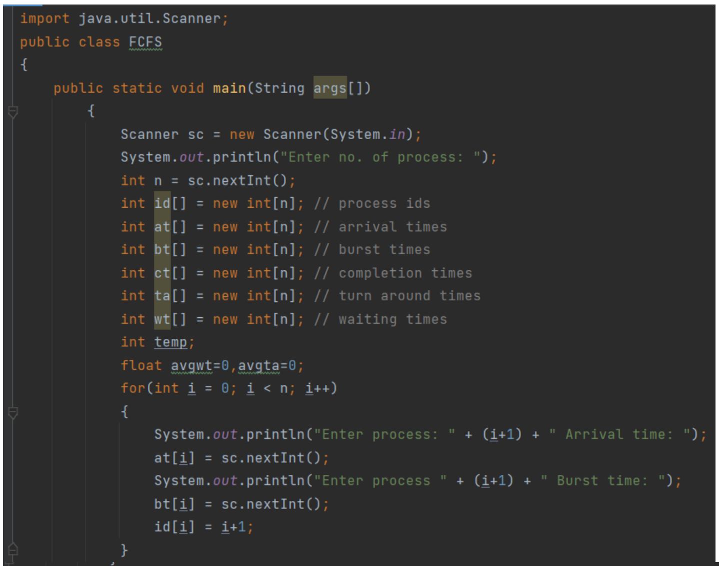
Operating Systems

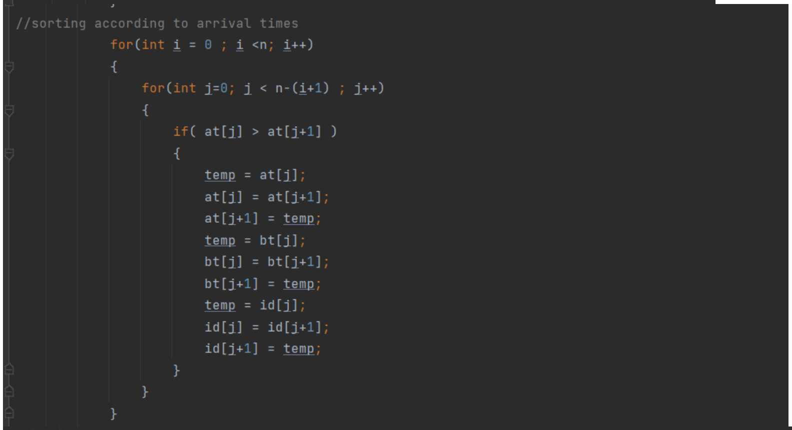
Assignment 3: Implementation of Scheduling Algorithms

1. First Come First Serve Algorithm

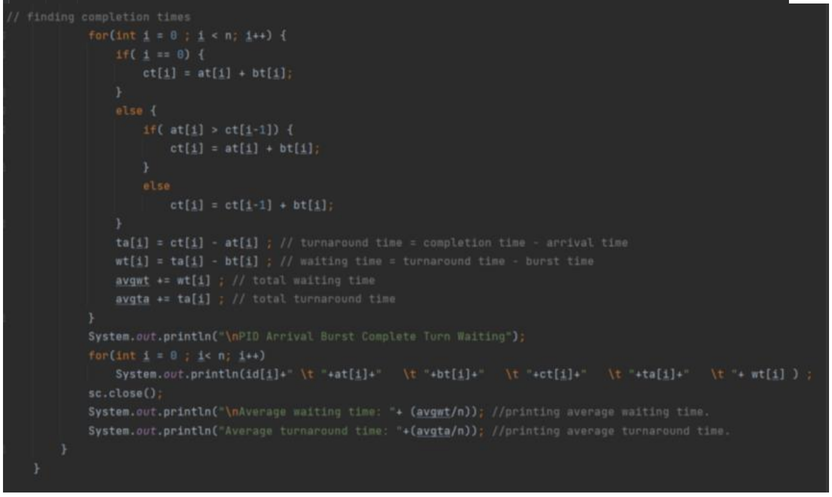
Code:



```java
import java.util.Scanner;
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 id[] = new int[n]; // process ids
            int at[] = new int[n]; // arrival times
            int bt[] = new int[n]; // burst 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: ");
                at[i] = sc.nextInt();
                System.out.println("Enter process " + (i+1) + " Burst time: ");
                bt[i] = sc.nextInt();
                id[i] = i+1;
            }
```

