8. Job Scheduling Algorithm :-

Aim :- Implement Job Scheduling algorithm
1.) FCFS.
2.) Shortest Job first
3.) Priority.
4.) Round Robin

Problem statement :- Write a Java program (using OOP features) to implement following scheduling algorithm FCFS, SJF (Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive)

Theory :-

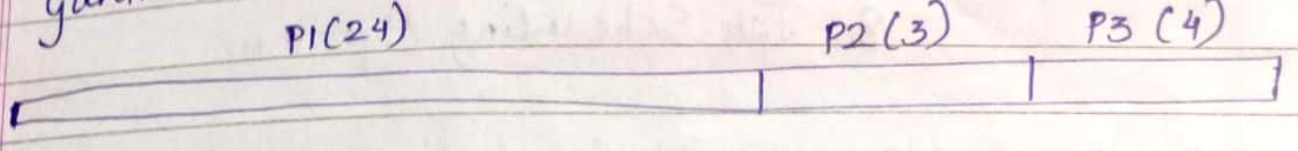
~~Roos~~ 1.) First Come first serve (FCFS)

This is the simplest CPU scheduling algorithm. The Process that request the CPU first, is the one which it is allocated first.

Example :-

Process	Duration	Order	Arrival time
P1	24	1	0
P2	3	2	0
P3	4	3	0

Gantt chart



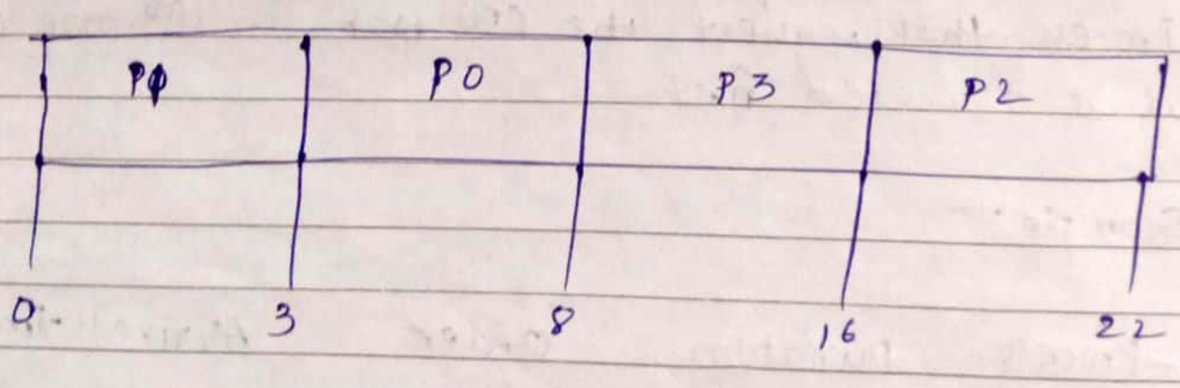
P1 waiting time : 0
P2 waiting time : 24
P3 waiting time : 27

The AWT :-
(0+24+27)/3 = 17

2) Shortest Job first :-

This algorithm associates with it the length of the next CPU burst

Process	Arrival time	Execute time	Service time
P0	0	5	0
P1	1	3	3
P2	2	8	8
P3	3	6	16



Shortest Remaining Time (SRT):-

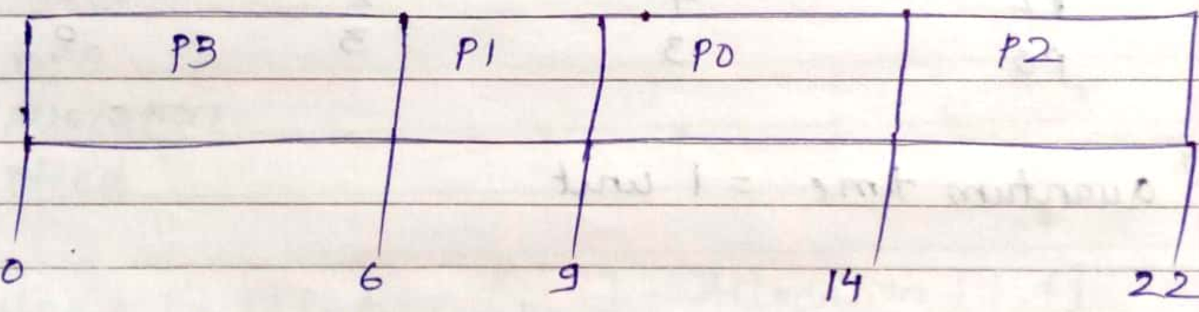
It is the preemptive version of the SJN algorithm

