NAME: KOIKI DAMILARE SOLOMON

COURSE: CSC 421

ASSIGNMENT: IMPLEMENTATION OF ROUND-ROBIN SCHEDULER

Round Robin is the pre-emptive process scheduling algorithm.

How Round Robin Scheduling works

1. Processes are dispatched in a FIFO manner but are given a limited amount of time called quantum or time-slice.
2. If the process is not able to execute completely in given quantum time then the process is pre-empted and is placed at the back of the ready list.
3. Now CPU is given to the next process in ready state.
4. Same steps go until all the processes are finished.

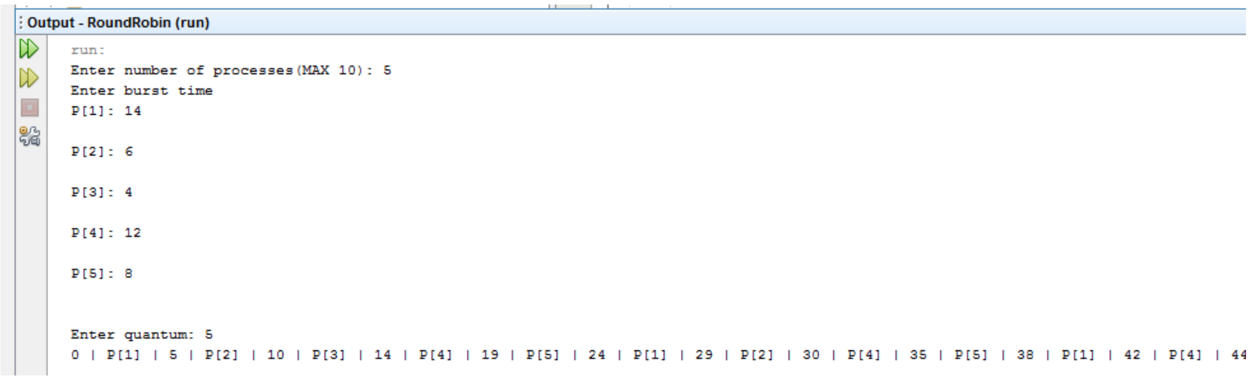
SOURCE CODE

```
import java.util.Scanner;

public class TestClass {

    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



```
run:
Enter number of processes(MAX 10): 5
Enter burst time
P[1]: 14

P[2]: 6

P[3]: 4

P[4]: 12

P[5]: 8


Enter quantum: 5
0 | P[1] | 5 | P[2] | 10 | P[3] | 14 | P[4] | 19 | P[5] | 24 | P[1] | 29 | P[2] | 30 | P[4] | 35 | P[5] | 38 | P[1] | 42 | P[4] | 44
```

Explanation of code with example:

Consider the processes below with their corresponding burst times and quantum = 5

Processes	Burst Time
P1	14
P2	6
P3	4
P4	12
P5	8

The processes are dispatched in a FIFO manner. Process P1 starts at 0 but could not complete because, quantum time is 5, so at time T=5, the process is pre-empted and placed at the back of the jobs with burst time of value 9 remaining. P2 is dispatched and it is pre-empted at time T=10 with burst time of value 1 remaining. Process P3 executed completely at once because it can be executed within the quantum time given (I.e. it has a burst time of 4 which is less than or equals to 5, the quantum time given). P4 has burst time greater than the burst time, so it has to pre-empt and it does so at time T=19. P5 also has to pre-empt and it does so at time T=24. This step continues for all unfinished processes until all processes are completely executed.

The steps above give this **Gantt chart** which corresponds with the output generated by the source code

P1	P2	P3	P4	P5	P1	P2	P4	P5	P1	P4	
0	5	10	14	19	24	29	30	35	38	42	44