Exp. 8

**Problem Statement:**

Implement the Round Robin Algorithm.

CODE:-

import java.util.LinkedList;

import java.util.Queue;

import java.util.Scanner;

class Process {

int id; // Process ID

int burstTime; // Burst time of the process

int remainingTime; // Remaining time of the process

public Process(int id, int burstTime) {

this.id = id;

this.burstTime = burstTime;

this.remainingTime = burstTime;

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input number of processes

System.out.print("Enter the number of processes: ");

int n = scanner.nextInt();

// Input burst times for each process

Queue<Process> processQueue = new LinkedList<>();

for (int i = 0; i < n; i++) {

System.out.print("Enter burst time for process " + (i + 1) + ": ");

int burstTime = scanner.nextInt();

processQueue.add(new Process(i + 1, burstTime));

}

// Input time quantum

System.out.print("Enter time quantum: ");

int timeQuantum = scanner.nextInt();

// Execute Round Robin Scheduling

int currentTime = 0;

System.out.println("\nProcess Execution Order:");

while (!processQueue.isEmpty()) {

Process currentProcess = processQueue.poll();

if (currentProcess.remainingTime > timeQuantum) {

// Process can't finish within this quantum

currentTime += timeQuantum;

currentProcess.remainingTime -= timeQuantum;

System.out.println("Process " + currentProcess.id + " executed for " + timeQuantum + " units. Remaining time: " + currentProcess.remainingTime);

processQueue.add(currentProcess); // Add back to the queue

} else {

// Process will finish in this quantum

currentTime += currentProcess.remainingTime;

System.out.println("Process " + currentProcess.id + " executed for " + currentProcess.remainingTime + " units. Process completed.");

currentProcess.remainingTime = 0;

}

}

System.out.println("\nAll processes completed.");

scanner.close();

}

}

OUTPUT:-