3) Priority Based Scheduling :-

- Priority scheduling is a non-preemptive algorithm and one of the most common algorithm.

- Each process is assigned a priority. Process with highest priority is to be executed first & so on.

Process	Arrival time	Execute time	Priority	Service Time
P0	0	5	1	9
P1	1	3	2	6
P2	2	8	1	14
P3	3	6	3	0



Wait time

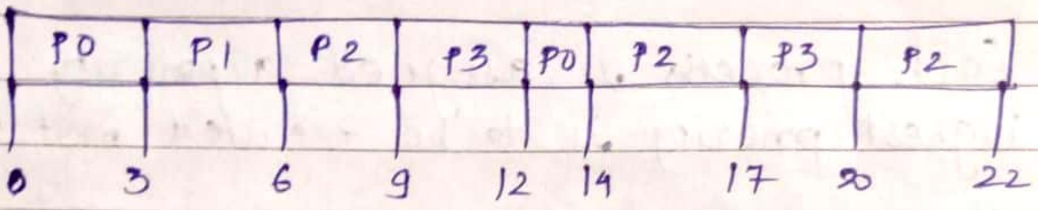
Process	Wait time : Service - Arrival time
P0	9 - 0 = 9
P1	6 - 1 = 5
P2	14 - 2 = 12
P3	0 - 0 = 0

Date

4) Round Robin Scheduling

It is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way.

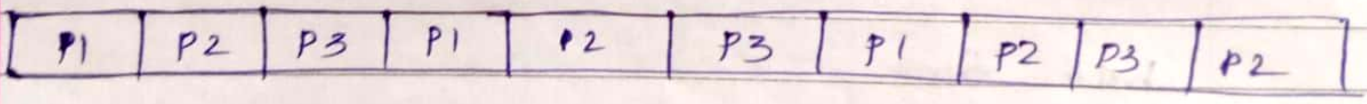
Quantum = 3



Example :-

Process	Duration	Order	Arrival time
P1	3	1	0
P2	4	2	0
P3	3	3	0

quantum time = 1 unit



P1 W.T = 4 ; AWT = (4+6+6)/3 = 5.33
P2 W.T = 6
P3 W.T = 6

Assignment No. 08

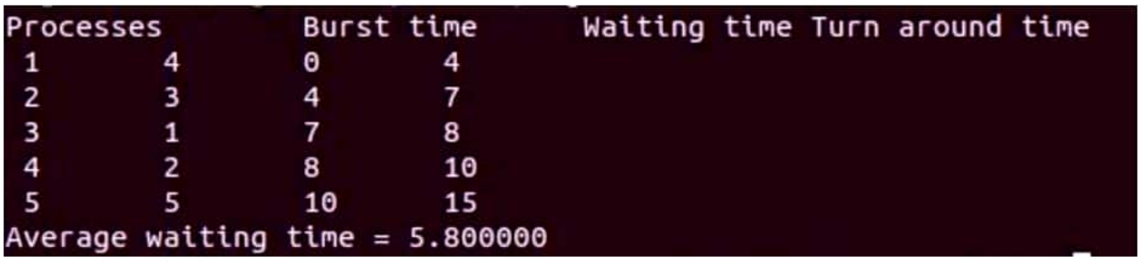
Problem Satement: Write a Java program (using OOP features) to implement following scheduling algorithms:
FCFS , SJF (Preemptive), Priority (Non - Preemptive) and Round Robin (Preemptive)

