**NonPreemptive Priority CPU Scheduling Algorithm**

**Code:**

import java.util.Scanner;

public class NonPreemptivePriorityCPUSchedulingAlgorithm {

int burstTime[];

int priority[];

int arrivalTime[];

String[] processId;

int numberOfProcess;

void getProcessData(Scanner input) {

System.out.print("Enter the number of Process for Scheduling : ");

int inputNumberOfProcess = input.nextInt();

numberOfProcess = inputNumberOfProcess;

burstTime = new int[numberOfProcess];

priority = new int[numberOfProcess];

arrivalTime = new int[numberOfProcess];

processId = new String[numberOfProcess];

String st = "P";

for (int i = 0; i < numberOfProcess; i++) {

processId[i] = st.concat(Integer.toString(i));

System.out.print("Enter the burst time for Process - " + (i) + " : ");

burstTime[i] = input.nextInt();

System.out.print("Enter the arrival time for Process - " + (i) + " : ");

arrivalTime[i] = input.nextInt();

System.out.print("Enter the priority for Process - " + (i) + " : ");

priority[i] = input.nextInt();

}

}

void sortAccordingArrivalTimeAndPriority(int[] at, int[] bt, int[] prt, String[] pid) {

int temp;

String stemp;

for (int i = 0; i < numberOfProcess; i++) {

for (int j = 0; j < numberOfProcess - i - 1; j++) {

if (at[j] > at[j + 1]) {

// swapping arrival time

temp = at[j];

at[j] = at[j + 1];

at[j + 1] = temp;

// swapping burst time

temp = bt[j];

bt[j] = bt[j + 1];

bt[j + 1] = temp;

// swapping priority

temp = prt[j];

prt[j] = prt[j + 1];

prt[j + 1] = temp;

// swapping process identity

stemp = pid[j];

pid[j] = pid[j + 1];

pid[j + 1] = stemp;

}

// sorting according to priority when arrival timings are same

if (at[j] == at[j + 1]) {

if (prt[j] > prt[j + 1]) {

// swapping arrival time

temp = at[j];

at[j] = at[j + 1];

at[j + 1] = temp;

// swapping burst time

temp = bt[j];

bt[j] = bt[j + 1];

bt[j + 1] = temp;

// swapping priority

temp = prt[j];

prt[j] = prt[j + 1];

prt[j + 1] = temp;

// swapping process identity

stemp = pid[j];

pid[j] = pid[j + 1];

pid[j + 1] = stemp;

}

}

}

}

}

void priorityNonPreemptiveAlgorithm() {

int finishTime[] = new int[numberOfProcess];

int bt[] = burstTime.clone();

int at[] = arrivalTime.clone();

int prt[] = priority.clone();

String pid[] = processId.clone();

int waitingTime[] = new int[numberOfProcess];

int turnAroundTime[] = new int[numberOfProcess];

sortAccordingArrivalTimeAndPriority(at, bt, prt, pid);

// calculating waiting & turn-around time for each process

finishTime[0] = at[0] + bt[0];

turnAroundTime[0] = finishTime[0] - at[0];

waitingTime[0] = turnAroundTime[0] - bt[0];

for (int i = 1; i < numberOfProcess; i++) {

finishTime[i] = bt[i] + finishTime[i - 1];

turnAroundTime[i] = finishTime[i] - at[i];

waitingTime[i] = turnAroundTime[i] - bt[i];

}

float sum = 0;

for (int n : waitingTime) {

sum += n;

}

float averageWaitingTime = sum / numberOfProcess;

sum = 0;

for (int n : turnAroundTime) {

sum += n;

}

float averageTurnAroundTime = sum / numberOfProcess;

// print on console the order of processes along with their finish time & turn

// around time

System.out.println("Priority Scheduling Algorithm : ");

System.out.format("%20s%20s%20s%20s%20s%20s%20s\n", "ProcessId", "BurstTime", "ArrivalTime", "Priority",

"FinishTime", "WaitingTime", "TurnAroundTime");

for (int i = 0; i < numberOfProcess; i++) {

System.out.format("%20s%20d%20d%20d%20d%20d%20d\n", pid[i], bt[i], at[i], prt[i], finishTime[i],

waitingTime[i], turnAroundTime[i]);

}

System.out.format("%100s%20f%20f\n", "Average", averageWaitingTime, averageTurnAroundTime);

}

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

NonPreemptivePriorityCPUSchedulingAlgorithm obj = new NonPreemptivePriorityCPUSchedulingAlgorithm();

obj.getProcessData(input);

obj.priorityNonPreemptiveAlgorithm();

}

}

**Output:**

Enter the number of Process for Scheduling : 5

Enter the burst time for Process - 0 : 4

Enter the arrival time for Process - 0 : 2

Enter the priority for Process - 0 : 6

Enter the burst time for Process - 1 : 5

Enter the arrival time for Process - 1 : 3

Enter the priority for Process - 1 : 5

Enter the burst time for Process - 2 : 5

Enter the arrival time for Process - 2 : 3

Enter the priority for Process - 2 : 3

Enter the burst time for Process - 3 : 3

Enter the arrival time for Process - 3 : 3

Enter the priority for Process - 3 : 4

Enter the burst time for Process - 4 : 1

Enter the arrival time for Process - 4 : 6

Enter the priority for Process - 4 : 3

Priority Scheduling Algorithm :

ProcessId BurstTime ArrivalTime Priority FinishTime WaitingTime TurnAroundTime

P0 4 2 6 6 0 4

P2 5 3 3 11 3 8

P3 3 3 4 14 8 11

P1 5 3 5 19 11 16

P4 1 6 3 20 13 14

Average 7.000000 10.600000