# Bansilal Ramnath Agarwal Charitable Trust’s

Vishwakarma Institute of Technology, Pune-37

*(Autonomous Institute of Savitribai Phule Pune University)*



**Department of Multidisciplinary Engineering**

|  |  |
| --- | --- |
| **Division** | AI-B |
| **Batch** | 1 |
| **GR-no** | 12210457 |
| **Rollno** | 17 |
| **Name** | Mohan Kabade |

**Title:** Write a program to compute the finish time, turnaround time and waiting time for the following algorithms: a) First come First serve b) Shortest Job First (Preemptive and Non Preemptive)  c) Priority (Preemptive and Non Preemptive) d) Round robin

**a) First come First serve:**

**Code:**

import java.util.Scanner;

import java.util.Arrays;

public class FCFS {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int n = sc.nextInt();

int[] bt = new int[n];

int[] at = new int[n];

System.out.println("\nEnter the Arrival Time for each process.");

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

System.out.print("\nFor Process " + (i + 1) + ": ");

at[i] = sc.nextInt();

}

System.out.println("\nEnter the Burst Time for each process.");

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

System.out.print("\nFor Process " + (i + 1) + ": ");

bt[i] = sc.nextInt();

}

avg\_wt\_tt(n, at, bt);

sc.close();

}

private static void waiting\_time(int n, int[] at, int[] bt, int[] wt) {

int[] service\_time = new int[n];

service\_time[0] = at[0];

wt[0] = 0;

for (int i = 1; i < n; i++) {

service\_time[i] = service\_time[i-1] + bt[i-1];

wt[i] = service\_time[i] - at[i];

if (wt[i] < 0) {

wt[i] = 0;

}

}

}

private static void turnaround\_time(int n, int[] bt, int[] wt, int[] tt) {

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

tt[i] = bt[i] + wt[i];

}

}

private static void avg\_wt\_tt(int n, int[] at, int[] bt) {

int[] wt = new int[n];

int[] tt = new int[n];

int[][] processes = new int[n][3];

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

processes[i][0] = i + 1;

processes[i][1] = at[i];

processes[i][2] = bt[i];

}

Arrays.sort(processes, (a, b) -> a[1] - b[1]);

int[] sorted\_bt = new int[n];

int[] sorted\_at = new int[n];

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

sorted\_at[i] = processes[i][1];

sorted\_bt[i] = processes[i][2];

}

waiting\_time(n, sorted\_at, sorted\_bt, wt);

turnaround\_time(n, sorted\_bt, wt, tt);

System.out.println("\nProcesses || Arrival Time || Burst Time || Waiting Time || Turn-Around Time ");

float awt = 0;

float att = 0;

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

awt += wt[i];

att += tt[i];

System.out.println(processes[i][0] + "\t||\t" + sorted\_at[i] + "\t||\t" + sorted\_bt[i] + "\t||\t" + wt[i] + "\t||\t " + tt[i]);

}

awt /= n;

att /= n;

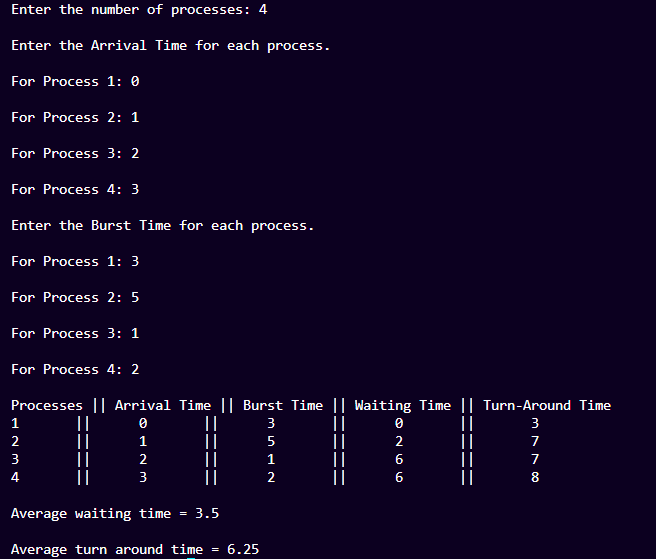
System.out.println("\nAverage waiting time = " + awt);

