NAME: TEJAS NALAWADE PRACTICAL NO: 5 ROLL NO: TCOD01

**Assignment Title**: Implement following process scheduling algorithms: FCFS , SJF (Preemptive), Priority (Non-Preemptive).

**Problem Statement**: Write a Java program (using OOP features) to implement following scheduling algorithms: FCFS, SJF (Preemptive), Priority (Non-Preemptive).

**FCFS**

**CODE**:

import java.util.\*;

class Process {

    int processId;

    int arrivalTime;

    int burstTime;

    int completionTime;

    int turnaroundTime;

    int waitingTime;

    public Process(int processId, int arrivalTime, int burstTime) {

        this.processId = processId;

        this.arrivalTime = arrivalTime;

        this.burstTime = burstTime;

    }

}

public class Fcfs {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

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

        int n = sc.nextInt();

        List<Process> processes = new ArrayList<>();

        for (int i = 0; i < n; i++) {

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

            int arrival = sc.nextInt();

            int burst = sc.nextInt();

            processes.add(new Process(i + 1, arrival, burst));

        }

        processes.sort(Comparator.comparingInt(p -> p.arrivalTime));

        int currentTime = 0;

        double totalTAT = 0;

        double totalWT = 0;

        // Calculate times

        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;

            totalTAT += p.turnaroundTime;

            totalWT += p.waitingTime;

        }

        System.out.println("\nProcess\tAT\tBT\tCT\tTAT\tWT");

        for (Process p : processes) {

            System.out.println("P" + p.processId + "\t" +

                    p.arrivalTime + "\t" +

                    p.burstTime + "\t" +

                    p.completionTime + "\t" +

                    p.turnaroundTime + "\t" +

                    p.waitingTime);

        }

        double avgTAT = totalTAT / n;

        double avgWT = totalWT / n;

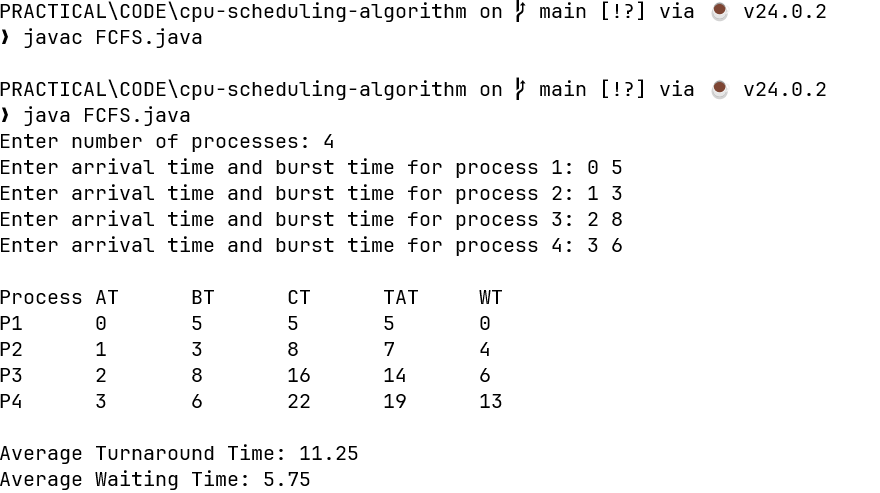
        System.out.printf("\nAverage Turnaround Time: %.2f\n", avgTAT);

        System.out.printf("Average Waiting Time: %.2f\n", avgWT);

    }

}

**OUTPUT**:



**SJF(Preemptive)**

**CODE:**

import java.util.\*;

class Process {

    int processId;

    int arrivalTime;

    int burstTime;

    int remainingTime;

    int completionTime;

    int turnaroundTime;

    int waitingTime;

    boolean isCompleted;

    public Process(int processId, int arrivalTime, int burstTime) {

        this.processId = processId;

        this.arrivalTime = arrivalTime;

        this.burstTime = burstTime;

        this.remainingTime = burstTime;

        this.isCompleted = false;

    }

}

public class SJFPreemptive {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

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

        int n = sc.nextInt();

        List<Process> processes = new ArrayList<>();

        for (int i = 0; i < n; i++) {

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

            int at = sc.nextInt();

            int bt = sc.nextInt();

            processes.add(new Process(i + 1, at, bt));

        }

        int currentTime = 0;

        int completed = 0;

        double totalTAT = 0;

        double totalWT = 0;

        while (completed < n) {

            Process shortest = null;

            for (Process p : processes) {

                if (p.arrivalTime <= currentTime && !p.isCompleted && p.remainingTime > 0) {

                    if (shortest == null || p.remainingTime < shortest.remainingTime) {

                        shortest = p;

                    }

                }

            }

            if (shortest != null) {

                shortest.remainingTime--;

                currentTime++;

                if (shortest.remainingTime == 0) {

                    shortest.isCompleted = true;

                    shortest.completionTime = currentTime;

                    shortest.turnaroundTime = shortest.completionTime - shortest.arrivalTime;

                    shortest.waitingTime = shortest.turnaroundTime - shortest.burstTime;

                    totalTAT += shortest.turnaroundTime;

                    totalWT += shortest.waitingTime;

                    completed++;

