Write a program to simulate CPU Scheduling Algorithms: FCFS, SJF (Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive).

FCFS :

```
package P1;

import java.util.*;

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

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

public class CPUScheduler {

    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();
            processes[i] = new Process(i + 1, arrivalTime, burstTime);
        }

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

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


System.out.println("\nPID\tArrival\tBurst\tCompletion\tWaiting\tTurnaround");
        for (Process p : processes) {
            if (currentTime < p.arrivalTime) {
                currentTime = p.arrivalTime;
            }
            p.completionTime = currentTime + p.burstTime;
            p.turnaroundTime = p.completionTime - p.arrivalTime;
            p.waitingTime = p.turnaroundTime - p.burstTime;

            totalWaitingTime += p.waitingTime;
```

```
            totalTurnaroundTime += p.turnaroundTime;

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

            currentTime = p.completionTime;
        }

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

Output:

```
Enter number of processes: 4
Enter arrival time for process 1: 0
Enter burst time for process 1: 3
Enter arrival time for process 2: 2
Enter burst time for process 2: 5
Enter arrival time for process 3: 1
Enter burst time for process 3: 4
Enter arrival time for process 4: 5
Enter burst time for process 4: 10

PID Arrival  Burst  Completion   Waiting  Turnaround
1      0       3       3            0          3
3      1       4       7            2          6
2      2       5       12           5         10
4      5       10      22           7         17

Average Waiting Time: 3.50
Average Turnaround Time: 9.00
```

Explanation

- **Process Class**: Represents each process with attributes like `pid`, `arrivalTime`, `burstTime`, `priority`, `waitingTime`, `turnaroundTime`, and `completionTime`.
- **Input**: The program prompts the user to enter the number of processes, and for each process, its arrival time, burst time, and priority.
- **Sorting**: Processes are sorted based on their arrival times, and in case of a tie, by their priority values (lower priority value indicates higher priority).
- **Scheduling**:
    - The program iterates through the processes, selecting the one with the highest priority (lowest priority value) that has arrived and is not yet completed.
    - It calculates the completion time, waiting time, and turnaround time for each process.
- **Averages**: The program calculates and displays the average waiting time and turnaround time for all processes.