System.out.println("\nAverage turn around time = " + att);

}

}

**Output :**



**b) Shortest Job First:**

**i) Preemptive:**

**Code:**

import java.util.Scanner;

class Process

{

int pid;

int bt;

int art;

public Process(int pid, int bt, int art)

{

this.pid = pid;

this.bt = bt;

this.art = art;

}

}

public class sjfpreem

{

static void findWaitingTime(Process proc[], int n, int wt[])

{

int rt[] = new int[n];

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

rt[i] = proc[i].bt;

int complete = 0, t = 0, minm = Integer.MAX\_VALUE;

int shortest = 0, finish\_time;

boolean check = false;

while (complete != n) {

for (int j = 0; j < n; j++)

{

if ((proc[j].art <= t) && (rt[j] < minm) && rt[j] > 0) {

minm = rt[j];

shortest = j;

check = true;

}

}

if (check == false) {

t++;

continue;

}

rt[shortest]--;

minm = rt[shortest];

if (minm == 0)

minm = Integer.MAX\_VALUE;

if (rt[shortest] == 0) {

complete++;

check = false;

finish\_time = t + 1;

wt[shortest] = finish\_time - proc[shortest].bt - proc[shortest].art;

if (wt[shortest] < 0)

wt[shortest] = 0;

}

t++;

}

}

static void findTurnAroundTime(Process proc[], int n, int wt[], int tat[])

{

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

tat[i] = proc[i].bt + wt[i];

}

static void findavgTime(Process proc[], int n)

{

int wt[] = new int[n], tat[] = new int[n];

int total\_wt = 0, total\_tat = 0;

findWaitingTime(proc, n, wt);

findTurnAroundTime(proc, n, wt, tat);

System.out.println("Processes " + " Burst time " + " Waiting time " + " Turn around time");

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

total\_wt = total\_wt + wt[i];

total\_tat = total\_tat + tat[i];

System.out.println(" " + proc[i].pid + "\t\t" + proc[i].bt + "\t\t " + wt[i] + "\t\t" + tat[i]);

}

System.out.println("Average waiting time = " + (float)total\_wt / (float)n);

System.out.println("Average turn around time = " + (float)total\_tat / (float)n);

}

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

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

int numProcesses = sc.nextInt();

Process[] proc = new Process[numProcesses];

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

System.out.println("Enter process " + (i + 1) + " details:");

System.out.print("Process ID: ");

int pid = sc.nextInt();

System.out.print("Burst Time: ");

int bt = sc.nextInt();

System.out.print("Arrival Time: ");

int art = sc.nextInt();

proc[i] = new Process(pid, bt, art);

}

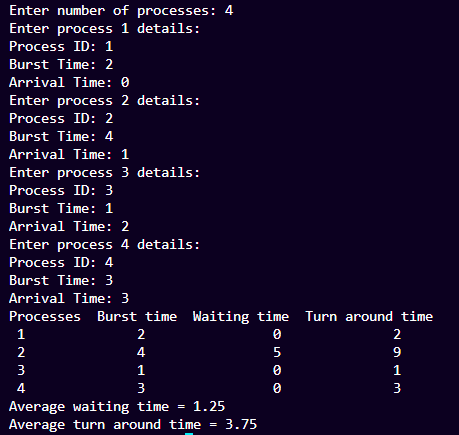
findavgTime(proc, proc.length);

sc.close();

}

}

**Output:**

****

**ii) Non Preemptive:**

**Code :**

import java.util.Scanner;

class Process {

int pid;

int bt;

int art;

public Process(int pid, int bt, int art) {

this.pid = pid;

this.bt = bt;

this.art = art;

