Problem Statement 7 priority and sjf

package C;

import java.util.Scanner;

class Process {

int pid;

int waitingTime;

int arrivalTime;

int burstTime;

int turnAroundTime;

int timeToComplete;

int completionTime = 0;

int priority;

Process(int pid, int sub, int bur, int priority) {

this.pid = pid;

this.arrivalTime = sub;

this.burstTime = bur;

this.priority = priority;

this.timeToComplete = burstTime;

}

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("ADITYA KOKATE");

System.out.println("22147");

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, Burst Time, and Priority: ");

int sub = s.nextInt();

int bur = s.nextInt();

int priority = s.nextInt();

myProcess[i] = new Process(i + 1, sub, bur, priority);

}

System.out.println("Select the type of scheduler to be used:");

System.out.println("1. SJF (Preemptive)");

System.out.println("2. Priority (Non-preemptive)");

System.out.println("3. Exit");

System.out.println("Enter your choice:");

int choice = s.nextInt();

switch (choice) {

case 1:

SJF(myProcess);

break;

case 2:

PriorityScheduling(myProcess);

break;

case 3:

System.out.println("Exited Program");

s.close();

System.exit(0);

break;

default:

System.out.println("Incorrect Choice");

break;

}

s.close();

}

static void SJF(Process myProcess[]) {

int curTimeInterval = 0, completedProcesses = 0;

Process curProcess;

curProcess = myProcess[0];

while (completedProcesses < myProcess.length) {

for (int i = 0; i < myProcess.length; i++) {

if (myProcess[i].timeToComplete > 0) {

curProcess = myProcess[i];

break;

}

}

System.out.println("Current Time Interval = " + curTimeInterval);

System.out.println("No of Processes Completed = " + completedProcesses);

for (int i = 0; i < myProcess.length; i++) {

if (myProcess[i].arrivalTime > curTimeInterval || myProcess[i].timeToComplete == 0) {

continue;

}

if (myProcess[i].timeToComplete < curProcess.timeToComplete) {

curProcess = myProcess[i];

}

}

curProcess.timeToComplete -= 1;

if (curProcess.timeToComplete == 0) {

completedProcesses++;

curProcess.completionTime = curTimeInterval + 1;

}

curTimeInterval++;

}

for (int i = 0; i < myProcess.length; i++) {

myProcess[i].waitingTime = myProcess[i].completionTime - myProcess[i].arrivalTime - myProcess[i].burstTime;

myProcess[i].turnAroundTime = myProcess[i].waitingTime + myProcess[i].burstTime;

System.out.println("Process " + myProcess[i].pid + ":");

System.out.println("Turnaround Time\tCompletion\tWaiting Time");

System.out.println(myProcess[i].turnAroundTime + "\t\t\t" + myProcess[i].completionTime + "\t\t" + myProcess[i].waitingTime);

}

}

static void PriorityScheduling(Process myProcess[]) {

Process temp;

for (int i = 0; i < myProcess.length; i++) {

for (int j = i; j < myProcess.length; j++) {

if (myProcess[i].priority > myProcess[j].priority) {

temp = myProcess[j];

myProcess[j] = myProcess[i];

myProcess[i] = temp;

}

}

}

int x = 0;

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("Turnaround Time\tCompletion\tWaiting Time");

System.out.println(myProcess[i].turnAroundTime + "\t\t\t" + myProcess[i].completionTime + "\t\t" + myProcess[i].waitingTime);

}

}

}