**1. OS DETAILS:**

**Program:**

import java.io.\*;

class osdetails

{

public static void main(String args[])

{

String osname,osver,osarc,jver,jcver;

osname=new String(System.getProperty("os.name"));

osver=new String(System.getProperty("os.version"));

osarc=new String(System.getProperty("os.arch"));

jver=new String(System.getProperty("java.version"));

jcver=new String(System.getProperty("java.class.version"));

System.out.println("The OS Name = "+osname);

System.out.println("The OS Version = "+osver);

System.out.println("The OS Architecture = "+osarc);

System.out.println("The Java Version = "+jver);

System.out.println("The Java Class Version = "+jcver);

} }

**Output:**

The OS Name = Windows 2000

The OS Version = 5.0

The OS Architecture = x86

The Java Version = 1.3.0rc1

The Java Class Version = 47.0

**2. FCFS SCHEDULING:**

**Program:**

import java.io.\*;

import java.util.\*;

class fcfs

{

public static void main(String args[]) throws IOException

{

int i,n;

float tot[]=new float[100];

float cpu[]=new float[100];

float wait[]=new float[100];

Scanner sn= new Scanner(System.in);

System.out.println("\t\t\tWELCOME");

System.out.println("Enter the no. of processes");

n=sn.nextInt();

System.out.println("Enter cpu time\n");

for(i=1;i<=n;i++)

{

cpu[i]=sn.nextFloat();

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

}

wait[1]=0;

tot[1]=cpu[1];

for(i=2;i<=n;i++)

{

wait[i]=tot[i-1];

tot[i]=tot[i-1]+cpu[i];

}

System.out.println("\n\tprocid\tcpu time\twait time\ttotal time");

for(i=1;i<=n;i++)

{

System.out.println("\t"+i+"\t\t"+cpu[i]+"\t\t"+wait[i]+"\t\t"+tot[i]);

}

float totwait=0,total=0;

float avgwait,avgtot;

for(i=1;i<=n;i++)

{

totwait=totwait+wait[i];

total=total+tot[i];

}

avgtot=total/n;

avgwait=totwait/n;

System.out.println("the result is");

System.out.println("\n avg wait time = "+avgwait);

System.out.println("\n avg total time = " +avgtot);

}

}

**Output:**

WELCOME

Enter the no. of processes

3

Enter cpu time

7

5

3

procid cpu time wait time total time

1 7.0 0.0 7.0

2 5.0 7.0 12.0

3 3.0 12.0 15.0

the result is

avg wait time = 6.3333335

avg total time = 11.333333

**3. SJF SCHEDULING:**

**Program:**

import java.io.\*;

import java.util.\*;

class sjf

{

public static void main(String args[]) throws IOException

{

int i,j,n;

float tot[]=new float[100];

float cpu[]=new float[100];

float wait[]=new float[100];

int pid[]=new int[100];

Scanner sn= new Scanner(System.in);

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

System.out.println("enter the no. of processes");

n=sn.nextInt();

System.out.println("Enter cpu time\n");

for(i=1;i<=n;i++)

{

cpu[i]=sn.nextFloat();

pid[i]=i;

}

for(i=1;i<=n;i++)

{

for(j=i+1;j<=n;j++)

{

if(cpu[i]>cpu[j])

{

float t=cpu[i];

cpu[i]=cpu[j];

cpu[j]=t;

}

}

}

wait[1]=0;

tot[1]=cpu[1];

for(i=1;i<=n;i++)

{

wait[i]=tot[i-1];

tot[i]=tot[i-1]+cpu[i];

}

float totwait=0,total=0;

float avgwait,avgtotal;

for(i=1;i<=n;i++)

{

totwait=totwait+wait[i];

total=total+tot[i];

}

avgwait=totwait/n;

avgtotal=total/n;

System.out.println("the result is");

System.out.println("\n\tpid\tcpu time\twait time\ttotal time");

for(i=1;i<=n;i++)