1. FCFS Program:

```
// Java program for implementation of FCFS
// scheduling import

java.text.ParseException; class

FCFS {


        // Function to find the waiting time for all
        // processes
        static void findWaitingTime(int processes[], int n,
                        int bt[], int wt[]) {
                // waiting time for first process is 0
                wt[0] = 0;

                // calculating waiting time for (int
                i = 1; i < n; i++) { wt[i] = bt[i - 1]
                + wt[i - 1]; }
        }

        // Function 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];
                }
        }

        //Function to calculate average time
        static void findavgTime(int processes[], int n, int bt[]) {
                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);
                //Function to find turn around time for all processes
        findTurnAroundTime(processes, n, bt, wt, tat);
                //Display processes along with all details
```

```
            System.out.printf("Processes \t Burst time \t Waiting" +" time Turn around time\n");

            // 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.printf(" %d
            ", (i + 1));
                    System.out.printf("     %d ", bt[i]);
                    System.out.printf("     %d", wt[i]);
                    System.out.printf("     %d\n", tat[i]);
            }
            float s = (float)total_wt /(float) n;
            int t = total_tat / n;
            System.out.printf("Average waiting time = %f", s);
            System.out.printf("\n");
            System.out.printf("Average turn around time = %d ", t);
    }

    // Driver code
    public static void main(String[] args) throws ParseException {
            //process id's int processes[] =
            {1, 2, 3,4,5}; int n =
            processes.length;

            //Burst time of all processes int

            burst_time[] = {4,3,1,2,5};

            findavgTime(processes, n, burst_time);


    }
}
```

FCFS OUTPUT:



```
Processes        Burst time      Waiting time Turn around time
 1       4       0       4
 2       3       4       7
 3       1       7       8
 4       2       8       10
 5       5       10      15
Average waiting time = 5.800000
```

