Problem Statement 5 (FCFS & SJF)

package C;

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 arrival, int burst) {

this.pid = pid;

this.arrivalTime = arrival;

this.burstTime = burst;

this.timeToComplete = burstTime;

}

}

public class Scheduler {

static Scanner s = new Scanner(System.in);

public static void main(String[] args) {

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 arrival = s.nextInt();

int burst = s.nextInt();

myProcess[i] = new Process(i + 1, arrival, burst);

}

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

System.out.println("1. FCFS");

System.out.println("2. SJF (Preemptive)");

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

int choice = s.nextInt();

switch (choice) {

case 1:

FCFS(myProcess);

break;

case 2:

SJF(myProcess);

break;

default:

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

break;

}

s.close();

}

static void FCFS(Process[] myProcess) {

int timeElapsed = 0;

// Arrange processes by arrival time

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

for (int j = i + 1; j < myProcess.length; j++) {

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

Process temp = myProcess[i];

myProcess[i] = myProcess[j];

myProcess[j] = temp;

}

}

}

for (Process process : myProcess) {

timeElapsed += process.burstTime;

process.completionTime = timeElapsed;

process.turnAroundTime = process.completionTime - process.arrivalTime;

process.waitingTime = process.turnAroundTime - process.burstTime;

System.out.println("Process " + process.pid + ":");

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

System.out.println(process.turnAroundTime + "\t\t" + process.completionTime + "\t\t" + process.waitingTime);

}

}

static void SJF(Process[] myProcess) {

int currentTime = 0, completedProcesses = 0;

while (completedProcesses < myProcess.length) {

Process currentProcess = null;

for (Process process : myProcess) {

if (process.timeToComplete > 0 && process.arrivalTime <= currentTime) {

if (currentProcess == null || process.timeToComplete < currentProcess.timeToComplete) {

currentProcess = process;

}

}

}

if (currentProcess == null) {

currentTime++;

continue;

}

currentProcess.timeToComplete--;

if (currentProcess.timeToComplete == 0) {

completedProcesses++;

currentProcess.completionTime = currentTime + 1;

currentProcess.waitingTime = currentProcess.completionTime - currentProcess.arrivalTime - currentProcess.burstTime;

currentProcess.turnAroundTime = currentProcess.waitingTime + currentProcess.burstTime;

}

currentTime++;

}

for (Process process : myProcess) {

System.out.println("Process " + process.pid + ":");

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

System.out.println(process.turnAroundTime + "\t\t" + process.completionTime + "\t\t" + process.waitingTime);

}

}

}