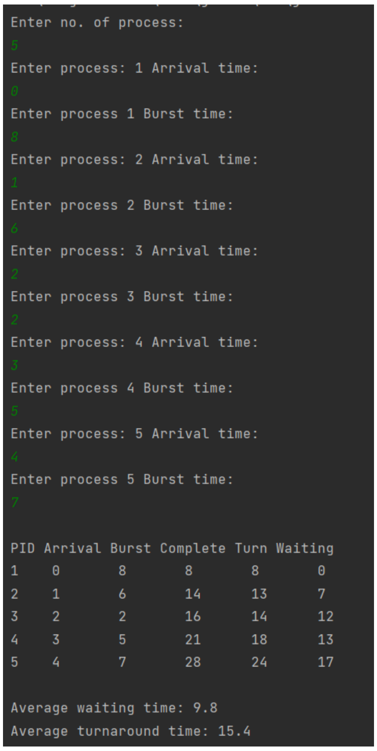


```java
//sorting according to arrival times
            for(int i = 0 ; i <n; i++)
            {
                for(int j=0; j < n-(i+1) ; j++)
                {
                    if( at[j] > at[j+1] )
                    {
                        temp = at[j];
                        at[j] = at[j+1];
                        at[j+1] = temp;
                        temp = bt[j];
                        bt[j] = bt[j+1];
                        bt[j+1] = temp;
                        temp = id[j];
                        id[j] = id[j+1];
                        id[j+1] = temp;
                    }
                }
            }
```



```java
// finding completion times
            for(int i = 0 ; i < n; i++) {
                if( i == 0) {
                    ct[i] = at[i] + bt[i];
                }
                else {
                    if( at[i] > ct[i-1]) {
                        ct[i] = at[i] + bt[i];
                    }
                    else
                        ct[i] = ct[i-1] + bt[i];
                }
                ta[i] = ct[i] - at[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(id[i]+" \t "+at[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.
        }
    }
```

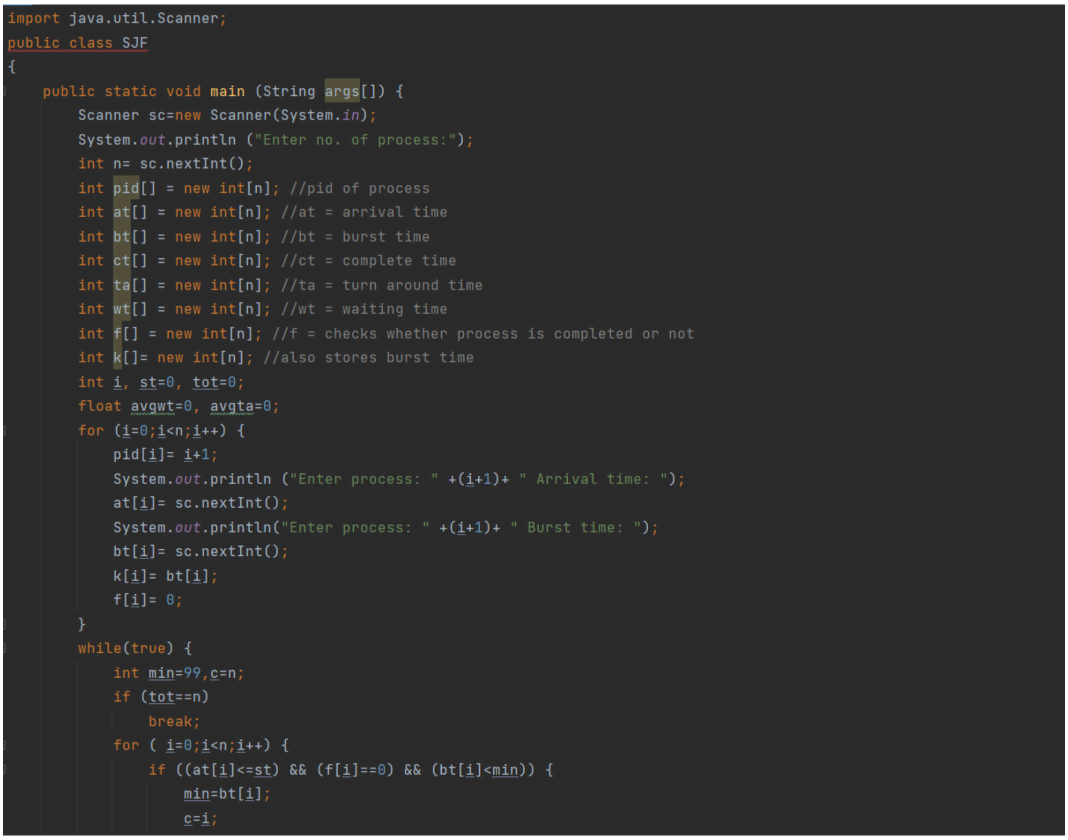
Output:



