```java
import java.util.Scanner;

public class roundr {

    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);

        int wtime[], btime[], rtime[], num, quantum;
        wtime = new int[10];   // Waiting times (not fully used in this code)
        btime = new int[10];   // Burst times
        rtime = new int[10];   // Remaining times

        // Step 1: Input number of processes
        System.out.print("Enter number of processes (MAX 10): ");
        num = s.nextInt();

        // Step 2: Input burst times
        System.out.print("Enter burst time for each process:");
        for (int i = 0; i < num; i++) {
            System.out.print("\nP[" + (i + 1) + "]: ");
            btime[i] = s.nextInt();
            rtime[i] = btime[i];  // Initially, remaining time = burst time
            wtime[i] = 0;
        }

        // Step 3: Input time quantum
        System.out.print("\nEnter quantum: ");
        quantum = s.nextInt();

        // Step 4: Round Robin Scheduling logic
        int rp = num;   // Remaining processes
        int i = 0;      // Process index
        int time = 0;   // Current time

        System.out.print("\nGantt Chart:\n");
        System.out.print(time);

        while (rp != 0) {
            if (rtime[i] > quantum) {
                // Process executes for full quantum
                rtime[i] -= quantum;
                time += quantum;
                System.out.print(" | P[" + (i + 1) + "] | " + time);
```

```
            }
            else if (rtime[i] > 0) {
                // Process executes for remaining time
                time += rtime[i];
                rtime[i] = 0;
                rp--;   // One process finished
                System.out.print(" | P[" + (i + 1) + "] | " + time);
            }

            i++;
            if (i == num) {
                i = 0; // Restart loop for Round Robin
            }
        }

        s.close();
    }
}
```

Output: