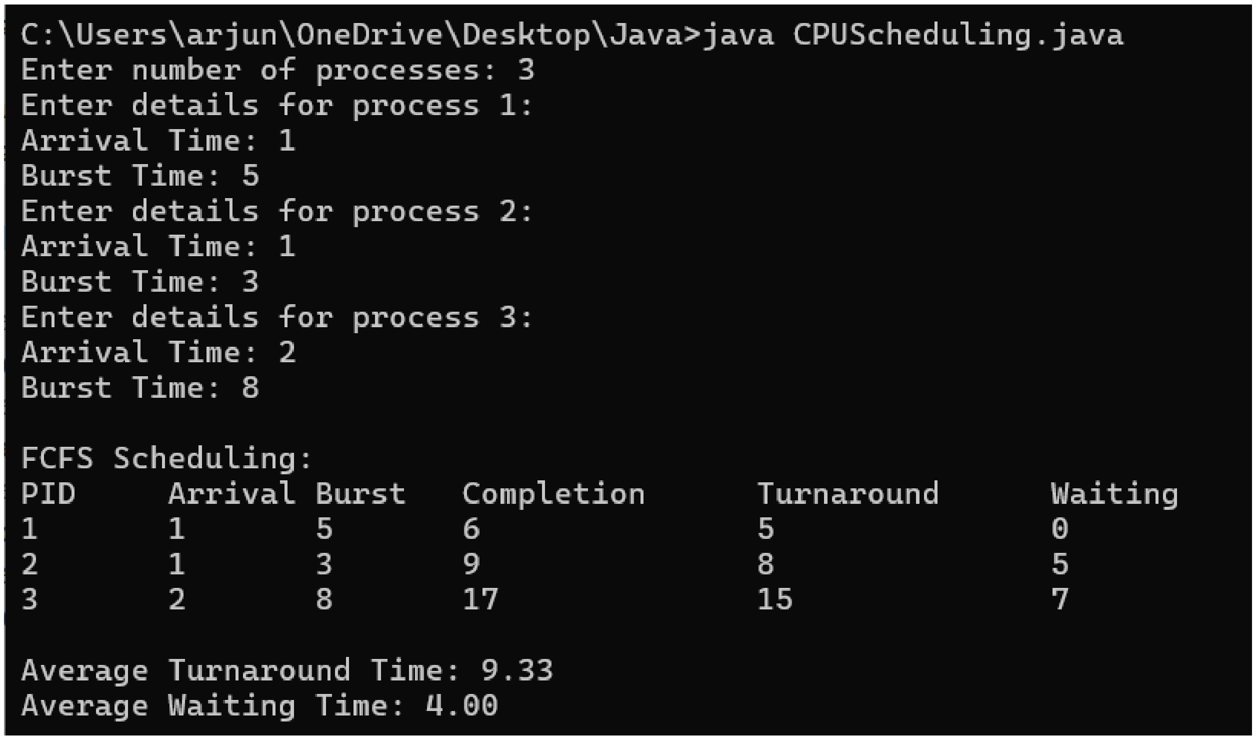
Arjun Chandrakant Prabhu D10B/54

Experiment No 10: Write a program to demonstrate any two CPU scheduling algorithms like FCFS, SJF, SRT,Round Robin,Priority.