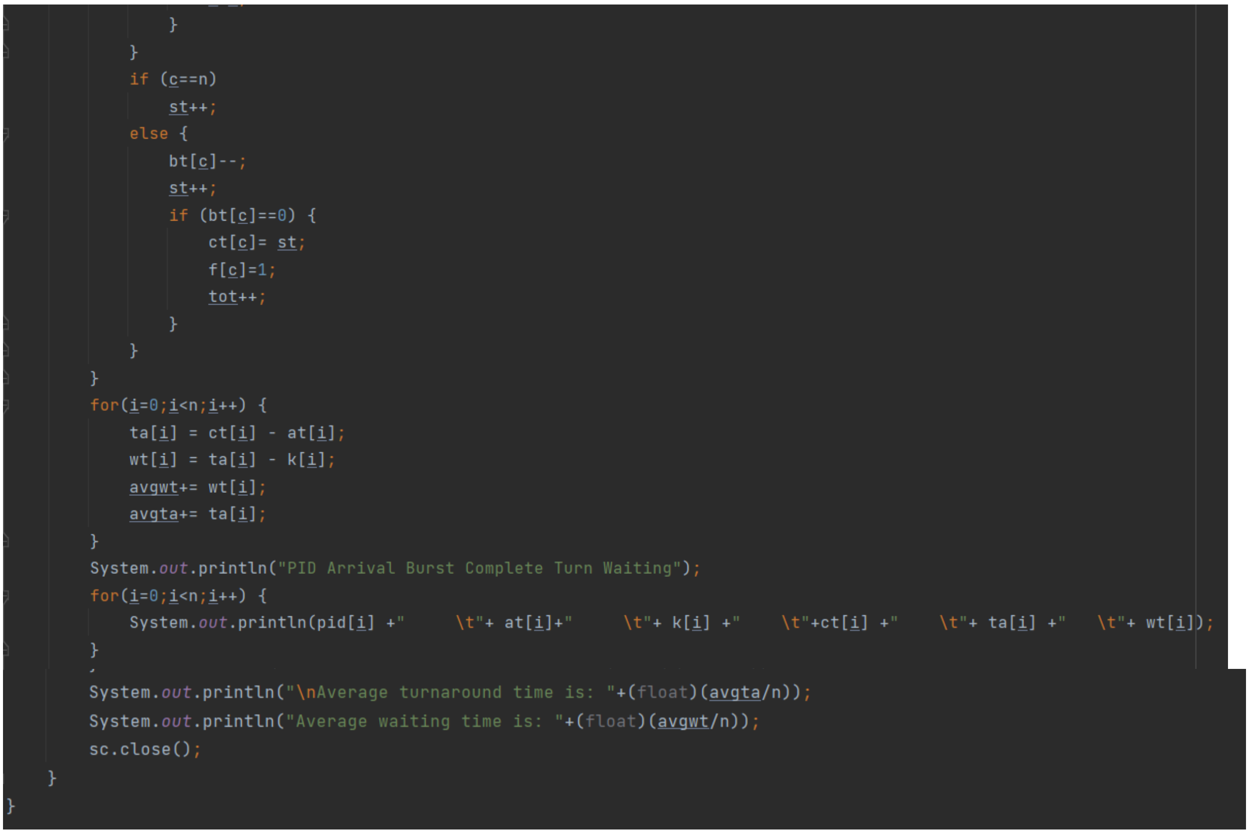
```
Enter no. of process:
5
Enter process: 1 Arrival time:
0
Enter process 1 Burst time:
8
Enter process: 2 Arrival time:
1
Enter process 2 Burst time:
6
Enter process: 3 Arrival time:
2
Enter process 3 Burst time:
2
Enter process: 4 Arrival time:
3
Enter process 4 Burst time:
5
Enter process: 5 Arrival time:
4
Enter process 5 Burst time:
7

PID Arrival Burst Complete Turn Waiting
1    0       8       8       8       0
2    1       6       14      13      7
3    2       2       16      14      12
4    3       5       21      18      13
5    4       7       28      24      17

Average waiting time: 9.8
Average turnaround time: 15.4
```