}

}

public class sjfnonpreem {

static void findWaitingTime(Process proc[], int n, int wt[]) {

int[] completeTime = new int[n];

int[] rt = new int[n];

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

rt[i] = proc[i].bt;

int complete = 0, t = 0;

int minm = Integer.MAX\_VALUE;

int shortest = 0;

boolean check = false;

while (complete != n) {

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

if ((proc[j].art <= t) && (rt[j] < minm) && rt[j] > 0) {

minm = rt[j];

shortest = j;

check = true;

}

}

if (check == false) {

t++;

continue;

}

t += rt[shortest];

rt[shortest] = 0;

minm = Integer.MAX\_VALUE;

if (rt[shortest] == 0) {

complete++;

check = false;

completeTime[shortest] = t;

wt[shortest] = t - proc[shortest].bt - proc[shortest].art;

if (wt[shortest] < 0)

wt[shortest] = 0;

}

}

}

static void findTurnAroundTime(Process proc[], int n, int wt[], int tat[]) {

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

tat[i] = proc[i].bt + wt[i];

}

static void findavgTime(Process proc[], int n) {

int wt[] = new int[n], tat[] = new int[n];

int total\_wt = 0, total\_tat = 0;

findWaitingTime(proc, n, wt);

findTurnAroundTime(proc, n, wt, tat);

System.out.println("Processes " + " Burst time " + " Waiting time " + " Turn around time");

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

total\_wt = total\_wt + wt[i];

total\_tat = total\_tat + tat[i];

System.out.println(" " + proc[i].pid + "\t\t" + proc[i].bt + "\t\t " + wt[i] + "\t\t" + tat[i]);

}

System.out.println("Average waiting time = " + (float) total\_wt / (float) n);

System.out.println("Average turn around time = " + (float) total\_tat / (float) n);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int numProcesses = sc.nextInt();

Process[] proc = new Process[numProcesses];

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

System.out.println("Enter process " + (i + 1) + " details:");

System.out.print("Process ID: ");

int pid = sc.nextInt();

System.out.print("Burst Time: ");

int bt = sc.nextInt();

System.out.print("Arrival Time: ");

int art = sc.nextInt();

proc[i] = new Process(pid, bt, art);

}

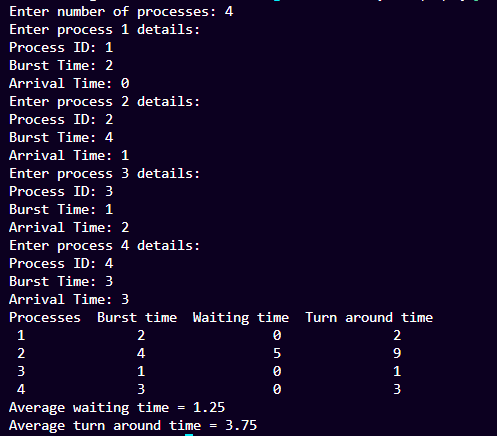
findavgTime(proc, proc.length);

sc.close();

}

}

**Output:**



**c) Priority:**

**i) Preemptive:**

**Code:**

import java.util.\*;

class Process {

String pid;

int bt, at, priority, rt;

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

this.pid = pid;

this.at = at;

this.bt = bt;

this.priority = priority;

this.rt = bt; // Remaining time is initialized to burst time

}

}

public class PriorityPreem {

static void findWaitingTime(Process proc[], int n, int wt[]) {

int t = 0; // Current time

int complete = 0; // Completed processes

int minPriority = Integer.MAX\_VALUE;

int shortest = 0;

boolean check = false;

while (complete != n) {

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

if ((proc[j].at <= t) && (proc[j].priority < minPriority) && (proc[j].rt > 0)) {

minPriority = proc[j].priority;

shortest = j;

check = true;

}

}

if (!check) {

t++;

continue;

