# ASSIGNMENT NO – 5

**Scheduler.java file**

```
import java.util.*;

// Base class for Scheduling algorithms
abstract class SchedulingAlgorithm {
    protected int n; // Number of processes
    protected Process[] processes; // Array to store processes
    public SchedulingAlgorithm(int n) {
        this.n = n;
        this.processes = new Process[n];
    }
    // Abstract method to be implemented by each scheduling algorithm
    public abstract void schedule();
    // Method to print process details
    protected void printResults() {
        System.out.println("Process\tAT\t\tBT\t\tCT\t\tTAT\tWT");
        System.out.println("---------------------------------------------------------------------");
        for (Process p : processes) {
            System.out.println(p);
        }
    }
    // Method to calculate average turnaround and waiting time
    protected void calculateAverages() {
        double avgTAT = 0, avgWT = 0;
        for (Process p : processes) {
            avgTAT += p.turnaroundTime;
            avgWT += p.waitingTime;
        }
```

```
        System.out.println("\nAverage Turnaround Time: " + (avgTAT / n) + "ms");
        System.out.println("Average Waiting Time: " + (avgWT / n) + "ms");
    }
}
// Class to represent a process
class Process {
    int processID;
    int arrivalTime;
    int burstTime;
    int completionTime;
    int turnaroundTime;
    int waitingTime;
    int remainingTime; // For preemptive algorithms
    public Process(int processID, int arrivalTime, int burstTime) {
        this.processID = processID;
        this.arrivalTime = arrivalTime;
        this.burstTime = burstTime;
        this.remainingTime = burstTime; // Initially, remainingTime is equal to burst time
    }
    @Override
    public String toString() {
        return "P" + processID + "\t\t" + arrivalTime + "ms\t\t" + burstTime + "ms\t\t"
                + completionTime + "ms\t\t\t" + turnaroundTime + "ms\t\t\t";
    }
}
// FCFS Scheduling Algorithm (First-Come-First-Serve)
class FCFS extends SchedulingAlgorithm {
    public FCFS(int n) {
        super(n);
```

```
    }

    @Override
    public void schedule() {
        Arrays.sort(processes, Comparator.comparingInt(p -> p.arrivalTime)); // Sort by arrival time
        int currentTime = 0;

        // Calculate Completion Time, Turnaround Time, Waiting Time for FCFS
        for (Process p : processes) {
            if (currentTime < p.arrivalTime) {
                currentTime = p.arrivalTime; // Process arrives later, so we wait until arrival
            }
            p.completionTime = currentTime + p.burstTime;
            p.turnaroundTime = p.completionTime - p.arrivalTime;
            p.waitingTime = p.turnaroundTime - p.burstTime;
            currentTime = p.completionTime;
        }

        printResults();
        calculateAverages();
    }
}

// Shortest Job First Scheduling Algorithm (Non-Preemptive)
class SJFNonPreemptive extends SchedulingAlgorithm {

    public SJFNonPreemptive(int n) {
        super(n);
    }
```

```
@Override
public void schedule() {
    Arrays.sort(processes, Comparator.comparingInt(p -> p.arrivalTime)); // Sort by arrival time
    int currentTime = 0;

    // Calculate Completion Time, Turnaround Time, Waiting Time for SJF
    for (int i = 0; i < n; i++) {
        Process p = getShortestJob(currentTime);
        currentTime += p.burstTime;
        p.completionTime = currentTime;
        p.turnaroundTime = p.completionTime - p.arrivalTime;
        p.waitingTime = p.turnaroundTime - p.burstTime;
    }

    printResults();
    calculateAverages();
}

private Process getShortestJob(int currentTime) {
    Process shortest = null;
    for (Process p : processes) {
        if (p.arrivalTime <= currentTime && p.remainingTime > 0 &&
            (shortest == null || p.burstTime < shortest.burstTime)) {
            shortest = p;
        }
    }
    return shortest;
}
```

```
}

// Priority Scheduling Algorithm (Non-Preemptive)
class PriorityScheduling extends SchedulingAlgorithm {

    public PriorityScheduling(int n) {
        super(n);
    }

    @Override
    public void schedule() {
        Arrays.sort(processes, Comparator.comparingInt(p -> p.arrivalTime)); // Sort by arrival time
        int currentTime = 0;

        // Calculate Completion Time, Turnaround Time, Waiting Time for Priority Scheduling
        for (Process p : processes) {
            if (currentTime < p.arrivalTime) {
                currentTime = p.arrivalTime; // Process arrives later, so we wait until arrival
            }
            p.completionTime = currentTime + p.burstTime;
            p.turnaroundTime = p.completionTime - p.arrivalTime;
            p.waitingTime = p.turnaroundTime - p.burstTime;
            currentTime = p.completionTime;
        }

        printResults();
        calculateAverages();
    }
}
```

```
// Main class
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Welcome to the Process Scheduling Simulator");
        System.out.print("Enter number of processes: ");
        int n = sc.nextInt();

        // Create the array of processes
        Process[] processes = new Process[n];
        for (int i = 0; i < n; i++) {
            System.out.print("Enter Arrival Time for Process P" + (i + 1) + ": ");
            int arrivalTime = sc.nextInt();
            System.out.print("Enter Burst Time for Process P" + (i + 1) + ": ");
            int burstTime = sc.nextInt();
            processes[i] = new Process(i + 1, arrivalTime, burstTime);
        }

        // Menu for selecting scheduling algorithm
        System.out.println("\nSelect Scheduling Algorithm:");
        System.out.println("1. First Come First Serve (FCFS)");
        System.out.println("2. Shortest Job First (SJF) - Non Preemptive");
        System.out.println("3. Priority Scheduling - Non Preemptive");
        System.out.print("Enter your choice: ");
        int choice = sc.nextInt();

        SchedulingAlgorithm algorithm = null;
```

```
        switch (choice) {
            case 1:
                algorithm = new FCFS(n);
                break;
            case 2:
                algorithm = new SJFNonPreemptive(n);
                break;
            case 3:
                algorithm = new PriorityScheduling(n);
                break;
            default:
                System.out.println("Invalid choice!");
                return;
        }
        // Assign the processes to the chosen algorithm and run it
        algorithm.processes = processes;
        algorithm.schedule();
    }
}
```