{

System.out.println("\n\t"+pid[i]+"\t"+cpu[i]+"\t\t"+wait[i]+"\t\t"+tot[i]);

}

System.out.println("\n avg wait time = " +avgwait);

System.out.println("\n avg total time = "+avgtotal);

}

}

**Output:**

WELCOME

enter the no. of processes

3

Enter cpu time

5

6

8

the result is

pid cpu time wait time total time

1 5.0 0.0 5.0

2 6.0 5.0 11.0

3 8.0 11.0 19.0

avg wait time = 5.3333335

avg total time = 11.666667

**4. PRIORITY SCHEDULING:**

**Program:**

import java.io.\*;

import java.util.\*;

class priority

{

public static void main(String args[]) throws IOException

{

int i,j,n;

float tot[]=new float[100];

float cpu[]=new float[100];

float wait[]=new float[100];

int pri[]=new int[10];

Scanner sn=new Scanner(System.in);

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

System.out.println("enter the no. of processes");

n=sn.nextInt();

System.out.println("Enter cpu time");

for(i=1;i<=n;i++)

{

cpu[i]=sn.nextFloat();

}

System.out.println("Enter the priority");

for(i=1;i<=n;i++)

{

pri[i]=sn.nextInt();

}

for(i=1;i<=n;i++)

{

for(j=i+1;j<=n;j++)

{

if(pri[i]>pri[j])

{

int p=pri[i];

pri[i]=pri[j];

pri[j]=p;

float t=cpu[i];

cpu[i]=cpu[j];

cpu[j]=t;

}

}

}

wait[1]=0;

tot[1]=cpu[1];

for(i=1;i<=n;i++)

{

wait[i]=tot[i-1];

tot[i]=tot[i-1]+cpu[i];

}

float totwait=0,total=0;

float avgwait,avgtot;

for(i=1;i<=n;i++)

{

totwait=totwait+wait[i];

total=total+tot[i];

}

avgwait=totwait/n;

avgtot=total/n;

System.out.println("the result is");

System.out.println("\n\t pid\tcpu time\tpriority\twait time\ttotal time");

for(i=1;i<=n;i++)

{

System.out.println("\n\t"+i+"\t\t"+cpu[i]+"\t\t"+pri[i]+"\t\t"+wait[i]+"\t\t"+tot[i]);

}

System.out.println("\n avg wait time = "+avgwait);

System.out.println("\n avg total time = "+avgtot);

}

}

**Output:**

WELCOME

enter the no. of processes

3

Enter cpu time

5

6

3

Enter the priority

2

3

1

the result is

pid cpu time priority wait time total time

1 3.0 1 0.0 3.0

2 5.0 2 3.0 8.0

3 6.0 3 8.0 14.0

avg wait time = 3.6666667

avg total time = 8.333333

**5. MEMORY MANAGEMENT:**

**Program:**

import java.io.\*;

import java.lang.\*;

import java.util.\*;

class memmgnt

{

public static void main(String args[])throws IOException

{

int found,i,n,j,ch,p=0;

int all[]=new int[10];

int size[]=new int[10];

int req[]=new int[10];

System.out.println("\t\t\tMEMORY MANAGEMENT\n");

int allocate[][]=new int[10][10];

Scanner sn=new Scanner(System.in);

System.out.println("Enter the No. of Partition: ");

p=sn.nextInt();

System.out.println("Enter the Part Size: ");

for(i=1;i<=p;i++)

size[i]=sn.nextInt();

System.out.println("Enter N: ");

n=sn.nextInt();

System.out.println("Enter the Mem Req of Jobs");

for(i=1;i<=n;i++)

{

req[i]=sn.nextInt();

}

System.out.println("CHOOSE TYPE");

System.out.println("1. FIRST FIT");

System.out.println("2. BEST FIT");

System.out.println("3. WORST FIT");

System.out.println("Enter UR Choice");

ch=sn.nextInt();

switch(ch)

{

case 1:

break;

case 2:

for (int m=1;m<=n;m++)

{

for(int o=m+1;o<=p;o++)