}

proc[shortest].rt--;

minPriority = proc[shortest].rt == 0 ? Integer.MAX\_VALUE : proc[shortest].priority;

if (proc[shortest].rt == 0) {

complete++;

check = false;

int finish\_time = t + 1;

wt[shortest] = finish\_time - proc[shortest].bt - proc[shortest].at;

if (wt[shortest] < 0) {

wt[shortest] = 0;

}

}

t++;

}

}

static void findTurnAroundTime(Process proc[], int n, int wt[], int tat[]) {

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

tat[i] = proc[i].bt + wt[i];

}

}

static void findavgTime(Process proc[], int n) {

int wt[] = new int[n], tat[] = new int[n];

int total\_wt = 0, total\_tat = 0;

findWaitingTime(proc, n, wt);

findTurnAroundTime(proc, n, wt, tat);

System.out.println("Processes " + " Burst time " + " Waiting time " + " Turn around time");

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

total\_wt += wt[i];

total\_tat += tat[i];

System.out.println(" " + proc[i].pid + "\t\t" + proc[i].bt + "\t\t " + wt[i] + "\t\t " + tat[i]);

}

System.out.println("Average waiting time = " + (float) total\_wt / n);

System.out.println("Average turn around time = " + (float) total\_tat / n);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int numProcesses = sc.nextInt();

Process[] proc = new Process[numProcesses];

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

String pid = "P" + (i + 1);

System.out.println("Enter arrival time for " + pid + ":");

int at = sc.nextInt();

System.out.println("Enter burst time for " + pid + ":");

int bt = sc.nextInt();

System.out.println("Enter priority for " + pid + ":");

int priority = sc.nextInt();

proc[i] = new Process(pid, at, bt, priority);

}

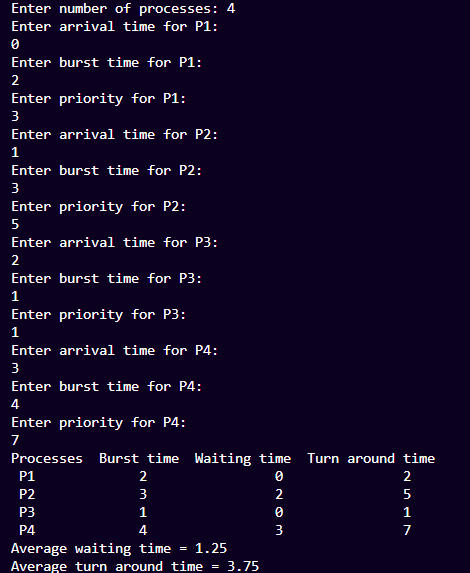
findavgTime(proc, numProcesses);

sc.close();

}

}

**Output:**



**ii) Non Preemptive:**

**Code:**

import java.util.\*;

class Process {

int at, bt, pr, pno;

Process(int pno, int at, int bt, int pr) {

this.pno = pno;

this.pr = pr;

this.at = at;

this.bt = bt;

}

}

public class PrioritySchedulingNonPreem {

static int totalprocess;

static Process[] proc;

static void get\_wt\_time(int[] wt) {

int[] service = new int[totalprocess];

service[0] = proc[0].at;

wt[0] = 0;

for (int i = 1; i < totalprocess; i++) {

service[i] = proc[i - 1].bt + service[i - 1];

wt[i] = service[i] - proc[i].at;

if (wt[i] < 0) {

wt[i] = 0;

}

}

}

static void get\_tat\_time(int[] tat, int[] wt) {

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

tat[i] = proc[i].bt + wt[i];

}

}

static void findgc() {

int[] wt = new int[totalprocess];

int[] tat = new int[totalprocess];

double wavg = 0, tavg = 0;

get\_wt\_time(wt);

get\_tat\_time(tat, wt);

int[] stime = new int[totalprocess];

int[] ctime = new int[totalprocess];

stime[0] = proc[0].at;

ctime[0] = stime[0] + tat[0];

for (int i = 1; i < totalprocess; i++) {

stime[i] = ctime[i - 1];

ctime[i] = stime[i] + tat[i] - wt[i];

}

System.out.println("Process\_no\tStart\_time\tComplete\_time\tTurn\_Around\_Time\tWaiting\_Time");

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

wavg += wt[i];

tavg += tat[i];

System.out.println(proc[i].pno + "\t\t" + stime[i] + "\t\t" + ctime[i] + "\t\t" + tat[i] + "\t\t\t" + wt[i]);

}

System.out.println("Average waiting time is : " + wavg / totalprocess);

System.out.println("Average turnaround time : " + tavg / totalprocess);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

totalprocess = sc.nextInt();

proc = new Process[totalprocess];

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

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

int at = sc.nextInt();

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

int bt = sc.nextInt();

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

int pr = sc.nextInt();

proc[i] = new Process(i + 1, at, bt, pr);

}

Arrays.sort(proc, (a, b) -> {

if (a.at == b.at) {

return a.pr - b.pr;

} else {

return a.at - b.at;

}

});

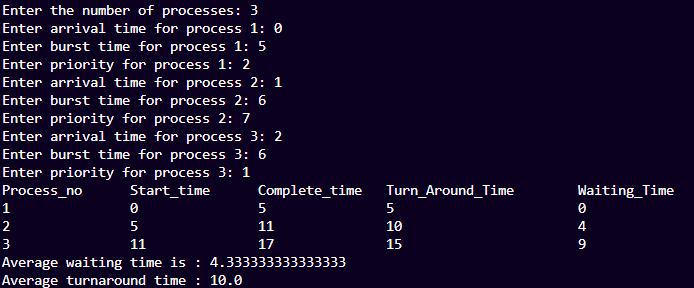
findgc();

sc.close();

}

}

**Output:**



**d) Round Robin:**

**Code:**

import java.util.Scanner;

public class RoundRobin {

public static void main(String args[]) {

int n, qt;

int[] bt, wt, tat, rem\_bt, art, ct;

float awt = 0, atat = 0;

Scanner s = new Scanner(System.in);

System.out.print("Enter the number of processes (maximum 10) = ");

n = s.nextInt();

bt = new int[n];

wt = new int[n];

tat = new int[n];

rem\_bt = new int[n];

art = new int[n];

ct = new int[n];

System.out.print("Enter the burst time of the processes\n");

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

System.out.print("P" + (i + 1) + " = ");

bt[i] = s.nextInt();

rem\_bt[i] = bt[i];

}

System.out.print("Enter the arrival time of the processes\n");

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

System.out.print("P" + (i + 1) + " = ");

art[i] = s.nextInt();

}

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

qt = s.nextInt();

int t = 0;

boolean done;

do {

done = true;

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

if (rem\_bt[i] > 0 && art[i] <= t) {

done = false;

if (rem\_bt[i] > qt) {

t += qt;

rem\_bt[i] -= qt;

} else {

t += rem\_bt[i];

rem\_bt[i] = 0;

ct[i] = t;

tat[i] = ct[i] - art[i];

}

}

}

} while (!done);

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

tat[i] = ct[i] - art[i];

wt[i] = tat[i] - bt[i];

awt += wt[i];

atat += tat[i];

}

awt /= n;

atat /= n;

System.out.print("--------------------------------------------------------------------------------\n");

System.out.print("Process\tBurst Time\tArrival Time\tCompletion Time\tTurnaround Time\tWaiting Time\n");

System.out.print("--------------------------------------------------------------------------------\n");

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

System.out.print("P" + (i + 1) + "\t" + bt[i] + "\t\t" + art[i] + "\t\t" + ct[i] + "\t\t" + tat[i] + "\t\t" + wt[i] + "\n");

}

System.out.println("\nAverage waiting time = " + awt);

System.out.println("Average turnaround time = " + atat);

s.close();

}

}

**Output:**

