Round Robin

RR.java

```
import java.util.*;
import java.io.*;

public class RR{

   public static void main(String[] args) {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter number of processes : ");
      int n = s.nextInt();

      System.out.println("Enter quantum time : ");
      int q = s.nextInt();

      int pid[] = new int[n];
      int at[] = new int[n];
      int bt[] = new int[n];
      int ct[] = new int[n];
      int tat[] = new int[n];
      int wt[] = new int[n];
      int rbt[] = new int[n];
      int f[] = new int[n];

      float avgwt = 0, avgta = 0;
      int temp;

      for (int i = 0; i < n; i++) {
         System.out.println("Enter process ID for process " + (i + 1) + ": ");
         pid[i] = s.nextInt();

        System.out.println("Enter Arrival Time for process " + (i + 1) + ": ");
         at[i] = s.nextInt();

         System.out.println("Enter Burst Time for process " + (i + 1) + ": ");
         bt[i] = s.nextInt();
         rbt[i] = bt[i];
         f[i] = 0;
      }
      System.out.println("Input accepted.");

      // Sort processes based on arrival time
      for (int i = 0; i < n; i++) {
         for (int j = i + 1; j < n; j++) {
            if (at[i] > at[j]) {
               temp = at[i];
               at[i] = at[j];
               at[j] = temp;
```

```
                temp = bt[i];
                bt[i] = bt[j];
                bt[j] = temp;

                temp = pid[i];
                pid[i] = pid[j];
                pid[j] = temp;
            }
        }
    }

    // Implementing Round Robin Scheduling
    int st = 0, tot = 0;
    while (true) {
        boolean done = true;
        for (int i = 0; i < n; i++) {
            if (at[i] <= st) {
                if (rbt[i] > 0) {
                    done = false;
                    if (rbt[i] > q) {
                        st += q;
                        rbt[i] -= q;
                    } else {
                        st += rbt[i];
                        ct[i] = st;
                        rbt[i] = 0;
                        tot++;
                    }
                }
            }
        }
        if (done)
            break;
    }


    for (int i = 0; i < n; i++) {
        tat[i] = ct[i] - at[i];
        wt[i] = tat[i] - bt[i];
        avgta += tat[i];
        avgwt += wt[i];
    }


    System.out.println("\nProcess ID\tArrival Time\tBurst Time\tCompletion Time\tTurnaround
Time\tWaiting Time");
    for (int i = 0; i < n; i++) {
```

```
            System.out.println(pid[i] + "\t\t" + at[i] + "\t\t" + bt[i] + "\t\t" + ct[i] + "\t\t\t" + tat[i] + "\t\t\t"
+ wt[i]);
        }


        System.out.println("\nAverage Turnaround Time: " + (avgta / n));
        System.out.println("Average Waiting Time: " + (avgwt / n));

        s.close();
    }
}
```

OUTPUT:-

```
Enter number of processes :
3
Enter quantum time :
3
Enter process ID for process 1:
1
Enter Arrival Time for process 1:
0
Enter Burst Time for process 1:
4
Enter process ID for process 2:
2
Enter Arrival Time for process 2:
8
Enter Burst Time for process 2:
0
Enter process ID for process 3:
3
Enter Arrival Time for process 3:
5
Enter Burst Time for process 3:
7
Input accepted.

Process ID      Arrival Time    Burst Time      Completion Time Turnaround Time Waiting Time
1               0               4               4                       4                       0
3               5               7               0                       -5                      -12
2               8               0               0                       -8                      -8

Average Turnaround Time: -3.0
Average Waiting Time: -6.6666665
```