                }

            } else {

                currentTime++;

            }

        }

        System.out.println("\nProcess\tAT\tBT\tCT\tTAT\tWT");

        for (Process p : processes) {

            System.out.println("P" + p.processId + "\t" +

                    p.arrivalTime + "\t" +

                    p.burstTime + "\t" +

                    p.completionTime + "\t" +

                    p.turnaroundTime + "\t" +

                    p.waitingTime);

        }

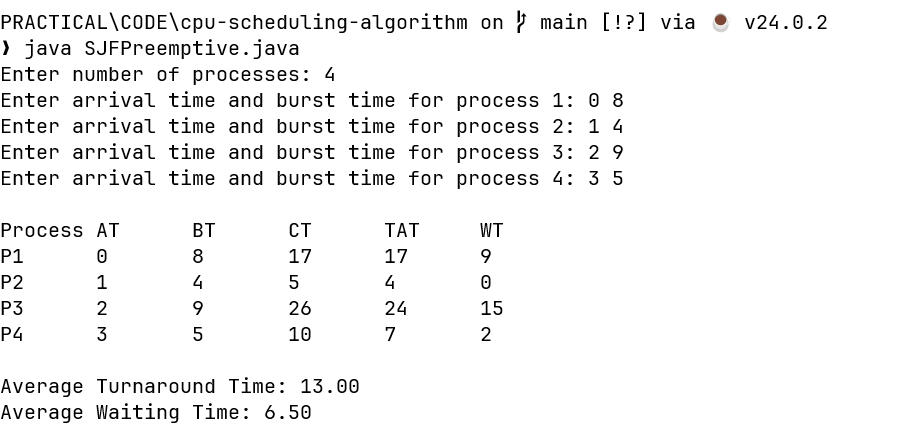
        System.out.printf("\nAverage Turnaround Time: %.2f\n", totalTAT / n);

        System.out.printf("Average Waiting Time: %.2f\n", totalWT / n);

    }

}

**OUTPUT:**



**Priority (Non-Preemptive)**

**CODE:**

import java.util.\*;

class Process {

    int pid, at, bt, ct, tat, wt;

    boolean completed = false;

    Process(int pid, int at, int bt) {

        this.pid = pid;

        this.at = at;

        this.bt = bt;

    }

}

public class Sjf\_non\_preemptive {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

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

        int n = sc.nextInt();

        double twt=0;

        double ttat=0;

        List<Process> list = new ArrayList<>();

        for (int i = 0; i < n; i++) {

            System.out.print("Enter AT and BT for P" + (i + 1) + ": ");

            list.add(new Process(i + 1, sc.nextInt(), sc.nextInt()));

        }

        int time = 0, completed = 0;

        while (completed < n) {

            Process shortest = null;

            for (Process p : list) {

                if (!p.completed && p.at <= time) {

                    if (shortest == null || p.bt < shortest.bt)

                        shortest = p;

                }

            }

            if (shortest == null) {

                time++;

            } else {

                shortest.ct = time + shortest.bt;

                shortest.tat = shortest.ct - shortest.at;

                shortest.wt = shortest.tat - shortest.bt;

                shortest.completed = true;

                time = shortest.ct;

                completed++;

                twt=twt+shortest.wt;

                ttat=ttat+shortest.tat;

            }

        }

        System.out.println("\nPID\tAT\tBT\tCT\tTAT\tWT");

        for (Process p : list)

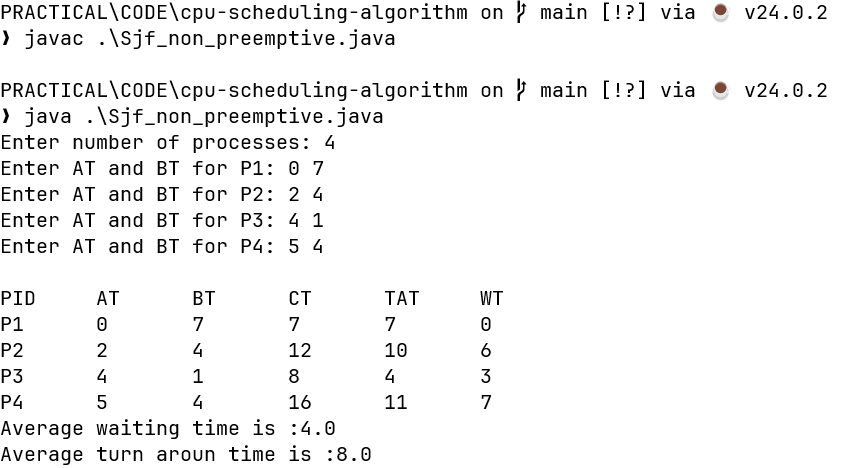
            System.out.println("P" + p.pid + "\t" + p.at + "\t" + p.bt + "\t" + p.ct + "\t" + p.tat + "\t" + p.wt);

        System.out.println("Average waiting time is :"+(twt/n));

        System.out.println("Average turn aroun time is :"+(ttat/n));

    }

}

**OUTPUT: **