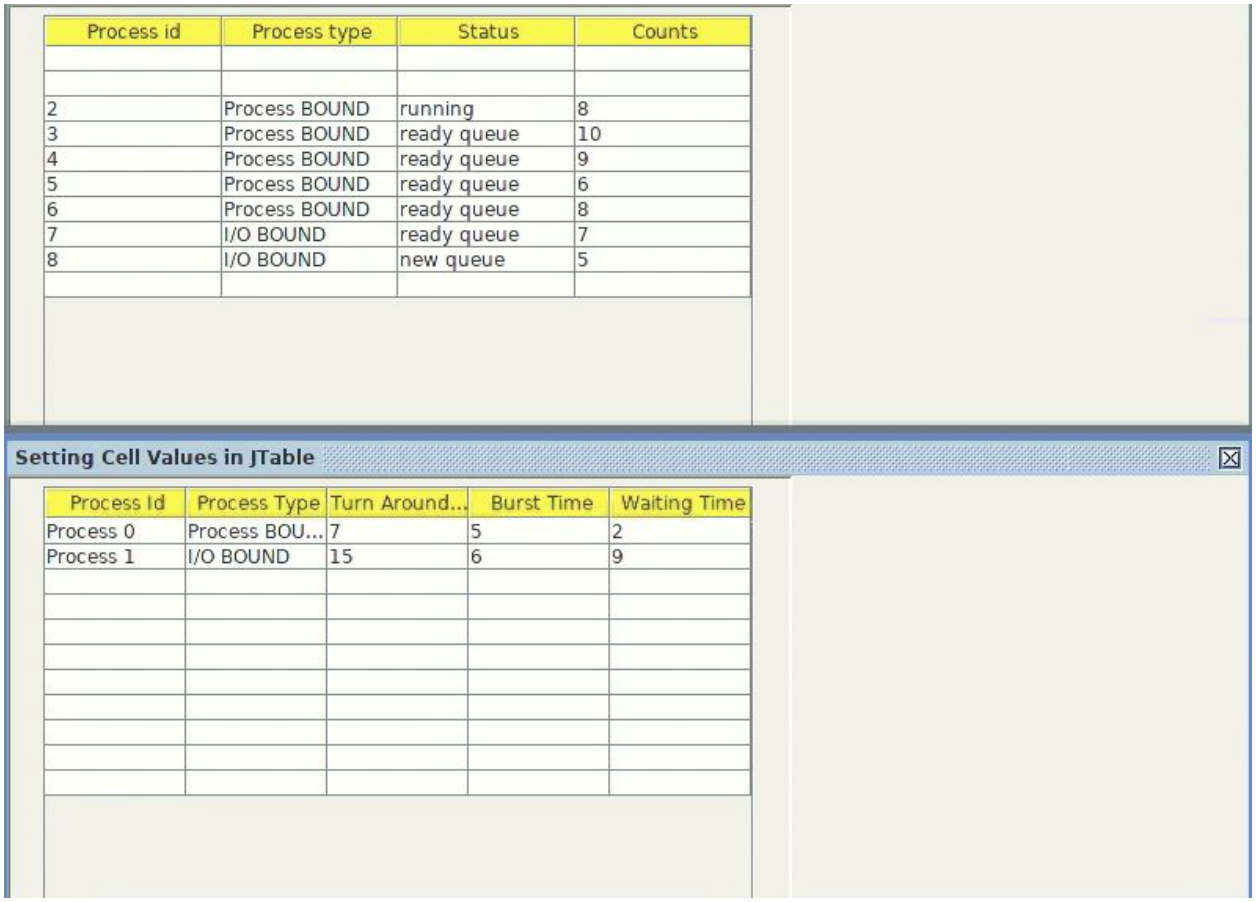
Classes implemented in the Code

1. **Process Creation**
    a. A random number is generated between 5 to 10. Now, a decision over the number is taken for creating a process or not.
    b. Once that decision is taken a new process is then created which is added to new queue.
    c. When number of processes is greater than a threshold the long term scheduler is invoked for further execution of program. This is done so that once processes are generated the execution will not have to remain idle.
2. **Long Term Scheduler**
    a. After the new processes are generated the work of long term scheduler is to shift the new processes to ready queue.
    b. After every space that is created in ready queue the long term scheduler checks for process in new queue and shifts to ready queue.
    c. This way multiprogramming is taken care of using long term scheduler.
3. **Execution**
    a. In this class the execution of processes is done in a Round-Robin fashion.
    b. The process in the ready queue is removed and brought in for execution. Here the while loop is kept running for the time quantum or for process count is not zero i.e. process completion.
    c. Afterwards the process is checked for process type. If the process type is “I/O Bound” the process is shifted to block queue to wait for the event to occur. Else the process is placed at the end of the ready queue for further execution or it is removed from the program if it is completed.
4. **Medium Term Scheduler**
    a. The implementation of medium term scheduler is to make shifts between processes from block queue to ready queue.
    b. For an event to occur in real OS implementation many interrupts and signals are used.
    c. Here, we implement it in a different manner where again a random number is used for taking decisions for event occurrence.
    d. A number between 5 to 10 is generated and when the number is equal to 10 an event occurrence is portrayed.
    e. The process then moves from block queue to running state, which states high priority for block process when an event occurs.

