**Write a Program to Demonstrate the use of FCFS CPU Scheduling algorithm.**

import java.util.\*;

public class FCFS

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.println("enter no of process: ");

int n = sc.nextInt();

int pid[] = new int[n];

int ar[] = new int[n];

int bt[] = new int[n];

int ct[] = new int[n];

int ta[] = new int[n];

int wt[] = new int[n];

int temp;

float avgwt=0,avgta=0;

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

{

System.out.println("enter process " + (i+1) + " arrival time: ");

ar[i] = sc.nextInt();

System.out.println("enter process " + (i+1) + " burst time: ");

bt[i] = sc.nextInt();

pid[i] = i+1;

}

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

{

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

{

if( ar[j] > ar[j+1] )

{

temp = ar[j];

ar[j] = ar[j+1];

ar[j+1] = temp;

temp = bt[j];

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

bt[j+1] = temp;

temp = pid[j];

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

pid[j+1] = temp;

}

}

}

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

{

if( i == 0)

{

ct[i] = ar[i] + bt[i];

}

else

{

if( ar[i] > ct[i-1])

{

ct[i] = ar[i] + bt[i];

}

else

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

}

ta[i] = ct[i] - ar[i] ;

wt[i] = ta[i] - bt[i] ;

avgwt = avgwt + wt[i] ;

avgta = avgta + ta[i] ;

}

System.out.println("\npid arrival burst complete turn waiting");

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

{

System.out.println(pid[i] + " \t " + ar[i] + "\t" + bt[i] + "\t" + ct[i] + "\t" + ta[i] + "\t" + wt[i] ) ;

}

sc.close();

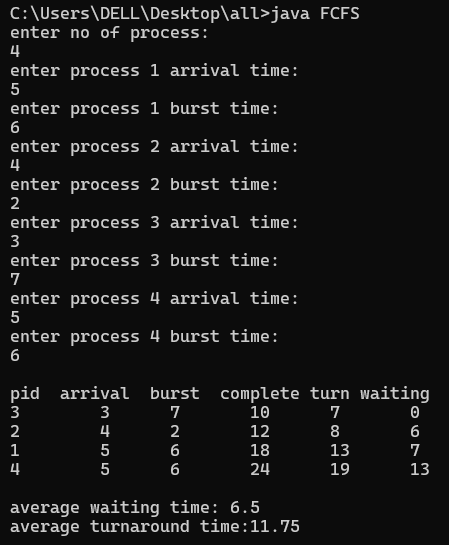
System.out.println("\naverage waiting time: "+ (avgwt/n) );

System.out.println("average turnaround time:"+ (avgta/n) );

}

}

**OUTPUT:**

****

**Write a program to Demonstrate the use of SJF CPU Scheduling algorithm.**

import java.util.\*;

public class SJF

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.println(" Enter no of process: ");

int n = sc.nextInt();

int pid[] = new int[n];

int at[] = new int[n];

int bt[] = new int[n];

int ct[] = new int[n];

int ta[] = new int[n];

int wt[] = new int[n];

int f[] = new int[n];

int st=0,tot=0;

float avgwt=0, avgta=0;

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

{

System.out.println("Enter process " + (i+1) + "arrival time: ");

at[i]=sc.nextInt();

System.out.println("Enter process " + (i+1) + "Brust time: ");

bt[i]=sc.nextInt();

pid[i]=i+1;

f[i]=0;

}

boolean a=true;

while (true)

{

int c=n, min=999;

if (tot == n)

break;

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

{

if((at[i]<=st) && (f[i]==0) && (bt[i]<min))

{

min= bt[i];

c=i;

}

}

if(c == n)

st ++;

else

{

ct[c] = st + bt[c];

st ++;

ta[c] = ct[c] - at[c];

wt[c] = ta[c] - bt[c];

f[c] = 1;

tot ++;

}

}

System.out.println("\n pid arrival burst complete turn waiting");

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

{

avgwt = avgwt + wt[i];

avgta = avgta + ta[i];

System.out.println(pid[i] + "\t" + at[i] + "\t" + bt[i] + "\t" + ct[i] + "\t" + ta[i] + "\t" + wt[i]);

}

System.out.println("\n average tat is " + (float) (avgta/n));