1)FCFS(First Come First Serve)

```
import java.util.Arrays;
import java.util.Comparator;
import java.util.Scanner;

public class CPUScheduling {

    // Nested Process class
    static class Process {
        int pid;        // Process ID
        int arrivalTime;
        int burstTime;
        int completionTime;
        int waitingTime;
        int turnaroundTime;

        Process(int pid, int arrivalTime, int burstTime) {
            this.pid = pid;
            this.arrivalTime = arrivalTime;
            this.burstTime = burstTime;
        }
    }

    // FCFS Scheduling Algorithm
    public static void fcfs(Process[] processes) {
        int n = processes.length;
        Arrays.sort(processes, Comparator.comparingInt(p -> p.arrivalTime));

        int currentTime = 0;
        for (Process p : processes) {
            if (currentTime < p.arrivalTime) {
                currentTime = p.arrivalTime;
            }
            p.completionTime = currentTime + p.burstTime;
            p.turnaroundTime = p.completionTime - p.arrivalTime;
            p.waitingTime = p.turnaroundTime - p.burstTime;
            currentTime = p.completionTime;
        }

        System.out.println("\nFCFS Scheduling:");
        displayResults(processes);
    }

    // Display results
    public static void displayResults(Process[] processes) {
        System.out.println("PID\tArrival\tBurst\tCompletion\tTurnaround\tWaiting");
        for (Process p : processes) {
            System.out.println(p.pid + "\t" + p.arrivalTime + "\t" + p.burstTime + "\t" +
                    p.completionTime + "\t\t" + p.turnaroundTime + "\t\t" + p.waitingTime);
        }
        calculateAverage(processes);
    }

    // Calculate and display average turnaround and waiting time
    public static void calculateAverage(Process[] processes) {
        int n = processes.length;
        double totalTurnaround = 0, totalWaiting = 0;

        for (Process p : processes) {
            totalTurnaround += p.turnaroundTime;
            totalWaiting += p.waitingTime;
        }

        System.out.printf("\nAverage Turnaround Time: %.2f\n", (totalTurnaround / n));
        System.out.printf("Average Waiting Time: %.2f\n", (totalWaiting / n));
    }

    // Main method
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of processes: ");
        int n = sc.nextInt();

        Process[] processes = new Process[n];
        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for process " + (i + 1) + ":");
            System.out.print("Arrival Time: ");
            int arrivalTime = sc.nextInt();
            System.out.print("Burst Time: ");
            int burstTime = sc.nextInt();
            processes[i] = new Process(i + 1, arrivalTime, burstTime);
        }

        // Execute FCFS
        fcfs(processes);
    }
}
```

Example/Output:



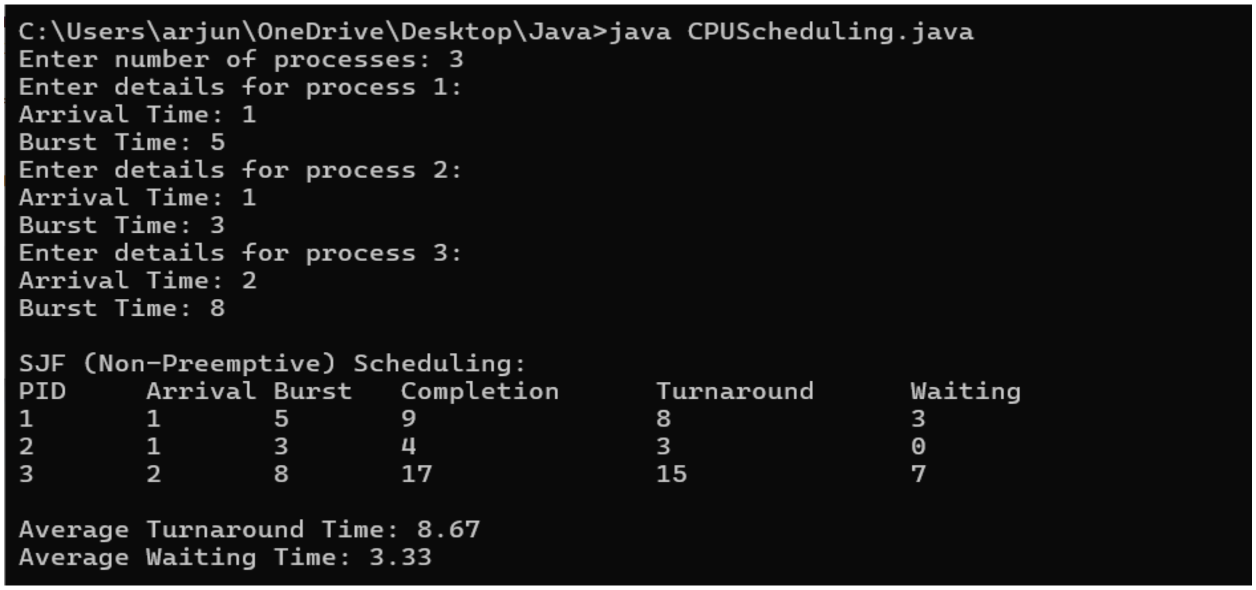
```
C:\Users\arjun\OneDrive\Desktop\Java>java CPUScheduling.java
Enter number of processes: 3
Enter details for process 1:
Arrival Time: 1
Burst Time: 5
Enter details for process 2:
Arrival Time: 1
Burst Time: 3
Enter details for process 3:
Arrival Time: 2
Burst Time: 8

FCFS Scheduling:
PID     Arrival Burst   Completion      Turnaround      Waiting
1       1       5       6               5               0
2       1       3       9               8               5
3       2       8       17              15              7

Average Turnaround Time: 9.33
Average Waiting Time: 4.00
```

