```
package a7;


import java.util.Scanner;


class Process {
   int pid;
   int waitingTime;
   int arrivalTime;
   int burstTime;
   int turnAroundTime;
   int timeToComplete;
   int completionTime = 0;


   Process(int pid, int sub, int bur) {
      this.pid = pid;
      this.arrivalTime = sub;
      this.burstTime = bur;
      this.timeToComplete = burstTime;
   }
}


public class Scheduler {
   static Scanner s = new Scanner(System.in);
```

```
public static void main(String[] args) {
    System.out.println("NAME: Bhavika Patil");
    System.out.println("ROLL NO.: TBCO22172");
    System.out.println("Enter the number of processes:");
    int n = s.nextInt();
    Process[] myProcess = new Process[n];
    for (int i = 0; i < n; i++) {
        System.out.println("Enter Arrival time and Burst Time: ");
        int sub = s.nextInt();
        int bur = s.nextInt();
        myProcess[i] = new Process(i + 1, sub, bur);
    }

    System.out.println("Select the type of scheduler to be used:");
    System.out.println("1. FCFS");
    System.out.println("2. Round Robin");
    System.out.println("3. Exit");
    System.out.println("Enter your choice:");
    int choice = s.nextInt();

    switch (choice) {
        case 1:
            FCFS(myProcess);
            break;
```

```
        case 2:
            RoundRobin(myProcess);
            break;
        case 3:
            s.close();
            System.exit(1);
            break;
        default:
            System.out.println("Incorrect Choice");
            break;
    }
    s.close();
}

static void FCFS(Process myProcess[]) {
    int x = 0;
    // Arrange processes according to their arrival time in ascending order
    Process temp;
    for (int i = 0; i < myProcess.length; i++) {
        for (int j = i; j < myProcess.length; j++) {
            if (myProcess[i].arrivalTime > myProcess[j].arrivalTime) {
                temp = myProcess[j];
                myProcess[j] = myProcess[i];
                myProcess[i] = temp;
```

```
            }
        }
    }

    for (int i = 0; i < myProcess.length; i++) {
        x = x + myProcess[i].burstTime;
        myProcess[i].completionTime = x;
        myProcess[i].turnAroundTime = myProcess[i].completionTime - myProcess[i].arrivalTime;
        myProcess[i].waitingTime = myProcess[i].turnAroundTime - myProcess[i].burstTime;
        System.out.println("Process " + myProcess[i].pid + ":");
        System.out.println("turnAroundTime\tCompletion\twaitingTime");
        System.out.println(myProcess[i].turnAroundTime + "\t\t\t" + myProcess[i].completionTime + "\t\t" + myProcess[i].waitingTime);
    }
}

static void RoundRobin(Process myProcess[]) {
    int curTimeInterval = 0, completedProcesses = 0;

    System.out.println("Specify time quantum: ");
    int quantum = s.nextInt();

    // Keep traversing all processes while all processes
    // are not done. Do following for i'th process if it is not done yet.
    while (completedProcesses < myProcess.length) {
```

```
    for (int i = 0; i < myProcess.length; i++) {
        if (myProcess[i].timeToComplete > 0 && myProcess[i].timeToComplete > quantum) {
            // Execute the process for the time quantum
            curTimeInterval += quantum;
            myProcess[i].timeToComplete -= quantum;
        } else {
            if (myProcess[i].timeToComplete > 0) {
                // Execute last cycle for the process
                curTimeInterval += myProcess[i].timeToComplete;
                myProcess[i].timeToComplete = 0;
                myProcess[i].completionTime = curTimeInterval;
                myProcess[i].turnAroundTime = myProcess[i].completionTime - myProcess[i].arrivalTime;
                myProcess[i].waitingTime = myProcess[i].turnAroundTime - myProcess[i].burstTime;
                completedProcesses++;
            }
        }
    }
}

for (int i = 0; i < myProcess.length; i++) {
    System.out.println("Process " + myProcess[i].pid + ":");
    System.out.println("turnAroundTime\tCompletion\twaitingTime");
    System.out.println(myProcess[i].turnAroundTime + "\t\t\t" + myProcess[i].completionTime + "\t\t" + myProcess[i].waitingTime);
}
```

```
  }
}
```