**TASK 1**

import java.util.\*;

public class Task1 {

public static void main (String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println ("Enter no. of process and the arrival time and\nburst time of the processes sequentially:");

int n = sc.nextInt(); // n means total number of processes

int[] pid = new int[n]; // pid means process id

int[] arrival\_time = new int[n];

int[] burst\_time = new int[n]; // this array will be updated every time the process runs for 1 unit time

int[] bt = new int[n]; // this array is for storing the original burst time which we will use to calculate the waiting time

int[] complete\_time = new int[n];

int[] turnaround\_time = new int[n];

int[] waiting\_time = new int[n];

boolean[] completed\_flag = new boolean[n]; // it checks if a process is completed or not

int sys\_time = 0, completed\_processes = 0;

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

pid[i] = i+1;

arrival\_time[i] = sc.nextInt();

burst\_time[i] = sc.nextInt();

bt[i] = burst\_time[i];

}

while(true) {

int current\_process = n, min\_burst = 999;

if (completed\_processes == n) // if executed process = no. of total processes

break; // our calculation is finished

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

if (arrival\_time[i] <= sys\_time

&& !completed\_flag[i]

&& burst\_time[i] < min\_burst) {

current\_process = i;

min\_burst = burst\_time[i]; // always updating the lowest burst time

}

}

++sys\_time; // in every iteration, system time will increase by 1

if (current\_process != n) {

--burst\_time[current\_process]; // because the current process has run 1 unit time

if (burst\_time[current\_process] == 0) { // checking if execution of current process is finished nor not

completed\_flag[current\_process] = true;

complete\_time[current\_process] = sys\_time;

completed\_processes++; // 1 more process completed execution

}

}

}

double avg\_turnaround\_time = 0.0, avg\_waiting\_time = 0.0;

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

turnaround\_time[i] = complete\_time[i] - arrival\_time[i];

waiting\_time[i] = turnaround\_time[i] - bt[i];

avg\_turnaround\_time += turnaround\_time[i];

avg\_waiting\_time += waiting\_time[i];

}

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

System.out.println("\nProcess " + (i+1));

System.out.println("Completion Time: " + complete\_time[i]);

System.out.println("Turnaround Time: " + turnaround\_time[i]);

System.out.println("Waiting Time: " + waiting\_time[i]);

}

System.out.println("\nAverage Turnaround Time: " + (avg\_turnaround\_time/n));

System.out.println("Average Waiting Time: " + (avg\_waiting\_time/n));

}

}

**TASK 2**

import java.util.\*;

public class Task2 {

public static void main (String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println ("Enter no. of process and the burst time and\npriority of the processes sequentially:");

int n = sc.nextInt(); // n means total number of processes

int[] pid = new int[n]; // pid means process id

int[] arrival\_time = new int[n];

int[] burst\_time = new int[n]; // this array will be updated every time the process runs for 1 unit time

int[] bt = new int[n]; // this array is for storing the original burst time which we will use to calculate the waiting time

int[] priority = new int[n];

int[] complete\_time = new int[n];

int[] turnaround\_time = new int[n];

int[] waiting\_time = new int[n];

boolean[] completed\_flag = new boolean[n]; // it checks if a process is completed or not

int sys\_time = 0, completed\_processes = 0;

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

pid[i] = i+1;

burst\_time[i] = sc.nextInt();

priority[i] = sc.nextInt();

bt[i] = burst\_time[i];

}

while(true) {

int current\_process = n, min\_priority = 999;

if (completed\_processes == n) // if executed process = no. of total processes

break; // our calculation is finished

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

if (arrival\_time[i] <= sys\_time

&& !completed\_flag[i]

&& priority[i] < min\_priority) {

current\_process = i;

min\_priority = priority[i]; // always updating the highest priority

}

}

++sys\_time; // in every iteration, system time will increase by 1

if (current\_process != n) {

--burst\_time[current\_process]; // because the current process has run 1 unit time

if (burst\_time[current\_process] == 0) { // checking if execution of current process is finished nor not

completed\_flag[current\_process] = true;

complete\_time[current\_process] = sys\_time;

completed\_processes++; // 1 more process completed execution

}

}

}

double avg\_turnaround\_time = 0.0, avg\_waiting\_time = 0.0;

