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

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;

}

}

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. Priority (Non-preemptive)");

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:

PriorityScheduling(myProcess);

break;

case 2:

RoundRobin(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 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);

}

}

static void RoundRobin(Process myProcess[]) {

int curTimeInterval = 0, completedProcesses = 0;

System.out.println("Specify time quantum: ");

int quantum = s.nextInt();

while (completedProcesses < myProcess.length) {

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

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

curTimeInterval += quantum;

myProcess[i].timeToComplete -= quantum;

} else {

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

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

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

}

}

}