2. CPU Scheduling Using SJF

Code:

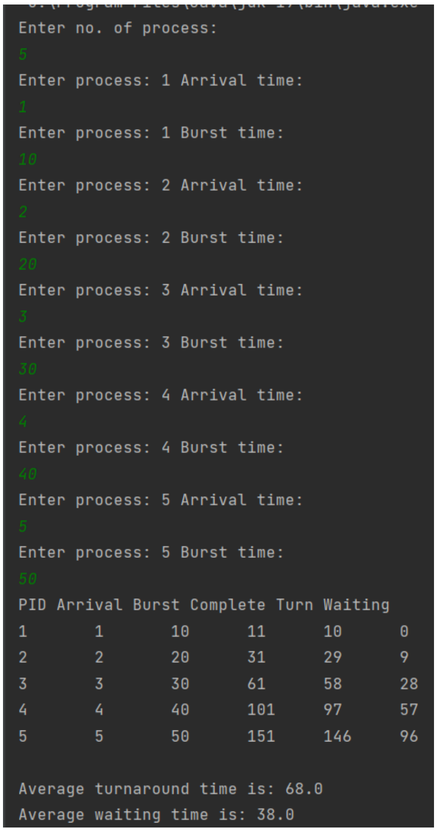


```java
import java.util.Scanner;
public class SJF
{
    public static void main (String args[]) {
        Scanner sc=new Scanner(System.in);
        System.out.println ("Enter no. of process:");
        int n= sc.nextInt();
        int pid[] = new int[n]; //pid of process
        int at[] = new int[n]; //at = arrival time
        int bt[] = new int[n]; //bt = burst time
        int ct[] = new int[n]; //ct = complete time
        int ta[] = new int[n]; //ta = turn around time
        int wt[] = new int[n]; //wt = waiting time
        int f[] = new int[n]; //f = checks whether process is completed or not
        int k[]= new int[n]; //also stores burst time
        int i, st=0, tot=0;
        float avgwt=0, avgta=0;
        for (i=0;i<n;i++) {
            pid[i]= i+1;
            System.out.println ("Enter process: " +(i+1)+ " Arrival time: ");
            at[i]= sc.nextInt();
            System.out.println("Enter process: " +(i+1)+ " Burst time: ");
            bt[i]= sc.nextInt();
            k[i]= bt[i];
            f[i]= 0;
        }
        while(true) {
            int min=99,c=n;
            if (tot==n)
                break;
            for ( i=0;i<n;i++) {
                if ((at[i]<=st) && (f[i]==0) && (bt[i]<min)) {
                    min=bt[i];
                    c=i;
```



```
                }
            }
            if (c==n)
                st++;
            else {
                bt[c]--;
                st++;
                if (bt[c]==0) {
                    ct[c]= st;
                    f[c]=1;
                    tot++;
                }
            }
        }
        for(i=0;i<n;i++) {
            ta[i] = ct[i] - at[i];
            wt[i] = ta[i] - k[i];
            avgwt+= wt[i];
            avgta+= ta[i];
        }
        System.out.println("PID Arrival Burst Complete Turn Waiting");
        for(i=0;i<n;i++) {
            System.out.println(pid[i] +"     \t"+ at[i]+"     \t"+ k[i] +"     \t"+ct[i] +"     \t"+ ta[i] +"     \t"+ wt[i]);
        }
        System.out.println("\nAverage turnaround time is: "+(float)(avgta/n));
        System.out.println("Average waiting time is: "+(float)(avgwt/n));
        sc.close();
    }
}
```

