

## RR Scheduling (Preemptive):

```
import java.util.*;

class Process {
    int processID;
    int arrival, burst, waiting, turnAround, remainingTime;
    int finish, completionTime;
}

public class RRScheduling {

    public static void main(String[] args) {
        int n, sumBurst = 0, quantum, time;
        double avgWAT = 0, avgTAT = 0;
        Scanner sc = new Scanner(System.in);
        Queue<Integer> q = new LinkedList<>();
        System.out.println("*** RR Scheduling (Preemptive) ***");
        System.out.print("Enter Number of Process: ");
        n = sc.nextInt();
        Process[] p = new Process[n];
        for (int i = 0; i < n; i++) {
            p[i] = new Process();
            p[i].processID = i + 1;
            System.out.print("Enter the arrival time for P" + (i + 1) + ": ");
            p[i].arrival = sc.nextInt();
            System.out.print("Enter the burst time for P" + (i + 1) + ": ");
            p[i].burst = sc.nextInt();
            p[i].remainingTime = p[i].burst;
            p[i].finish = 0;
            sumBurst += p[i].burst;
        }
    }
}
```

```

        System.out.println();
    }
    System.out.print("\nEnter time quantum: ");
    quantum = sc.nextInt();
    Process pTemp;
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (p[i].arrival > p[j].arrival) {
                pTemp = p[i];
                p[i] = p[j];
                p[j] = pTemp;
            }
        }
    }
    q.add(0);
    for (time = p[0].arrival; time < sumBurst;) {
        Integer I = q.remove();
        int i = I.intValue();
        if (p[i].remainingTime <= quantum) {
            time += p[i].remainingTime;
            p[i].remainingTime = 0;
            p[i].finish = 1;
            p[i].completionTime = time;
            p[i].waiting = time - p[i].arrival - p[i].burst;
            p[i].turnAround = time - p[i].arrival;
            for (int j = 0; j < n; j++) {
                Integer J = Integer.valueOf(j);
                if ((p[j].arrival <= time) && (p[j].finish != 1) && (!q.contains(J)))
                    q.add(j);
            }
        }
    }

```

```

    } else {
        time += quantum;
        p[i].remainingTime -= quantum;
        for (int j = 0; j < n; j++) {
            Integer J = Integer.valueOf(j);
            if (p[j].arrival <= time && p[j].finish != 1 && i != j && (!q.contains(J)))
                q.add(j);
        }
        q.add(i);
    }
}

System.out.println("\n*** RR Scheduling (Preemptive) ***");

System.out.println("Processor\tArrival time\tBurst time\tCompletion Time\tTurn
around time\tWaiting time");

System.out.println(
    "-----"
    "-----");

for (int i = 0; i < n; i++) {
    System.out.println("P" + p[i].processID + "\t\t" + p[i].arrival + "ms\t\t" + p[i].burst +
"ms\t\t"
    + p[i].completionTime + "ms\t\t\t" + p[i].turnAround + "ms\t\t\t" + p[i].waiting +
"ms");
    avgWAT += p[i].waiting;
    avgTAT += p[i].turnAround;
}

System.out.println("\nAverage turn around time of processor: " + (avgTAT / n)
    + "ms\nAverage waiting time of processor: " + (avgWAT / n) + "ms");
}
}

```

Output:

\*\*\* RR Scheduling (Preemptive) \*\*\*

Enter Number of Process: 5

Enter the arrival time for P1: 0

Enter the burst time for P1: 5

Enter the arrival time for P2: 1

Enter the burst time for P2: 3

Enter the arrival time for P3: 2

Enter the burst time for P3: 1

Enter the arrival time for P4: 3

Enter the burst time for P4: 2

Enter the arrival time for P5: 4

Enter the burst time for P5: 3

Enter time quantum: 2

\*\*\* RR Scheduling (Preemptive) \*\*\*

Processor	Arrival time	Burst time	Completion Time	Turn around time	Waiting time
-----					
P1	0ms	5ms	13ms	13ms	8ms
P2	1ms	3ms	12ms	11ms	8ms
P3	2ms	1ms	5ms	3ms	2ms
P4	3ms	2ms	9ms	6ms	4ms
P5	4ms	3ms	14ms	10ms	7ms

Average turn around time of processor: 8.6ms

Average waiting time of processor: 5.8ms