

FCFS Scheduling Algorithm :

```
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 burst 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 :

The screenshot shows a Java IDE with a file named 'Main.java' and an 'Output' window. The code implements a process scheduling algorithm where users input arrival and burst times for five processes. The program then sorts these processes by arrival time and calculates various metrics including completion time, turnaround time, and average waiting/turnaround times.

Main.java Source Code:

```

1 import java.util.*;
2
3 public class FCFS {
4     public static void main(String args[])
5     {
6         Scanner sc = new Scanner(System.in);
7         System.out.println("enter no of process: ");
8         int n = sc.nextInt();
9         int pid[] = new int[n]; // process ids
10        int ar[] = new int[n]; // arrival times
11        int bt[] = new int[n]; // burst or execution times
12        int ct[] = new int[n]; // completion times
13        int ta[] = new int[n]; // turn around times
14        int wt[] = new int[n]; // waiting times
15        int temp;
16        float avgwt=0,avgta=0;
17
18        for(int i = 0; i < n; i++)
19        {
20            System.out.println("enter process " + (i+1) + " arrival time: ");
21            ar[i] = sc.nextInt();
22            System.out.println("enter process " + (i+1) + " burst time: ");
23            bt[i] = sc.nextInt();
24            pid[i] = i+1;
25        }
26
27        //sorting according to arrival times
28        for(int i = 0 ; i < n; i++)
29        {
30            for(int j=0; j < n-(i+1) ; j++)
31            {

```

Output:

```

java -cp /tmp/U19cP20p08 FCFS
enter no of process:
5
enter process 1 arrival time:
1
enter process 1 burst time:
5
enter process 2 arrival time:
3
enter process 2 burst time: 2
enter process 3 arrival time:
4
enter process 3 burst time:
6
enter process 4 arrival time:
4
enter process 4 burst time:
2
enter process 5 arrival time:
3
enter process 5 burst time:
2
pid arrival burst complete turn waiting
1 1 5 6 5 0
2 3 2 8 5 3
5 3 2 10 7 5
3 4 6 16 12 6
4 4 2 18 14 12
average waiting time: 5.2
average turnaround time:8.6

```

Priority Scheduling Algorithm :

```
import java.util.Scanner;

public class PRIO {

    public static void main(String args[]) {

        Scanner s = new Scanner(System.in);

        int ct[],a[],x,n,p[],pp[],bt[],w[],t[],i,k=0;

        float atat,awt;

        p = new int[10];
        pp = new int[10];
        bt = new int[10];
        w = new int[10];
        t = new int[10];
        a= new int[10];
        ct=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 : aarival time\n");

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

```

        System.out.print("\nProcess["+(i+1)+"]:"");
        bt[i] = s.nextInt();
        pp[i] = s.nextInt();
        a[i]=s.nextInt();
        p[i]=i+1;
    }
//SORT ON THE BASIS OF ARRIVAL TIME AND PRIORITY
for(i=0;i<n-1;i++)
{

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

        if(a[i]>=a[j] || 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;
            x=a[i];
            a[i]=a[j];
            a[j]=x;
        }

    }

}

//sorting on the basis of priority

```

```

for(i=1;i<=n;i++)
{
    if(i==1)
    { k=bt[0];

        ct[0]=k;}
    else{
        k=bt[i-1]+k;
        ct[i-1]=k;}
    for(int j=i+1;j<=n;j++)
    {

        if(pp[i]<pp[j] && a[j]<=k)
        {
            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;
            x=a[i];
            a[i]=a[j];
            a[j]=x;
        }
    }
}

w[0]=0;
awt=0;
t[0]=bt[0];
atat=t[0];

```

```

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

    t[i]=ct[i]-a[i];

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

    awt+=w[i];

    atat+=t[i];

}

//priority logic

//Displaying the process

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

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

    System.out.print("\n " +p[i]+" \t\t " +a[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 :

```

Main.java
23 //awt is average waiting time
24 //atat is average turnaround time
25
26
27 System.out.print("Enter the number of process : ");
28 n = s.nextInt();
29 System.out.print("\n\t Enter burst time : time priorities : arrival time\n");
30
31 for(i=0;i<n;i++)
32 {
33     System.out.print("\nProcess["+(i+1)+"] :");
34     bt[i] = s.nextInt();
35     pp[i] = s.nextInt();
36     a[i]=s.nextInt();
37     p[i]=i+1;
38 }
39 //SORT ON THE BASIS OF ARRIVAL TIME AND PRIORITY
40 for(i=0;i<n-1;i++)
41 {
42
43     for(int j=i+1;j<n;j++)
44     {
45
46         if(a[i]>=a[j] || pp[i]>pp[j])
47         {
48             x=pp[i];
49             pp[i]=pp[j];
50             pp[j]=x;
51             x=bt[i];
52             bt[i]=bt[j];
53             bt[j]=x;

```