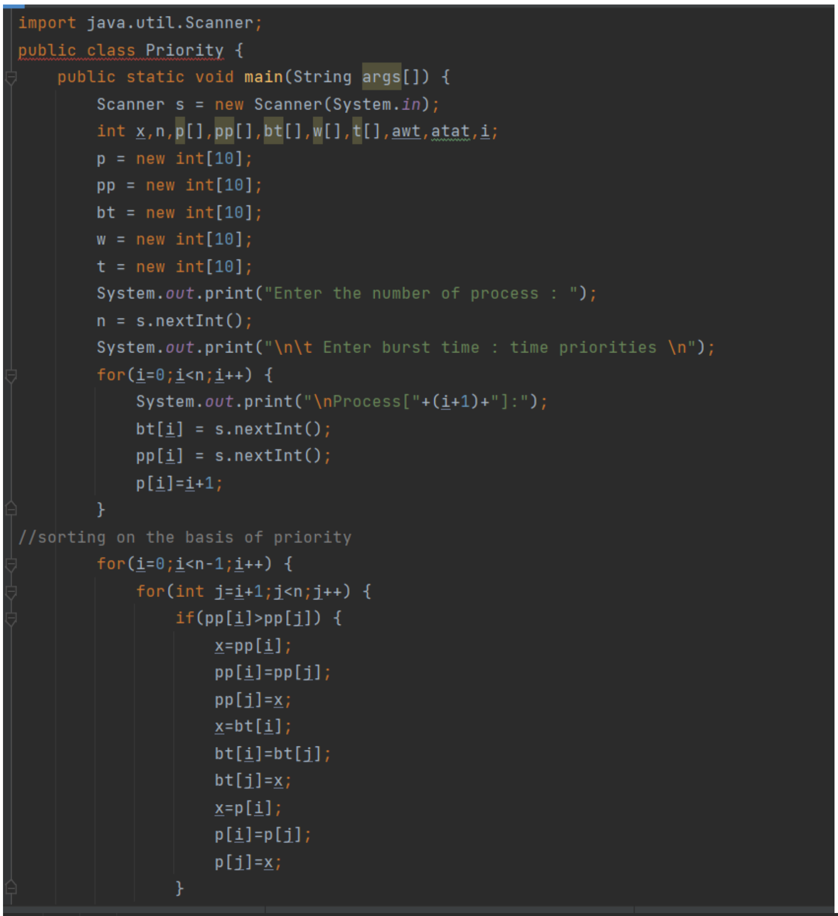
Output:



