**FCFS.java:-->**

import java.util.\*;

public class FCFS

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.println("enter no of process: ");

int n = sc.nextInt();

int pid[] = new int[n]; // process ids

int ar[] = new int[n]; // arrival times

int bt[] = new int[n]; // burst or execution times

int ct[] = new int[n]; // completion times

int ta[] = new int[n]; // turn around times

int wt[] = new int[n]; // waiting times

int temp;

float avgwt=0,avgta=0;

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

{

System.out.println("enter process " + (i+1) + " arrival time: ");

ar[i] = sc.nextInt();

System.out.println("enter process " + (i+1) + " brust time: ");

bt[i] = sc.nextInt();

pid[i] = i+1;

}

//sorting according to arrival times

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

{

for(int j=0; j < n-(i+1) ; j++)

{

if( ar[j] > ar[j+1] )

{

temp = ar[j];

ar[j] = ar[j+1];

ar[j+1] = temp;

temp = bt[j];

bt[j] = bt[j+1];

bt[j+1] = temp;

temp = pid[j];

pid[j] = pid[j+1];

pid[j+1] = temp;

}

}

}

// finding completion times

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

{

if( i == 0)

{

ct[i] = ar[i] + bt[i];

}

else

{

if( ar[i] > ct[i-1])

{

ct[i] = ar[i] + bt[i];

}

else

ct[i] = ct[i-1] + bt[i];

}

ta[i] = ct[i] - ar[i] ; // turnaround time= completion time- arrival time

wt[i] = ta[i] - bt[i] ; // waiting time= turnaround time- burst time

avgwt += wt[i] ; // total waiting time

avgta += ta[i] ; // total turnaround time

}

System.out.println("\npid arrival brust complete turn waiting");

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

{

System.out.println(pid[i] + " \t " + ar[i] + "\t" + bt[i] + "\t" + ct[i] + "\t" + ta[i] + "\t" + wt[i] ) ;

}

sc.close();

System.out.println("\naverage waiting time: "+ (avgwt/n)); // printing average waiting time.

System.out.println("average turnaround time:"+(avgta/n)); // printing average turnaround time.

}

}

**OUTPUT:-->**

student@student-OptiPlex-3020:~$ javac FCFS.java

student@student-OptiPlex-3020:~$ java FCFS

enter no of process:

5

enter process 1 arrival time:

2

enter process 1 brust time:

3

enter process 2 arrival time:

5

enter process 2 brust time:

6

enter process 3 arrival time:

8

enter process 3 brust time:

1

enter process 4 arrival time:

4

enter process 4 brust time:

6

enter process 5 arrival time:

5

enter process 5 brust time:

8

pid arrival brust complete turn waiting

1 2 3 5 3 0

4 4 6 11 7 1

2 5 6 17 12 6

5 5 8 25 20 12

3 8 1 26 18 17

average waiting time: 7.2

average turnaround time:12.0

student@student-OptiPlex-3020:~$

**GFG.java:-->**

// Java program for implementation of RR scheduling

public class GFG

{

// Method to find the waiting time for all

// processes

static void findWaitingTime(int processes[], int n,

int bt[], int wt[], int quantum)

{

// Make a copy of burst times bt[] to store remaining

// burst times.

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

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

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

int t = 0; // Current time

// Keep traversing processes in round robin manner

// until all of them are not done.

while(true)

{

boolean done = true;

// Traverse all processes one by one repeatedly

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

{

// If burst time of a process is greater than 0

// then only need to process further

if (rem\_bt[i] > 0)

{

done = false; // There is a pending process

if (rem\_bt[i] > quantum)

{

// Increase the value of t i.e. shows

// how much time a process has been processed

t += quantum;

// Decrease the burst\_time of current process

// by quantum

rem\_bt[i] -= quantum;

}

// If burst time is smaller than or equal to

// quantum. Last cycle for this process

else

{

// Increase the value of t i.e. shows

// how much time a process has been processed

t = t + rem\_bt[i];

// Waiting time is current time minus time

// used by this process

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

// As the process gets fully executed

// make its remaining burst time = 0

rem\_bt[i] = 0;

}

}

}

// If all processes are done

if (done == true)

break;

}

}

// Method to calculate turn around time

static void findTurnAroundTime(int processes[], int n,

int bt[], int wt[], int tat[])

{

// calculating turnaround time by adding

// bt[i] + wt[i]

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

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

}

// Method to calculate average time

static void findavgTime(int processes[], int n, int bt[],

int quantum)

{

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

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

// Function to find waiting time of all processes

findWaitingTime(processes, n, bt, wt, quantum);

// Function to find turn around time for all processes

findTurnAroundTime(processes, n, bt, wt, tat);

// Display processes along with all details

System.out.println("Processes " + " Burst time " +

" Waiting time " + " Turn around time");

// Calculate total waiting time and total 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(" " + (i+1) + "\t\t" + bt[i] +"\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);

}

// Driver Method

public static void main(String[] args)

{

// process id's

int processes[] = { 1, 2, 3};

int n = processes.length;

// Burst time of all processes

int burst\_time[] = {10, 5, 8};