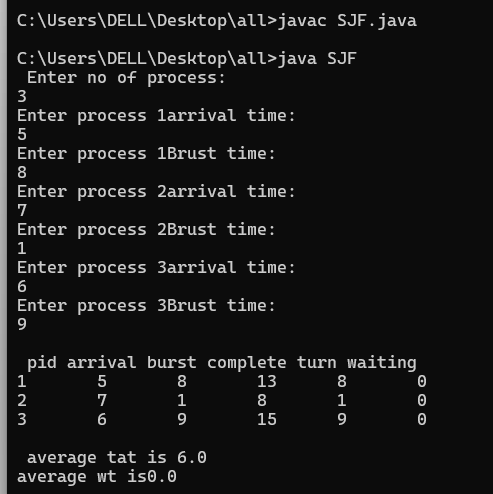
System.out.println("average wt is" + (float) (avgwt/n));

sc.close();

}

}

**OUTPUT**:



**Write a program to Demonstrate the use of Priority CPU Scheduling algorithm**

import java.util.\*;

class PriorityTest

{

public static void main(String args[])

{

int n;

Scanner in = new Scanner(System.in);

System.out.println("Enter the number of processes;");

n=in.nextInt();

int b[]=new int[n];

int p[]=new int[n];

int pid[]=new int[n];

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

{

System.out.println(" Enter burst time and priority value for process:"+(i+1));

b[i]=in.nextInt();

p[i]=in.nextInt();

pid[i]=i+1;

}

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

{

int a=p[i],m=i;

for(int j=i; j<n; j++)

{

if (p[j]>a)

{

a=p[j];

m=j;

}

}

int temp;

temp = p[i];

p[i]=p[m];

p[m]=temp;

temp=b[i];

b[i]=b[m];

b[m]=temp;

pid[m]=temp;

}

int t=0;

System.out.println("\n order of process execution is:");

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

{

System.out.println("p"+pid[i]+"is executed from "+t+" to "+(t+b[i]));

t=t+b[i];

}

System.out.println("\n");

System.out.println("Process Id Burst Time Wait Time TurnAroundTime\n");

int wt=0;

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

{

int tt= wt+b[i];

System.out.println("P"+ pid[i]+ "\t\t" + b[i] + "\t\t" + wt + "\t\t" + tt);

wt=wt + b[i];

}

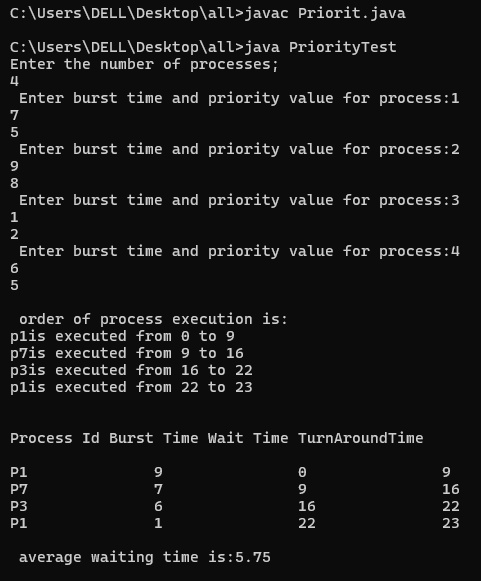
float avg\_wt=(float)wt/n;

System.out.println("\n average waiting time is:"+ avg\_wt);

}

}

**OUTPUT:**

****

**Write a program to demonstrate the use of Round Robin CPU Scheduling   
Algorithm.**

import java.util.Scanner;

public class RoundRobin

{

public static void main(String args [])

{

int n,i,qt,count=0,temp,sq=0,bt[],wt[],tat[],rem\_bt[];

float awt =0,atat=0;

bt=new int[10];

wt=new int[10];

tat=new int[10];

rem\_bt=new int[10];

Scanner s=new Scanner(System.in);

System.out.print("Enter the number of process(maximum 10)= ");

n=s.nextInt();

System.out.println ("Enter the burst time of the process \n");

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

{

System.out.print("P"+i+" = ");

bt[i]=s.nextInt();

rem\_bt[i]=bt[i];

}

System.out.print("Enter the Quantum time : ");

qt=s.nextInt();

while (true)

{

for(i=0,count=0;i<n;i++)

{

temp=qt;

if (rem\_bt[i]==0)

{

count ++;

continue;

}

if (rem\_bt[i]>qt)

rem\_bt [i]=rem\_bt[i]-qt;

else

if (rem\_bt[i]>=0)

{

temp =rem\_bt[i];

rem\_bt[i]=0;

}

sq=sq+temp;

tat [i] = sq;

}

if (n==count)

break;}

System.out.println("---------------------------------------------");

System.out.println("\nProcess\t Burst Time\t TurnAround time\t   
Waiting Time\n-----");

System.out.println("--------------------------------------------");