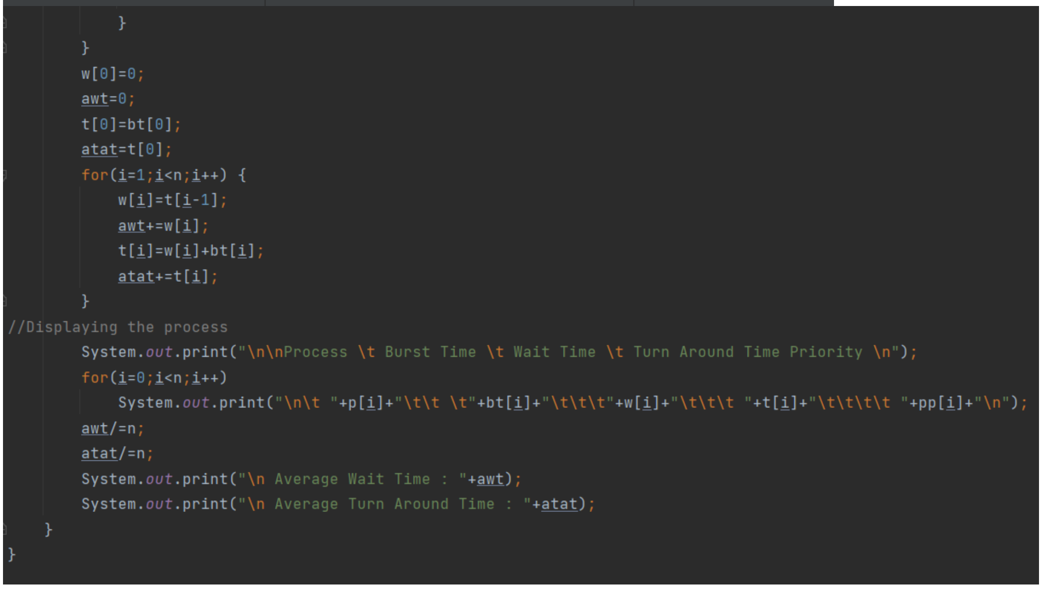
```
Enter no. of process:
5
Enter process: 1 Arrival time:
1
Enter process: 1 Burst time:
10
Enter process: 2 Arrival time:
2
Enter process: 2 Burst time:
20
Enter process: 3 Arrival time:
3
Enter process: 3 Burst time:
30
Enter process: 4 Arrival time:
4
Enter process: 4 Burst time:
40
Enter process: 5 Arrival time:
5
Enter process: 5 Burst time:
50
PID Arrival Burst Complete Turn Waiting
1       1       10      11      10      0
2       2       20      31      29      9
3       3       30      61      58      28
4       4       40      101     97      57
5       5       50      151     146     96

Average turnaround time is: 68.0
Average waiting time is: 38.0
```

3. CPU Scheduling Using Priority

Code:

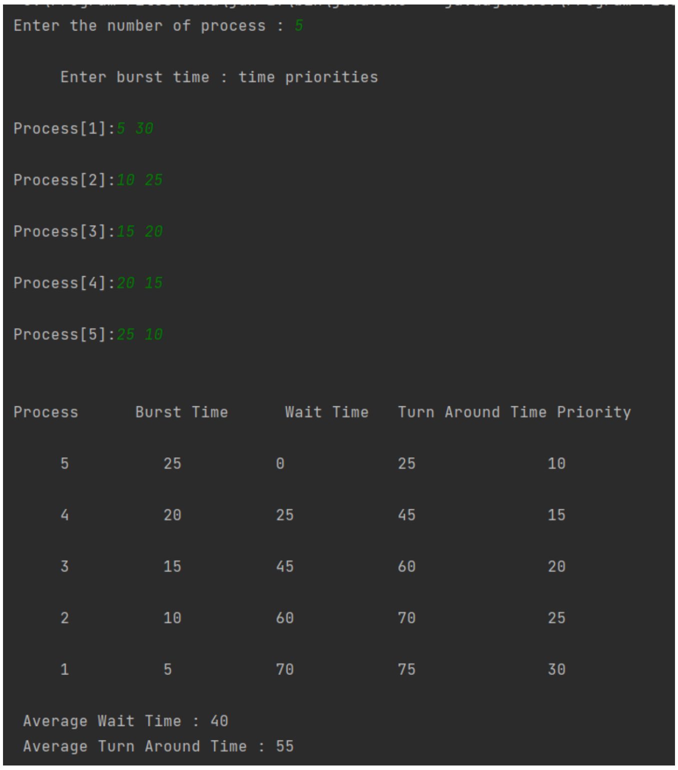


```
import java.util.Scanner;
public class Priority {
    public static void main(String args[]) {
        Scanner s = new Scanner(System.in);
        int x,n,p[],pp[],bt[],w[],t[],awt,atat,i;
        p = new int[10];
        pp = new int[10];
        bt = new int[10];
        w = new int[10];
        t = new int[10];
        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\t "+p[i]+"\t\t \t"+bt[i]+"\t\t\t"+w[i]+"\t\t\t "+t[i]+"\t\t\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);
    }
}
```