```

Output
java -cp /tmp/r2bI4p1sx0 PRIO
Enter the number of process : 5
Enter burst time : time priorities : arrival time
Process[1]:5 2 3
Process[2]:4 3 2
Process[3]:5 2 1
Process[4]:1 4 4
Process[5]:6 1 5
Process    Arrival Time    Burst Time    Wait Time    Turn Around Time    Priority
5          5              6              0              6              1
2          4              4              1              3              4
2          2              4              5              9              3
1          3              5              8              13             2
3          1              5              15             20              2
Average Wait Time : 6.0Average Turn Around Time : 10.2

```

SRTF Scheduling Algorithm :

```
import java.io.*;

public class SRTF {

    public static void main(String args[]) throws IOException
    {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        int n;

        System.out.println("Please enter the number of Processes: ");

        n = Integer.parseInt(br.readLine());

        int proc[][] = new int[n + 1][4]; //proc[][0] is the AT array,[][1] - RT,[][2] - WT,[][3] - TT

        for(int i = 1; i <= n; i++)
        {
            System.out.println("Please enter the Arrival Time for Process " + i + ": ");

            proc[i][0] = Integer.parseInt(br.readLine());

            System.out.println("Please enter the Burst Time for Process " + i + ": ");

            proc[i][1] = Integer.parseInt(br.readLine());
        }

        System.out.println();

        //Calculation of Total Time and Initialization of Time Chart array

        int total_time = 0;

        for(int i = 1; i <= n; i++)
        {
            total_time += proc[i][1];
        }

        int time_chart[] = new int[total_time];

        for(int i = 0; i < total_time; i++)
        {
            //Selection of shortest process which has arrived

            int sel_proc = 0;

            int min = 99999;

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



```

{
    if(proc[j][0] <= i)//Condition to check if Process has arrived
    {
        if(proc[j][1] < min && proc[j][1] != 0)
        {
            min = proc[j][1];
            sel_proc = j;
        }
    }
}

//Assign selected process to current time in the Chart
time_chart[i] = sel_proc;

//Decrement Remaining Time of selected process by 1 since it has been assigned the CPU for 1 unit of time
proc[sel_proc][1]--;

//WT and TT Calculation
for(int j = 1; j <= n; j++)
{
    if(proc[j][0] <= i)
    {
        if(proc[j][1] != 0)
        {
            proc[j][3]++;//If process has arrived and it has not already completed execution its TT is incremented by 1
            if(j != sel_proc)//If the process has not been currently assigned the CPU and has arrived its WT is
incremented by 1
                proc[j][2]++;
        }

        else if(j == sel_proc)//This is a special case in which the process has been assigned CPU and has completed
its execution
            proc[j][3]++;
    }
}

```

```

//Printing the Time Chart
if(i != 0)
{
    if(sel_proc != time_chart[i - 1])
        //If the CPU has been assigned to a different Process we need to print the current value of time and the
        name of
        //the new Process
        {
            System.out.print("--" + i + "--P" + sel_proc);
        }
}

else//If the current time is 0 i.e the printing has just started we need to print the name of the First selected
Process
    System.out.print(i + "--P" + sel_proc);

if(i == total_time - 1)//All the process names have been printed now we have to print the time at which
execution ends

    System.out.print("--" + (i + 1));

}

System.out.println();
System.out.println();

//Printing the WT and TT for each Process
System.out.println("P\t WT \t TT ");
for(int i = 1; i <= n; i++)
{
    System.out.printf("%d\t%2dms\t%2dms",i,proc[i][2],proc[i][3]);
    System.out.println();
}

System.out.println();

//Printing the average WT & TT

```

```

float WT = 0, TT = 0;

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

{

    WT += proc[i][2];

    TT += proc[i][3];

}

WT /= n;

TT /= n;

System.out.println("The Average WT is: " + WT + "ms");

System.out.println("The Average TT is: " + TT + "ms");

}

}

```

Output :

The screenshot shows a Java IDE with a file named 'Main.java'. The code implements the SRTF (Shortest Remaining Time First) scheduling algorithm. It prompts the user to enter the number of processes (5), then for each process, it asks for the arrival time and burst time. The code calculates the total time and initializes a time chart array. It then iterates through the processes, selecting the shortest process that has arrived at each step. The output window shows the execution results, including the order of process execution and the final average waiting time (WT) and average turn-around time (TT).

```

Main.java
1- import java.io.*;
2- public class SRTF {
3-     public static void main(String args[]) throws IOException
4-     {
5-         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
6-         int n;
7-         System.out.println("Please enter the number of Processes: ");
8-         n = Integer.parseInt(br.readLine());
9-         int proc[][] = new int[n + 1][4]; //proc[][0] is the AT array, [][1] - RT, [][2] -
            WT, [][3] - TT
10-         for(int i = 1; i <= n; i++)
11-         {
12-             System.out.println("Please enter the Arrival Time for Process " + i + ": ");
13-             proc[i][0] = Integer.parseInt(br.readLine());
14-             System.out.println("Please enter the Burst Time for Process " + i + ": ");
15-             proc[i][1] = Integer.parseInt(br.readLine());
16-         }
17-         System.out.println();
18-         //Calculation of Total Time and Initialization of Time Chart array
19-         int total_time = 0;
20-         for(int i = 1; i <= n; i++)
21-         {
22-             total_time += proc[i][1];
23-         }
24-         int time_chart[] = new int[total_time];
25-         for(int i = 0; i < total_time; i++)
26-         {
27-             //Selection of shortest process which has arrived
28-             int sel proc = 0;