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

{

wt[i] = tat[i]-bt[i];

awt=awt +wt[i];

atat=atat + tat[i];

System.out.print("\n "+ (i+1)+"\t"+bt[i]+"\t\t"+tat[i]+"\t\t"+wt  
[i]+"\n");

}

awt = awt/n;

atat = atat/n;

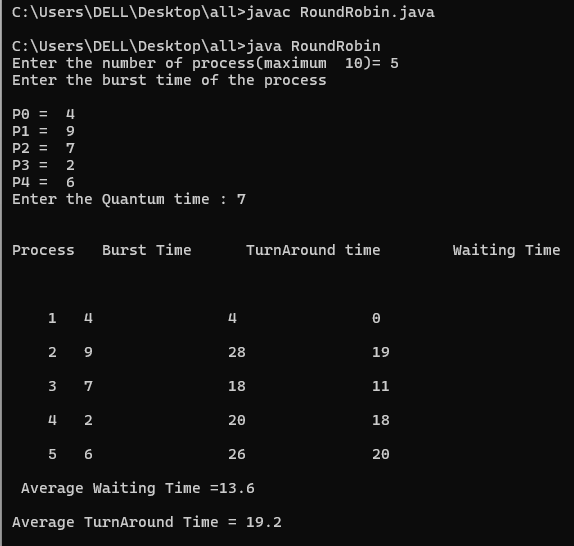
System.out.println("\n Average Waiting Time ="+awt +"\n");

System.out.println("Average TurnAround Time = " + atat );

}

}

**OUTPUT:**

****

**Write a program to demonstrate the use of resource allocation to identify whether the system is in safe state.**

import java.util.\*;

class BankersAlgo {

int max[][];

int need[][];

int available[][];

int allocation[][];

int np, nr;

public void input() {

Random random = new Random();

Scanner input = new Scanner(System.in);

System.out.println("Enter the number of Processes & number of Resources: ");

np = input.nextInt();

nr = input.nextInt();

max = new int[np][nr];

need = new int[np][nr];

available = new int[1][nr];

allocation = new int[np][nr];

// Generating random allocation and maximum need

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

for (int j = 0; j < nr; j++) {

allocation[i][j] = random.nextInt(6); // Random allocation between 0 to 5

max[i][j] = random.nextInt(5) + 5; // Random maximum need between 5 to 9

}

}

// Generating random available resources

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

available[0][i] = random.nextInt(10);

}

}

public void calculateNeed() {

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

for (int j = 0; j < nr; j++) {

need[i][j] = max[i][j] - allocation[i][j];

}

}

}

public boolean check(int p) {

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

if (available[0][i] < need[p][i]) {

return false;

}

}

return true;

}

public void algorithm() {

calculateNeed();

int c = 0;

boolean status[] = new boolean[np];

while (c < np) {

boolean allocated = false;

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

if (!status[i] && check(i)) {

status[i] = true;

allocated = true;

c++;

System.out.println("Allocated process: " + i);

for (int j = 0; j < nr; j++) {

available[0][j] += allocation[i][j];

}

}

}

if (!allocated) break; // If no allocation

}

if (c == np) // If all processes are allocated

System.out.println("\nAll processes allocated safely");

else

System.out.println("All processes cannot be allocated safely");

}

public static void main(String[] args) {

BankersAlgo obj = new BankersAlgo();

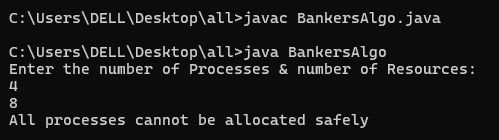
obj.input();

obj.algorithm();

}

}

**OUTPUT:**



**Write a program to demonstrate the use of First fit memory allocational algorithm.**

import java.util.Scanner;

import java.util.Random;

class FirstFit {

static void firstFit(int blockSize[], int m, int processSize[], int n) {

int allocation[] = new int[n];

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

allocation[i] = -1;

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

for (int j = 0; j < m; j++) {

if (blockSize[j] >= processSize[i]) {

allocation[i] = j;

blockSize[j] -= processSize[i];

break;

}

}

}

System.out.println("\nProcess No.\tProcess Size\tBlock Size\tBlock no.");

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

System.out.print(" " + (i+1) + "\t\t" + processSize[i] + "\t\t"+blockSize[i]+ "\t\t");

if (allocation[i] != -1)

System.out.print(allocation[i] + 1);

else

System.out.print("Not Allocated");

System.out.println();

}

}

public static void main(String[] args) {

Random random = new Random();

Scanner input = new Scanner(System.in);

