Write a program to simulate CPU Scheduling Algorithms: Priority (non PREEMPTIVE) in java for generalised

Priority (non PREEMPTIVE):

```
package P1;
import java.util.*;

class Process {
    int pid, arrivalTime, burstTime, priority, waitingTime, turnaroundTime,
completionTime;

    public Process(int pid, int arrivalTime, int burstTime, int priority) {
        this.pid = pid;
        this.arrivalTime = arrivalTime;
        this.burstTime = burstTime;
        this.priority = priority;
    }
}

public class NonPreemptivePriority {

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

        Process[] processes = new Process[n];
        for (int i = 0; i < n; i++) {
            System.out.print("Enter arrival time for process " + (i + 1) + ": ");
            int arrivalTime = sc.nextInt();
            System.out.print("Enter burst time for process " + (i + 1) + ": ");
            int burstTime = sc.nextInt();
            System.out.print("Enter priority for process " + (i + 1) + ": ");
            int priority = sc.nextInt();
            processes[i] = new Process(i + 1, arrivalTime, burstTime, priority);
        }

        // Sort processes by arrival time, then by priority
        Arrays.sort(processes, Comparator.comparingInt((Process p) ->
p.arrivalTime)
                .thenComparingInt(p -> p.priority));

        int currentTime = 0;
        int completed = 0;
        double totalWaitingTime = 0, totalTurnaroundTime = 0;


System.out.println("\nPID\tArrival\tBurst\tPriority\tCompletion\tWaiting\tTurnarou
nd");

        while (completed < n) {
```

```
            Process current = null;
            for (Process p : processes) {
                if (p.arrivalTime <= currentTime && p.completionTime == 0) {
                    if (current == null || p.priority < current.priority) {
                        current = p;
                    }
                }
            }

            if (current != null) {
                current.completionTime = currentTime + current.burstTime;
                current.turnaroundTime = current.completionTime -
current.arrivalTime;
                current.waitingTime = current.turnaroundTime - current.burstTime;

                totalWaitingTime += current.waitingTime;
                totalTurnaroundTime += current.turnaroundTime;

                System.out.printf("%d\t%d\t%d\t%d\t\t%d\t\t%d\t%d\n", current.pid,
current.arrivalTime,
                        current.burstTime, current.priority,
current.completionTime, current.waitingTime,
                        current.turnaroundTime);

                currentTime = current.completionTime;
                completed++;
            } else {
                currentTime++;
            }
        }

        System.out.printf("\nAverage Waiting Time: %.2f", totalWaitingTime / n);
        System.out.printf("\nAverage Turnaround Time: %.2f", totalTurnaroundTime /
n);

        sc.close();
    }
}
```

Output :

```
Enter number of processes: 4
Enter arrival time for process 1: 0
Enter burst time for process 1: 4
Enter priority for process 1: 2
Enter arrival time for process 2: 1
Enter burst time for process 2: 3
Enter priority for process 2: 1
Enter arrival time for process 3: 2
Enter burst time for process 3: 5
Enter priority for process 3: 3
Enter arrival time for process 4: 3
Enter burst time for process 4: 2
Enter priority for process 4: 1

PID    Arrival Burst Priority Completion  Waiting  Turnaround
1      0       4     2          4           0           4
2      1       3     1          7           3           6
4      3       2     1          9           4           6
3      2       5     3          14          7           12

Average Waiting Time: 3.50
Average Turnaround Time: 7.00
```