// Time quantum

int quantum = 2;

findavgTime(processes, n, burst\_time, quantum);

}

}

**OUTPUT:-->**

student@student-OptiPlex-3020:~$ javac GFG.java

student@student-OptiPlex-3020:~$ java GFG

Processes Burst time Waiting time Turn around time

1 10 13 23

2 5 10 15

3 8 13 21

Average waiting time = 12.0

Average turn around time = 19.666666

[student@student-OptiPlex-3020](mailto:student@student-OptiPlex-3020):~$

Priority.java -->

import java.util.Scanner;

public class Priority{

public static void main(String args[]) {

Scanner s = new Scanner(System.in);

int x,n,p[],pp[],bt[],w[],t[],awt,atat,i;

p = new int[10];

pp = new int[10];

bt = new int[10];

w = new int[10];

t = new int[10];

//n is number of process

//p is process

//pp is process priority

//bt is process burst time

//w is wait time

// t is turnaround time

//awt is average waiting time

//atat is average turnaround time

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

n = s.nextInt();

System.out.print("\n\t Enter burst time : time priorities \n");

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

{

System.out.print("\nProcess["+(i+1)+"]:");

bt[i] = s.nextInt();

pp[i] = s.nextInt();

p[i]=i+1;

}

//sorting on the basis of priority

for(i=0;i<n-1;i++)

{

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

{

if(pp[i]<pp[j])

{

x=pp[i];

pp[i]=pp[j];

pp[j]=x;

x=bt[i];

bt[i]=bt[j];

bt[j]=x;

x=p[i];

p[i]=p[j];

p[j]=x;

}

}

}

w[0]=0;

awt=0;

t[0]=bt[0];

atat=t[0];

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

{

w[i]=t[i-1];

awt+=w[i];

t[i]=w[i]+bt[i];

atat+=t[i];

}

//Displaying the process

System.out.print("\n\nProcess \t Burst Time \t Wait Time \t Turn Around Time Priority \n");

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

System.out.print("\n "+p[i]+"\t\t "+bt[i]+"\t\t "+w[i]+"\t\t "+t[i]+"\t\t "+pp[i]+"\n");

awt/=n;

atat/=n;

System.out.print("\n Average Wait Time : "+awt);

System.out.print("\n Average Turn Around Time : "+atat);

}

}

OUTPUT -->

student@student-OptiPlex-3020:~$ javac Priority.java

student@student-OptiPlex-3020:~$ java Priority

Enter the number of process : 3

Enter burst time : time priorities

Process[1]:10

3

Process[2]:5

1

Process[3]:9

2

Process Burst Time Wait Time Turn Around Time Priority

1 10 0 10 3

3 9 10 19 2

2 5 19 24 1

Average Wait Time : 9

Average Turn Around Time : 17student@student-OptiPlex-3020:~$

**my.java:-->**

// Java program to implement Shortest Remaining First

class Process

{

int pid; // Process ID

int bt; // Burst Time

int art; // Arrival Time

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

{

this.pid = pid;

this.bt = bt;

this.art = art;

}

}

public class my

{

// Method to find the waiting time for all

// processes

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

int wt[])

{

int rt[] = new int[n];

// Copy the burst time into rt[]

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;

// Process until all processes gets

// completed

while (complete != n) {

// Find process with minimum

// remaining time among the

// processes that arrives till the

// current time`

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;

}

// Reduce remaining time by one

rt[shortest]--;

// Update minimum

minm = rt[shortest];

if (minm == 0)

minm = Integer.MAX\_VALUE;

// If a process gets completely

// executed

if (rt[shortest] == 0) {

// Increment complete

complete++;

// Find finish time of current

// process

finish\_time = t + 1;

// Calculate waiting time

wt[shortest] = finish\_time -

proc[shortest].bt -

proc[shortest].art;

if (wt[shortest] < 0)

wt[shortest] = 0;

}

// Increment time

t++;

}

}

// Method to calculate turn around time

static void findTurnAroundTime(Process proc[], int n,

int wt[], int tat[])

{

// calculating turnaround time by adding

// bt[i] + wt[i]

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

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

}

// Method to calculate average time

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

{

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

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

// Function to find waiting time of all

// processes

findWaitingTime(proc, n, wt);

// Function to find turn around time for

// all processes

findTurnAroundTime(proc, n, wt, tat);

// Display processes along with all

// details

System.out.println("Processes " +

" Burst time " +

" Waiting time " +

" Turn around time");

// Calculate total waiting time and

// total turnaround 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);

}

// Driver Method

public static void main(String[] args)

{

Process proc[] = { new Process(1, 6, 1),

new Process(2, 8, 1),

new Process(3, 7, 2),

new Process(4, 3, 3)};

findavgTime(proc, proc.length);

}

}

**OUTPUT:-->**

student@student-OptiPlex-3020:~$ javac my.java

student@student-OptiPlex-3020:~$ java my

Processes Burst time Waiting time Turn around time

1 6 3 9

2 8 16 24

3 7 8 15

4 3 0 3

Average waiting time = 6.75

Average turn around time = 12.75

student@student-OptiPlex-3020:~$