2)SJSF(SHORTEST JOB FIRST SCHEDULING) : NON-PREEMPTIVE.

```
import java.util.Arrays;
import java.util.Comparator;
import java.util.Scanner;

public class CPUScheduling {

   // Nested Process class
   static class Process {
      int pid;        // Process ID
      int arrivalTime;
      int burstTime;
      int completionTime;
      int waitingTime;
      int turnaroundTime;
      boolean completed;

      Process(int pid, int arrivalTime, int burstTime) {
         this.pid = pid;
         this.arrivalTime = arrivalTime;
         this.burstTime = burstTime;
         this.completed = false;
      }
   }

   // SJF Non-Preemptive Scheduling Algorithm
   public static void sjf(Process[] processes) {
      int n = processes.length;
      int currentTime = 0, completedProcesses = 0;

      while (completedProcesses < n) {
         int shortestIndex = -1;
         int shortestBurst = Integer.MAX_VALUE;

         // Find the shortest burst time among available processes
         for (int i = 0; i < n; i++) {
            if (!processes[i].completed && processes[i].arrivalTime <= currentTime) {
               if (processes[i].burstTime < shortestBurst) {
                  shortestBurst = processes[i].burstTime;
                  shortestIndex = i;
```

```
                }
            }
        }

        if (shortestIndex == -1) {
            // No process available, increment time
            currentTime++;
        } else {
            Process p = processes[shortestIndex];
            p.completionTime = currentTime + p.burstTime;
            p.turnaroundTime = p.completionTime - p.arrivalTime;
            p.waitingTime = p.turnaroundTime - p.burstTime;
            currentTime = p.completionTime;
            p.completed = true;
            completedProcesses++;
        }
    }

    System.out.println("\nSJF (Non-Preemptive) Scheduling:");
    displayResults(processes);
}

// Display results
public static void displayResults(Process[] processes) {
    System.out.println("PID\tArrival\tBurst\tCompletion\tTurnaround\tWaiting");
    for (Process p : processes) {
        System.out.println(p.pid + "\t" + p.arrivalTime + "\t" + p.burstTime + "\t" +
                p.completionTime + "\t\t" + p.turnaroundTime + "\t\t" + p.waitingTime);
    }
    calculateAverage(processes);
}

// Calculate and display average turnaround and waiting time
public static void calculateAverage(Process[] processes) {
    int n = processes.length;
    double totalTurnaround = 0, totalWaiting = 0;

    for (Process p : processes) {
        totalTurnaround += p.turnaroundTime;
        totalWaiting += p.waitingTime;
        }

        System.out.printf("\nAverage Turnaround Time: %.2f\n", (totalTurnaround / n));
        System.out.printf("Average Waiting Time: %.2f\n", (totalWaiting / n));
    }
// Main method
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of processes: ");
        int n = sc.nextInt();

        Process[] processes = new Process[n];
        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for process " + (i + 1) + ":");
            System.out.print("Arrival Time: ");
            int arrivalTime = sc.nextInt();
            System.out.print("Burst Time: ");
            int burstTime = sc.nextInt();
            processes[i] = new Process(i + 1, arrivalTime, burstTime);
        }

        // Execute SJF Non-Preemptive
        sjf(processes);
    }
}
```

Example/Output:



```
C:\Users\arjun\OneDrive\Desktop\Java>java CPUScheduling.java
Enter number of processes: 3
Enter details for process 1:
Arrival Time: 1
Burst Time: 5
Enter details for process 2:
Arrival Time: 1
Burst Time: 3
Enter details for process 3:
Arrival Time: 2
Burst Time: 8

SJF (Non-Preemptive) Scheduling:
PID     Arrival Burst   Completion      Turnaround      Waiting
1       1       5       9               8               3
2       1       3       4               3               0
3       2       8       17              15              7

Average Turnaround Time: 8.67
Average Waiting Time: 3.33
```