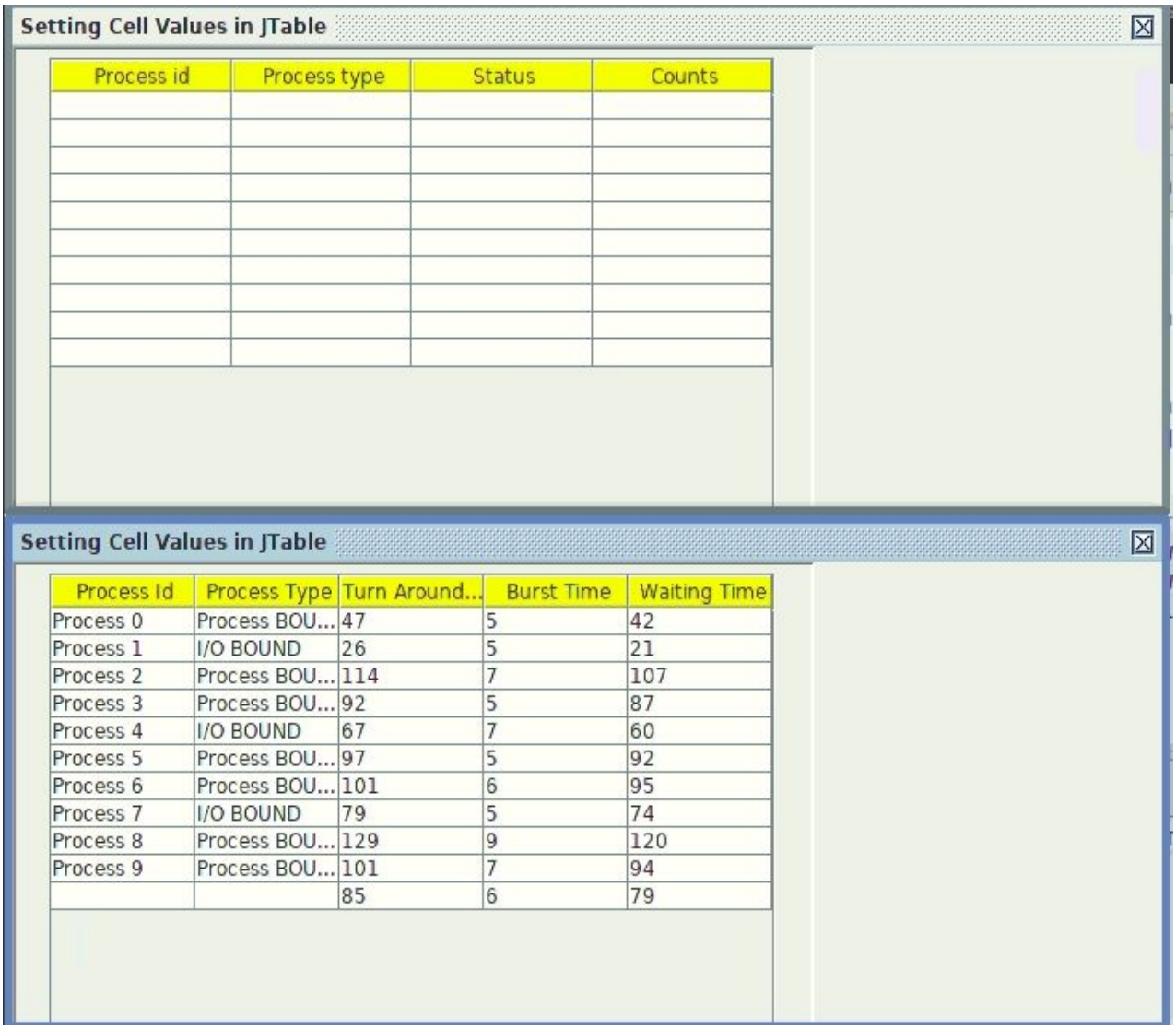
5. **Analysis**
    a. This class is implemented to make analysis for processes which are created and serviced by the program.
    b. Different parameters Start Time, End Time, Turn Around Time, Burst Time, Completion Time etc. are calculated.
    c. These are done by real time analysis using System call for time i.e. System.currentTimeMillis()
6. **Process**
    a. This class is the thread or process that are scheduled in our program.
    b. Process is given attributes like Process Id, Process Type, Process Count.

GUI Implementation

1. **FCFS Algorithm**



2. Round Robin Algorithm



Assumptions

1. In our case, the processes are shifted to ready queue only when the new queue size is greater than or equal to 5. This is done in order to show a transition of a process from new queue to the ready queue(task of long term scheduler).
2. A new process is generated only when a random number generated in the range of 5-10 gives a number which is either 8,9 or 10. Also another random number is used to decide for “Processor Bound” or “I/O Bound”.
3. A process is shifted from block queue to ready queue only when a random

number generated in the range of 5-10 gives 10.

4. The system works for 20 processes along with GUI display and can work for any number of processes without GUI display.
5. Whenever an I/O bound process moves to ready queue from the block queue it is directly sent for running.
6. The size of the ready queue is fixed which is 6.

REFERENCES

1. https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/
2. https://www.tutorialspoint.com/operating_system/os_process_scheduling_algorithms.htm
3. https://www.geeksforgeeks.org/program-round-robin-scheduling-set-1/
4. https://www.geeksforgeeks.org/program-for-fcfs-cpu-scheduling-set-1/
5. https://www.geeksforgeeks.org/process-schedulers-in-operating-system/