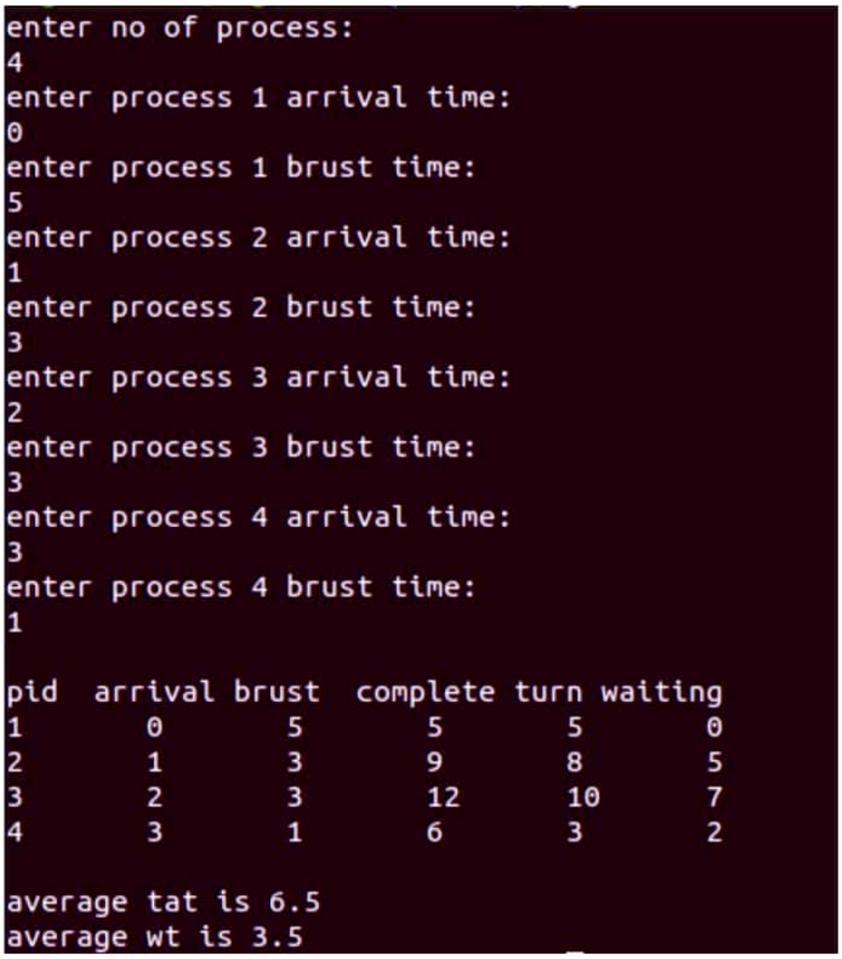
2. Shrtest Job First Program:

```
import java.util.*;


public class SJF { 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]; int at[] = new int[n]; //
        at means arrival time int bt[] = new int[n]; // bt means burst time int ct[] =
        new int[n]; // ct means complete time int ta[] = new int[n]; // ta means turn
        around time int wt[] = new int[n];  //wt means waiting time int f[] = new
        int[n];  // f means it is flag it checks process is completed or not int st=0,
        tot=0; float avgwt=0, avgta=0;


        for(int i=0;i<n;i++)
        {
            System.out.println ("enter process " + (i+1) + " arrival time:");
            at[i] = sc.nextInt();
            System.out.println ("enter process " + (i+1) + " brust
            time:"); bt[i] = sc.nextInt(); pid[i] = i+1; f[i] = 0;
        }


        boolean a = true;
        while(true)
        { int c=n, min=999; if (tot == n) // total no of process = completed process loop will
            be terminated break;
            for (int i=0; i<n; i++)
            {
                /*
                * If i'th process arrival time <= system time and its flag=0 and
burst<min
                * That process will be executed first
                 */ if ((at[i] <= st) && (f[i] == 0) &&
                (bt[i]<min))
                { min=bt[i]; c=i;
                }
            }
```

```
                /* If c==n means c value can not updated because no process arrival time<
system time so we increase the system time */
                if (c==n) st++;

                else
                {
                        ct[c]=st+bt[c];
                        st+=bt[c];
                        ta[c]=ct[c]-at[c];
                        wt[c]=ta[c]-
                        bt[c]; f[c]=1;
                        tot++;

                }
            }

            System.out.println("\npid  arrival brust  complete turn waiting");
            for(int i=0;i<n;i++)
            { avgwt+= wt[i];
                avgta+= ta[i];
                System.out.pri
                ntln(pid[i]+"\t
                "+at[i]+"\t"+bt
                [i]+"\t"+ct[i]+
                "\t"+ta[i]+"\
t"+wt[i]);
            }
            System.out.println ("\naverage tat is "+ (float)(avgta/n));
            System.out.println ("average wt is "+ (float)(avgwt/n));
            sc.close();
        }
}
```

SJF OUTPUT:

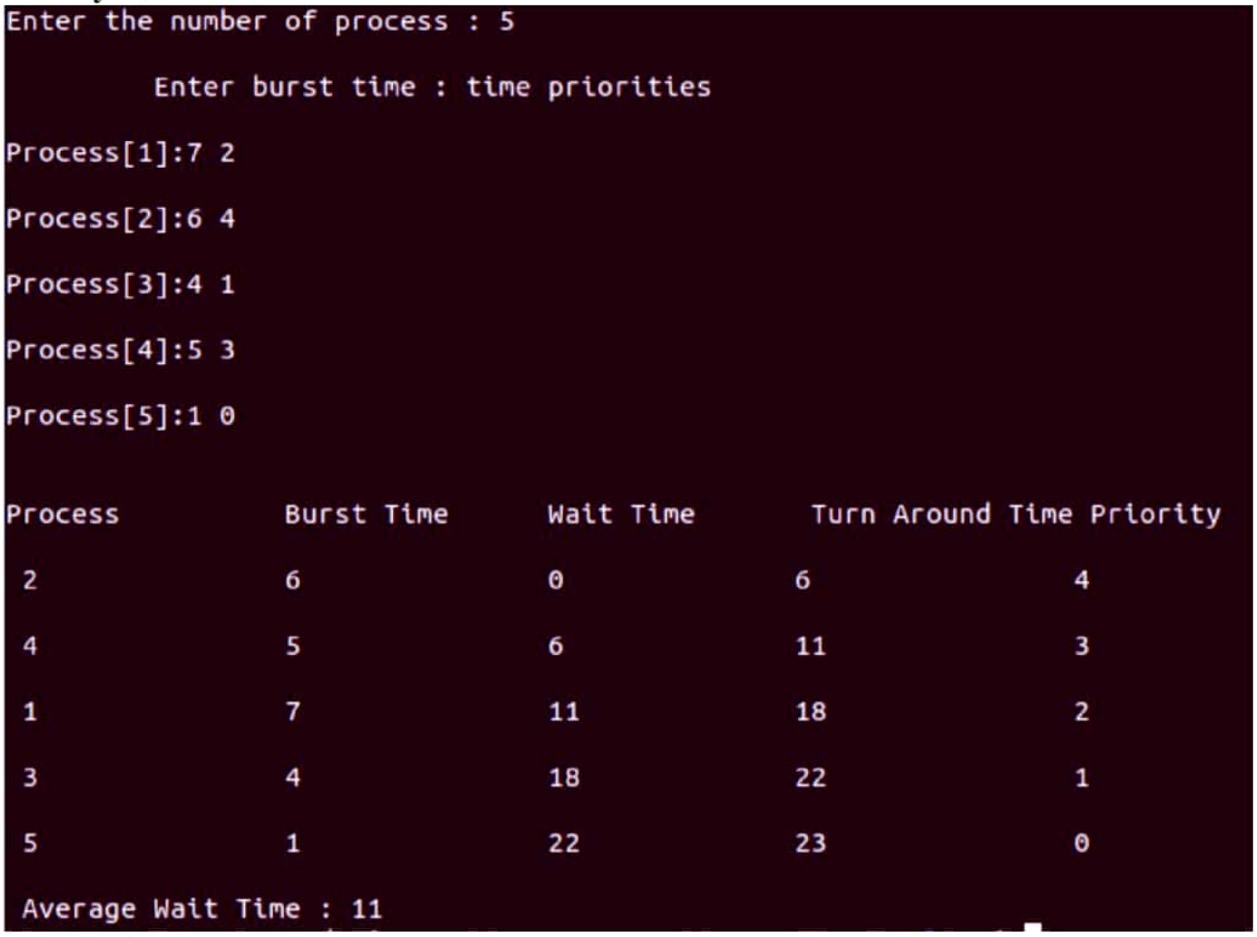


```
enter no of process:
4
enter process 1 arrival time:
0
enter process 1 brust time:
5
enter process 2 arrival time:
1
enter process 2 brust time:
3
enter process 3 arrival time:
2
enter process 3 brust time:
3
enter process 4 arrival time:
3
enter process 4 brust time:
1

pid  arrival brust  complete turn waiting
1       0       5       5       5       0
2       1       3       9       8       5
3       2       3       12      10      7
4       3       1       6       3       2

average tat is 6.5
average wt is 3.5
```

3. Priority Program:

```
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);
 }
}
```

Priority OUTPUT:



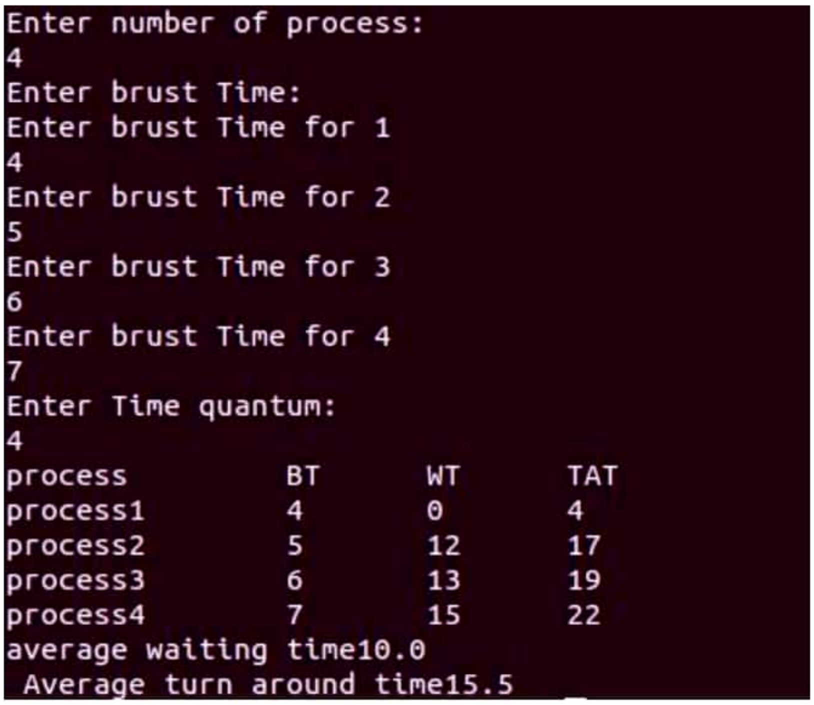
```
Enter the number of process : 5

         Enter burst time : time priorities

Process[1]:7 2

Process[2]:6 4

Process[3]:4 1

Process[4]:5 3

Process[5]:1 0

Process          Burst Time      Wait Time        Turn Around Time Priority

 2               6               0               6                4

 4               5               6               11               3

 1               7               11              18               2

 3               4               18              22               1

 5               1               22              23               0

 Average Wait Time : 11
```

4. Round Robin Program:

```
import java.io.*;
class RoundR {
public static void main(String args[])throws IOException
{
DataInputStream in=new DataInputStream(System.in);
int i,j,k,q,sum=0;
System.out.println("Enter number of process:");
int n=Integer.parseInt(in.readLine()); int
bt[]=new int[n]; int wt[]=new int[n]; int
tat[]=new int[n]; int a[]=new int[n];
System.out.println("Enter brust Time:");
for(i=0;i<n;i++)
{
System.out.println("Enter brust Time for "+(i+1));
bt[i]=Integer.parseInt(in.readLine());
}
System.out.println("Enter Time quantum:");
q=Integer.parseInt(in.readLine());
for(i=0;i<n;i++) a[i]=bt[i]; for(i=0;i<n;i++)
wt[i]=0; do {
for(i=0;i<n;i++)
{
```

```
if(bt[i]>q)
{
bt[i]-=q;
for(j=0;j<n;j++) {
if((j!=i)&&(bt[j]!=0))
wt[j]+=q; }
}
else {
for(j=0;j<n;j++) {
if((j!=i)&&(bt[j]!=0))
wt[j]+=bt[i];
}
bt[i]=0;
} } sum=0;
for(k=0;k<n;k++)
sum=sum+bt[k];
}
while(sum!=0);
for(i=0;i<n;i++)
tat[i]=wt[i]+a[i];
System.out.println("process\t\tBT\tWT\tTAT");
for(i=0;i<n;i++)
{
System.out.println("process"+(i+1)+"\t"+a[i]+"\t"+wt[i]+"\t"+tat[i]);
}
float avg_wt=0;
float avg_tat=0;
for(j=0;j<n;j++)
{
avg_wt+=wt[j];
}
for(j=0;j<n;j++)
{
avg_tat+=tat[j];
}
System.out.println("average waiting time"+(avg_wt/n)+"\n Average turn around
time"+(avg_tat/n));
}
}
```

Round Robin OUTPUT:



```
Enter number of process:
4
Enter brust Time:
Enter brust Time for 1
4
Enter brust Time for 2
5
Enter brust Time for 3
6
Enter brust Time for 4
7
Enter Time quantum:
4
process         BT      WT      TAT
process1        4       0       4
process2        5       12      17
process3        6       13      19
process4        7       15      22
average waiting time10.0
 Average turn around time15.5
```