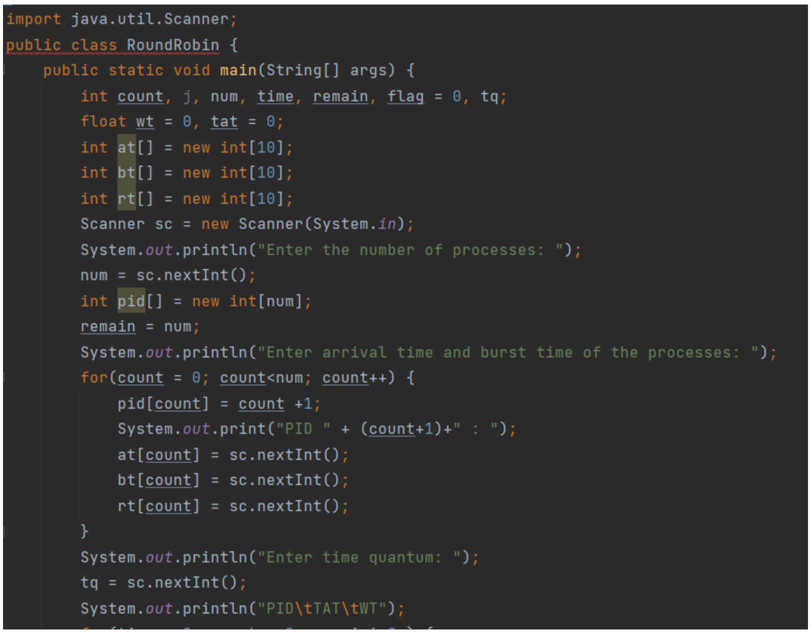
Output:



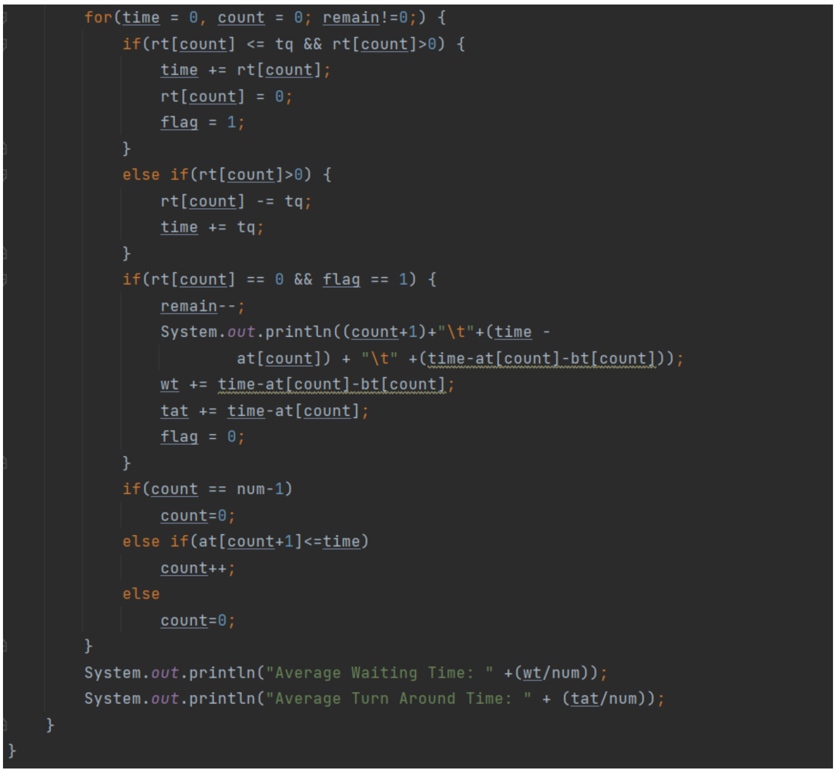
```
Enter the number of process : 5

     Enter burst time : time priorities

Process[1]:5 30

Process[2]:10 25

Process[3]:15 20

Process[4]:20 15

Process[5]:25 10


Process      Burst Time      Wait Time    Turn Around Time Priority

     5          25          0            25               10

     4          20          25           45               15

     3          15          45           60               20

     2          10          60           70               25

     1          5           70           75               30

 Average Wait Time : 40
 Average Turn Around Time : 55
```