```

```

Output
java -cp /tmp/A6kBZ2dVHF SRTF
Please enter the number of Processes: 5
Please enter the Arrival Time for Process 1: 5
Please enter the Burst Time for Process 1: 2
Please enter the Arrival Time for Process 2: 6
Please enter the Burst Time for Process 2: 4
Please enter the Arrival Time for Process 3: 3
Please enter the Burst Time for Process 3: 1
Please enter the Arrival Time for Process 4: 2
Please enter the Burst Time for Process 4: 7
Please enter the Arrival Time for Process 5: 5
Please enter the Burst Time for Process 5: 1
0--P0--2--P4--3--P3--4--P4--5--P5--6--P1--8--P2--12--P4--15
P   WT   TT
11ms   3ms 2ms   6ms
30ms   1ms
4   8ms 13ms
5   0ms 1ms

The Average WT is: 2.2ms
The Average TT is: 4.8ms

```

Round Robin Scheduling Algorithm :

```
import java.util.Scanner;

public class ROBIN {

    public static void main(String args[]) {

        Scanner s = new Scanner(System.in);

        int wtime[],btime[],rtime[],num,quantum,total;

        wtime = new int[10];
        btime = new int[10];
        rtime = new int[10];

        System.out.print("Enter number of processes(MAX 10): ");
        num = s.nextInt();

        System.out.print("Enter burst time");

        for(int i=0;i<num;i++) { System.out.print("\nP["+(i+1)+"] : "); btime[i] = s.nextInt(); rtime[i] = btime[i];
        wtime[i]=0; } System.out.print("\n\nEnter quantum: "); quantum = s.nextInt(); int rp = num; int i=0; int time=0;
        System.out.print("0"); wtime[0]=0; while(rp!=0) { if(rtime[i]>quantum)
        {
            rtime[i]=rtime[i]-quantum;

            System.out.print(" | P["+(i+1)+"] | ");

            time+=quantum;

            System.out.print(time);

        }
        else if(rtime[i]<=quantum && rtime[i]>0)
        {time+=rtime[i];
            rtime[i]=rtime[i]-rtime[i];

            System.out.print(" | P["+(i+1)+"] | ");

            rp--;

            System.out.print(time);

        }
    }
}
```

```

i++;

if(i==num)

{

i=0;

}

}

}

}

```

Output :

Main.java

Run

Output

Clear

```

13
14 System.out.print("Enter number of processes(MAX 10): ");
15 num = s.nextInt();
16 System.out.print("Enter burst time");
17 for(int i=0;i<num;i++) { System.out.print("\nP["+(i+1)+"]: "); btime[i] = s.nextInt();
    rtime[i] = btime[i]; wtime[i]=0; } System.out.print("\nEnter quantum: "); quantum
    = s.nextInt(); int rp = num; int i=0; int time=0; System.out.print("0"); wtime[0]=0;
    while(rp!=0) { if(rtime[i]>quantum)
18+ {
19     rtime[i]=rtime[i]-quantum;
20     System.out.print(" | P["+(i+1)+"] | ");
21     time+=quantum;
22     System.out.print(time);
23
24 }
25 else if(rtime[i]<=quantum && rtime[i]>0)
26 {time+=rtime[i];
27     rtime[i]=rtime[i]-rtime[i];
28     System.out.print(" | P["+(i+1)+"] | ");
29     rp--;
30     System.out.print(time);
31 }
32
33 i++;
34 if(i==num)
35+ {
36     i=0;
37 }
38
39 }
40

```

```

^ java -cp /tmp/A6kBZ2dVHF ROBIN
Enter number of processes(MAX 10): 5
Enter burst timeP[1]: 6
P[2]: 34
P[3]: 23
P[4]: 45
P[5]: 19
Enter quantum: 5
0| P[1] | 5 | P[2] | 10 | P[3] | 15 | P[4] | 20| P[5] | 25| P[1] | 26 | P[2] | 31 | P[3] | 36 |
P[4] | 41| P[5] | 46 | P[2] | 51 | P[3] | 56 | P[4] | 61 | P[5] | 66| P[2] | 71 | P[3] | 76 |
P[4] | 81| P[5] | 85 | P[2] | 90 | P[3] | 93 | P[4] | 98 | P[2] | 103 | P[4] | 108 | P[2] |
112 | P[4] | 117 | P[4] | 122 | P[4] | 127

```