**OUTPUT:**

```
Welcome to the Process Scheduling Simulator
Enter number of processes: 3
Enter Arrival Time for Process P1: 0
Enter Burst Time for Process P1: 3
Enter Arrival Time for Process P2: 2
Enter Burst Time for Process P2: 5
Enter Arrival Time for Process P3: 5
Enter Burst Time for Process P3: 7
```

**Select Scheduling Algorithm:**

**1. First Come First Serve (FCFS)**

**2. Shortest Job First (SJF) - Non Preemptive**

**3. Priority Scheduling - Non Preemptive**

**Enter your choice: 1**

| Process | AT | BT | CT | TAT | WT |
|---|---|---|---|---|---|
| P1 | 0ms | 3ms | 3ms | | 3ms |
| P2 | 2ms | 5ms | 8ms | | 6ms |
| P3 | 5ms | 7ms | 15ms | | 10ms |

**Average Turnaround Time: 6.333333333333333ms**

**Average Waiting Time: 1.3333333333333333ms**

**=== Code Execution Successful ===**

---

**Welcome to the Process Scheduling Simulator**

**Enter number of processes: 3**

**Enter Arrival Time for Process P1: 0**

**Enter Burst Time for Process P1: 3**

**Enter Arrival Time for Process P2: 2**

**Enter Burst Time for Process P2: 5**

**Enter Arrival Time for Process P3: 5**

**Enter Burst Time for Process P3: 7**

**Select Scheduling Algorithm:**

**1. First Come First Serve (FCFS)**

**2. Shortest Job First (SJF) - Non Preemptive**

**3. Priority Scheduling - Non Preemptive**

**Enter your choice: 2**

| Process | AT | BT | CT | TAT | WT |
|---|---|---|---|---|---|
| P1 | 0ms | 3ms | 9ms | | 9ms |
| P2 | 2ms | 5ms | 0ms | | 0ms |
| P3 | 5ms | 7ms | 0ms | | 0ms |

**Average Turnaround Time: 3.0ms**

**Average Waiting Time: 2.0ms**

**=== Code Execution Successful ===**

---

**Welcome to the Process Scheduling Simulator**

**Enter number of processes: 3**

**Enter Arrival Time for Process P1: 0**

**Enter Burst Time for Process P1: 3**

**Enter Arrival Time for Process P2: 2**

**Enter Burst Time for Process P2: 5**

**Enter Arrival Time for Process P3: 5**

**Enter Burst Time for Process P3: 7**

**Select Scheduling Algorithm:**

**1. First Come First Serve (FCFS)**

**2. Shortest Job First (SJF) - Non Preemptive**

**3. Priority Scheduling - Non Preemptive**

**Enter your choice: 3**

| Process | AT | BT | CT | TAT |
|---|---|---|---|---|
| P1 | 0ms | 3ms | 3ms | 3ms |

| | | | | |
|---|---|---|---|---|
| **P2** | **2ms** | **5ms** | **8ms** | **6ms** |
| **P3** | **5ms** | **7ms** | **15ms** | **10ms** |

**Average Turnaround Time: 6.333333333333333ms**

**Average Waiting Time: 1.3333333333333333ms**

**=== Code Execution Successful ===**