4. CPU Scheduling Using Round Robin

Code:

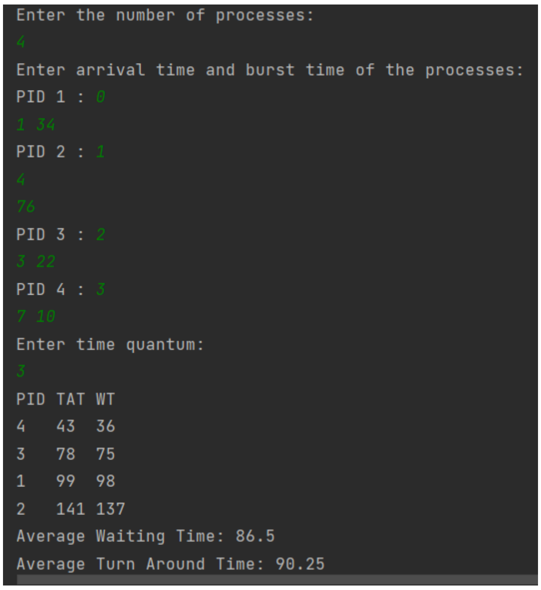


```java
import java.util.Scanner;
public class RoundRobin {
    public static void main(String[] args) {
        int count, j, num, time, remain, flag = 0, tq;
        float wt = 0, tat = 0;
        int at[] = new int[10];
        int bt[] = new int[10];
        int rt[] = new int[10];
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of processes: ");
        num = sc.nextInt();
        int pid[] = new int[num];
        remain = num;
        System.out.println("Enter arrival time and burst time of the processes: ");
        for(count = 0; count<num; count++) {
            pid[count] = count +1;
            System.out.print("PID " + (count+1)+" : ");
            at[count] = sc.nextInt();
            bt[count] = sc.nextInt();
            rt[count] = sc.nextInt();
        }
        System.out.println("Enter time quantum: ");
        tq = sc.nextInt();
        System.out.println("PID\tTAT\tWT");
```



```
        for(time = 0, count = 0; remain!=0;) {
            if(rt[count] <= tq && rt[count]>0) {
                time += rt[count];
                rt[count] = 0;
                flag = 1;
            }
            else if(rt[count]>0) {
                rt[count] -= tq;
                time += tq;
            }
            if(rt[count] == 0 && flag == 1) {
                remain--;
                System.out.println((count+1)+"\t"+(time -
                        at[count]) + "\t" +(time-at[count]-bt[count]));
                wt += time-at[count]-bt[count];
                tat += time-at[count];
                flag = 0;
            }
            if(count == num-1)
                count=0;
            else if(at[count+1]<=time)
                count++;
            else
                count=0;
        }
        System.out.println("Average Waiting Time: " +(wt/num));
        System.out.println("Average Turn Around Time: " + (tat/num));
    }
}
```

Output:



```
Enter the number of processes:
4
Enter arrival time and burst time of the processes:
PID 1 : 0
1 34
PID 2 : 1
4
76
PID 3 : 2
3 22
PID 4 : 3
7 10
Enter time quantum:
3
PID TAT WT
4   43  36
3   78  75
1   99  98
2   141 137
Average Waiting Time: 86.5
Average Turn Around Time: 90.25
```

Submitted By:
Harshita Pasupuleti
21BCE8421