System.out.print("Enter no. of Processes : ");

int in = input.nextInt();

System.out.print("Enter no. of Blocks : ");

int im = input.nextInt();

int processSize[] = new int[in];

int blockSize[] = new int[im];

for (int j = 0; j < in; j++) {

processSize[j] = random.nextInt(500);

}

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

blockSize[i] = random.nextInt(500);

}

int m = blockSize.length;

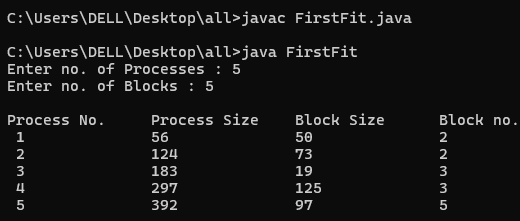
int n = processSize.length;

firstFit(blockSize, m, processSize, n);

}

}

**OUTPUT:**

****

**Write a program to demonstrate the use of LRU Page Replacement algorithm.**

import java.util.Scanner;

class LRU

{

public static int min(int counter[ ],int nFrames)

{

int minimum = counter[0];

int pos = 0;

for(int i=0;i<nFrames;i++)

{

if(minimum > counter[i])

pos = i;

}

return pos;

}

public static void main(String args[ ])

{

Scanner s = new Scanner(System.in);

int n,recent = 0,pageFault = 0,nFrames;

System.out.print("Enter the number of pages: ");

n = s.nextInt();

int pageString[] = new int[n];

System.out.print("Enter the page reference string: ");

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

pageString[i]=s.nextInt();

System.out.print("\nEnter the number of frames: ");

nFrames = s.nextInt();

int frames[] = new int[nFrames];

int counter[] = new int[nFrames];

for(int i=0;i<nFrames;i++)

{ frames[i] = 0;

counter[i] = 0;

}

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

{

int flag =0;

for(int j=0;j<nFrames;j++)

{

if(frames[ j ] == pageString[ i ])

{ flag=1;

counter[ j ] = recent++;

break;

}

}

if(flag == 0)

{

for(int j=0;j<nFrames;j++)

{if(frames[ j ] == 0)

{ frames[ j ] = pageString[i];

counter[ j ] = recent++;

flag=1;

pageFault++;

break;

}

}

}

if(flag == 0){

int PositionToreplace = min(counter, nFrames);

frames[PositionToreplace] = pageString[ i ];

counter[PositionToreplace] = recent++;

pageFault++;

}

System.out.println();

for(int j=0; j<nFrames; j++)

{

System.out.print(frames[ j ]+" ");

}

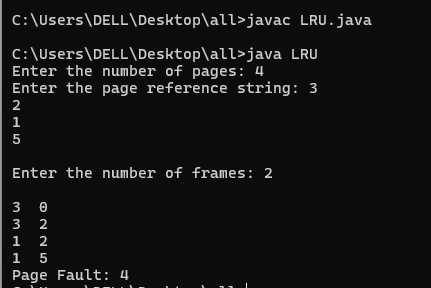
}

System.out.print("\nPage Fault: "+pageFault);

}

}

**OUTPUT:**

****

**Write a program to demonstrate the use of FIFO Page Replacement algorithm.**

import java.util.\*;

class FIFOPR

{

public static void main(String[]args)

{

Scanner s=new Scanner(System.in);

int frames, pointer=0, hit=0, fault=0, ref\_len;

int buffer[];

int reference[];

int mem\_layout[][];

System.out.println("Enter the number of Frames:");

frames=s.nextInt();

System.out.println("Enter the length of the Reference string:");

ref\_len=s.nextInt();

reference=new int[ref\_len];

mem\_layout=new int[ref\_len][frames];

buffer=new int[frames];

for(int j=0;j<frames;j++)

buffer[j]=-1;

System.out.println("Enter the reference string:");

for(int i=0;i<ref\_len;i++)

{

reference[i]=s.nextInt();

}

System.out.println();

for(int i=0;i<ref\_len;i++)

{

int search=-1;

for(int j=0;j<frames;j++)

{

if(buffer[j]==reference[i])

{

search=j;

hit++;

break;

}

}

if(search==-1)

{

buffer[pointer]=reference[i];

fault++;

pointer++;

if(pointer==frames)

pointer=0;

}

for(int j=0;j<frames;j++)

mem\_layout[i][j]=buffer[j];

}

for(int i=0;i<frames;i++)

{

for(int j=0;j<ref\_len;j++)

System.out.print(mem\_layout[j][i]+" ");

System.out.println();