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

turnaround\_time[i] = complete\_time[i] - arrival\_time[i];

waiting\_time[i] = turnaround\_time[i] - bt[i];

avg\_turnaround\_time += turnaround\_time[i];

avg\_waiting\_time += waiting\_time[i];

}

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

System.out.println("\nProcess " + (i+1));

System.out.println("Completion Time: " + complete\_time[i]);

System.out.println("Turnaround Time: " + turnaround\_time[i]);

System.out.println("Waiting Time: " + waiting\_time[i]);

}

System.out.println("\nAverage Turnaround Time: " + (avg\_turnaround\_time/n));

System.out.println("Average Waiting Time: " + (avg\_waiting\_time/n));

}

}

**TASK 3**

import java.util.\*;

public class Task3 {

static int getNextProcess(boolean[] a, int x) {

int iteration = 0;

for (int i = x+1;; ++i) {

if (i == a.length) i = 0;

if (!a[i]) return i;

++iteration;

if (iteration == a.length) return -1;

}

}

public static void main (String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the Time Quantum:");

int quantum = sc.nextInt();

System.out.println ("Enter no. of process and the burst time\nof the processes sequentially:");

int n = sc.nextInt(); // n means total number of processes

int[] pid = new int[n]; // pid means process id

int[] arrival\_time = new int[n];

int[] burst\_time = new int[n]; // this array will be updated every time the process runs for 1 unit time

int[] bt = new int[n]; // this array is for storing the original burst time which we will use to calculate the waiting time

int[] runtime = new int[n];

int[] complete\_time = new int[n];

int[] turnaround\_time = new int[n];

int[] waiting\_time = new int[n];

boolean[] completed\_flag = new boolean[n]; // it checks if a process is completed or not

int sys\_time = 0, completed\_processes = 0;

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

pid[i] = i+1;

burst\_time[i] = sc.nextInt();

bt[i] = burst\_time[i];

}

int next\_process = 0;

while(true) {

int current\_process = n;

if (completed\_processes == n) // if executed process = no. of total processes

break; // our calculation is finished

for (int i = next\_process; i < n; i++) {

if (!completed\_flag[i]) {

if (runtime[i] < quantum) {

current\_process = i;

break;

} else {

runtime[i] = 0;

next\_process = getNextProcess(completed\_flag, i);

if (next\_process == -1) break;

current\_process = next\_process;

}

}

}

++sys\_time; // in every iteration, system time will increase by 1

if (current\_process != n) {

runtime[current\_process]++;

--burst\_time[current\_process]; // because the current process has run 1 unit time

if (burst\_time[current\_process] == 0) { // checking if execution of current process is finished nor not

completed\_flag[current\_process] = true;

complete\_time[current\_process] = sys\_time;

completed\_processes++; // 1 more process completed execution

next\_process = getNextProcess(completed\_flag, current\_process);

if (next\_process == -1) break;

}

}

}

double avg\_turnaround\_time = 0.0, avg\_waiting\_time = 0.0;

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

turnaround\_time[i] = complete\_time[i] - arrival\_time[i];

waiting\_time[i] = turnaround\_time[i] - bt[i];

avg\_turnaround\_time += turnaround\_time[i];

avg\_waiting\_time += waiting\_time[i];

}

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

System.out.println("\nProcess " + (i+1));

System.out.println("Completion Time: " + complete\_time[i]);

System.out.println("Turnaround Time: " + turnaround\_time[i]);

System.out.println("Waiting Time: " + waiting\_time[i]);

}

System.out.println("\nAverage Turnaround Time: " + (avg\_turnaround\_time/n));

System.out.println("Average Waiting Time: " + (avg\_waiting\_time/n));

}

}