{

if(size[m]>size[o])

{

int temp=size[m];

size[m]=size[o];

size[o]=temp;

}

}

}

break;

case 3:

for (int m=1;m<=n;m++)

{

for(int o=m+1;o<=p;o++)

{

if(size[m]<=size[o])

{

int temp=size[m];

size[m]=size[o];

size[o]=temp;

} } }

break;

}

System.out.println("\t\t\tMEMORY MANAGEMENT\n");

for(j=1;j<=n;j++)

{

i=1;found=0;

while((i<=p)&&(found==0))

{

if((size[i]>=req[j])&&(all[i]==0))

{

allocate[j][1]=j;

allocate[j][2]=size[i];

allocate[j][3]=req[j];

all[i]=1;

found=1;

}

i++;

}

if((i>p)&&(found==0))

System.out.println("The job size "+req[j]+"cant be accomodated");

}

System.out.println("\n\t\t\tALLOCATION TABLE");

System.out.println("sno\tjob no\tsize part\tjobsize");

for(i=1;i<=n;i++)

{

System.out.println(i+"\t"+allocate[i][1]+"\t "+allocate[i][2]+"\t\t"+allocate[i][3]);

}

System.out.println("\n\t\t\tFREE TABLE");

System.out.println("PAR NO\tSize");

for(j=1;j<=p;j++)

if(all[j]!=1)

System.out.println(j+"\t"+size[j]);

} }

**Output:**

MEMORY MANAGEMENT

Enter the No. of Partition:

3

Enter the Part Size:

7

6

3

Enter N:

3

Enter the Mem Req of Jobs

4

3

2

CHOOSE TYPE

1. FIRST FIT

2. BEST FIT

3. WORST FIT

Enter UR Choice

1

MEMORY MANAGEMENT

ALLOCATION TABLE

sno job no size part jobsize

1 1 7 4

2 2 6 3

3 3 3 2

FREE TABLE

PAR NO Size

**6. DINING PHILOSOPHER PROBLEM:**

**Program:**

import java.io.\*;

import java.util.\*;

class phil

{

public static void main(String args[])throws IOException

{

int phi[]=new int[10];

System.out.println("No. of Philosophers are 1, 2, 3, 4 & 5");

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

System.out.println("2. Stop");

System.out.println("3. Exit");

System.out.println("4. Display");

Scanner sn=new Scanner(System.in);

int i,n=0,v;

while(true)

{

System.out.println("Enter the Choice");

v=sn.nextInt();

if(v==1)

{

System.out.println("Enter the philosopher no. who wants to eat");

n=sn.nextInt();

if(phi[n]==1)

{

System.out.println(n+"is already eating");

continue;

}

if(n!=1&&n!=5)

{

if(phi[n+1]==1||phi[n-1]==1)

System.out.println("Spoon not available for "+n);

if(phi[n+1]==0&&phi[n-1]==0)

{

System.out.println("Entering");

phi[n]=1;

System.out.println("Eating");

}

}

else if(n==1)

{

if(phi[5]!=0||phi[n+1]!=0)

System.out.println("Spoon not available for "+n);

if(phi[5]==0&&phi[n+1]==0)

{

System.out.println("Entering");

phi[n]=1;

System.out.println("Eating");

}

}

else if(n==5)

{

if(phi[1]!=0||phi[n-1]!=0)

System.out.println("Spoon not available for "+n);

if(phi[1]==0&&phi[n-1]==0)

{

System.out.println("Entering");

phi[n]=1;

System.out.println("Eating");

}

}

}

if(v==3)

{

break;

}

if(v==2)

{

int v1;

System.out.println("No. of philosopher who wants to stop eating");

v1=sn.nextInt();

if(phi[v1]==0)

{

System.out.println("The philosopher "+v1+" is not eating state");

}

else

{

System.out.println("Philosopher "+v1+" stops eating");

phi[v1]=0;

}

}

if(v==4)

{

int count=0;

for(i=1;i<=5;i++)

{

if(phi[i]==1)