}

System.out.println("The number of Hits:"+hit);

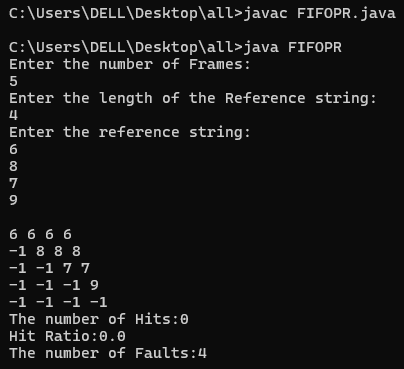
System.out.println("Hit Ratio:"+(float)((float)hit/ref\_len));

System.out.println("The number of Faults:"+fault);

}

}

**OUTPUT:**

****

**Write a program to Demonstrate the use of Optimal Page Replacement algorithm.**

import java.util.\*;

class OPR {

public static void main(String args[]) {

Scanner s = new Scanner(System.in);

int frames, pointer = 0, hit = 0, fault = 0, ref\_len;

boolean isFull = false;

int buffer[];

int reference[];

int mem\_layout[][];

System.out.println("Enter the number of frames:");

frames = s.nextInt();

System.out.println("Enter the length of the Reference string:");

ref\_len = s.nextInt();

reference = new int[ref\_len];

mem\_layout = new int[ref\_len][frames];

buffer = new int[frames];

for (int j = 0; j < frames; j++)

buffer[j] = -1;

System.out.println("Enter the reference string:");

for (int i = 0; i < ref\_len; i++) {

reference[i] = s.nextInt();

}

System.out.println();

for (int i = 0; i < ref\_len; i++) {

int search = -1;

for (int j = 0; j < frames; j++) {

if (buffer[j] == reference[i]) {

search = j;

hit++;

break;

}

}

if (search == -1) {

if (!isFull) {

buffer[pointer] = reference[i];

fault++;

pointer++;

if (pointer == frames) {

pointer = 0;

isFull = true;

}

} else {

int[] nextIndexes = new int[frames];

Arrays.fill(nextIndexes, Integer.MAX\_VALUE);

for (int j = 0; j < frames; j++) {

for (int k = i + 1; k < ref\_len; k++) {

if (reference[k] == buffer[j]) {

nextIndexes[j] = k;

break;

}

}

}

int maxIndex = -1;

int maxIndexValue = Integer.MIN\_VALUE;

for (int j = 0; j < frames; j++) {

if (nextIndexes[j] > maxIndexValue) {

maxIndexValue = nextIndexes[j];

maxIndex = j;

}

}

buffer[maxIndex] = reference[i];

fault++;

}

}

}

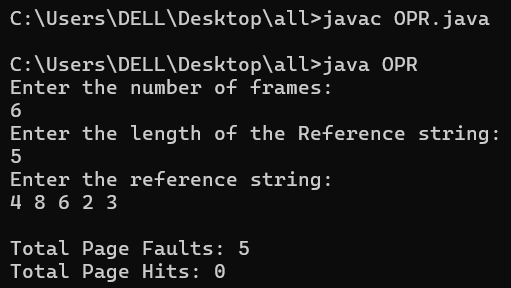
System.out.println("Total Page Faults: " + fault);

System.out.println("Total Page Hits: " + hit);

}

}

**OUTPUT:**

****

**Write a program to implement the producer-consumer problem using semaphores.**

import java.util.concurrent.Semaphore;

class Q {

int item;

static Semaphore semCon = new Semaphore(0);

static Semaphore semProd = new Semaphore(1); void get()

{

try { semCon.acquire();

}

catch (InterruptedException e) {

System.out.println("InterruptedException caught");

}

System.out.println("Consumer consumed item : " + item);

semProd.release();

}

void put(int item)

{

try { semProd.acquire();

}

catch (InterruptedException e) {

System.out.println("InterruptedException caught");

}

this.item = item;

System.out.println("Producer produced item : " + item);

semCon.release();

}

}

class Producer implements Runnable { Q q;

Producer(Q q)

{

this.q = q;

new Thread(this, "Producer").start();

}

public void run()

{

for (int i = 0; i < 5; i++) q.put(i);

}

}

class Consumer implements Runnable {

Q q;

Consumer(Q q)

{

this.q = q;

new Thread(this, "Consumer").start();

}

public void run(){

for (int i = 0; i < 5; i++)

// consumer get items

q.get();

}

}

class PC {

public static void main(String args[])

{

Q q = new Q();

new Consumer(q);

new Producer(q);

}

}

**OUTPUT:**