{

System.out.println("Philosopher "+i+" is eating");

count++;

}

}

if(count==0)

{

System.out.println("No one is eating");

}

}

}

}

}

**Output:**

No. of Philosophers are 1, 2, 3, 4 & 5

1. Eat

2. Stop

3. Exit

4. Display

Enter the Choice

1

Enter the philosopher no. who wants to eat

1

Entering

Eating

Enter the Choice

1

Enter the philosopher no. who wants to eat

2

Spoon not available for 2

Enter the Choice

4

Philosopher 1 is eating

Enter the Choice

1

Enter the philosopher no. who wants to eat

4

Entering

Eating

Enter the Choice

4

Philosopher 1 is eating

Philosopher 4 is eating

Enter the Choice

4

Philosopher 1 is eating

Philosopher 4 is eating

Enter the Choice

2

No. of philosopher who wants to stop eating

1

Philosopher 1 stops eating

Enter the Choice

2

No. of philosopher who wants to stop eating

4

Philosopher 4 stops eating

Enter the Choice

4

No one is eating

Enter the Choice

2

No. of philosopher who wants to stop eating

1

The philosopher 1 is not eating state

Enter the Choice

3

**7. READER WRITER PROBLEM:**

**Program:**

import java.io.\*;

import java.util.\*;

class monitor

{

static int nor=0;

static int now=0;

boolean busy=false;

synchronized void startread()

{

try

{

if(now!=0)

wait();

}

catch(Exception e)

{

System.out.println(e);

}

nor=nor+1;

}

synchronized void finishread()

{

nor=nor-1;

if(nor==0)

notify();

}

synchronized void startwrite()

{

try

{

if(busy||(nor>0))

wait();

}

catch(Exception e)

{

System.out.println(e);

}

now=now+1;

busy=true;

}

synchronized void finishwrite()

{

now=now-1;

busy=false;

if(now==0)

{

notifyAll();

}

}

}

class reader implements Runnable

{

monitor m=new monitor();

reader(monitor m1)

{

m=m1;

Thread t=new Thread(this);

t.start();

}

public void run()

{

m.startread();

System.out.println("Reading Started");

System.out.println("Reading Finished");

m.finishread();

}

}

class writer implements Runnable

{

monitor m=new monitor();

writer(monitor m1)

{

Thread t1=new Thread(this);

m=m1;

t1.start();

}

public void run()

{

m.startwrite();

System.out.println("Writing Started");

System.out.println("Writing Finished");

m.finishwrite();

}

}

class red

{

public static void main(String args[])

{

monitor s=new monitor();

new writer(s);

new reader(s);

new writer(s);

new reader(s);

}

}

**Output:**

Writing Started

Writing Finished

Writing Started

Writing Finished

Reading Started

Reading Started

Reading Finished

Reading Finished

**8. BANKER’S ALGORITHM:**

**Program:**

import java.io.\*;

import java.util.\*;

class deadlock

{

static int s=0;

public static void main(String args[])throws IOException

{

int p,c,i,j,n,x=0;

Scanner sn=new Scanner(System.in);

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

n=sn.nextInt();

int a[]=new int[n];

System.out.println("Enter the no. of instances of each Resources: ");

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

{

a[i]=sn.nextInt();

}

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

p=sn.nextInt();

int ord[]=new int[p];

int process[]=new int[p];

int rem[]=new int[p];

int finish[]=new int[p];

int ma[][]=new int[p][n];

int al[][]=new int[p][n];

int req[][]=new int[p][n];

System.out.println("Enter the max number of Resources required for each processes : ");

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

{

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

{

ma[i][j]=Integer.parseInt(in.readLine());

}

}

System.out.println("Enter the resources allocated for each processes : ");

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

{

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

{

al[i][j]=sn.nextInt();

}

}

System.out.println("Resources Left Out: ");

int z[]=new int[n];

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

{

z[i]=0;

}

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

{

for(j=0;j<p;j++)

{

z[i]=z[i]+al[j][i];

}

}

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

{

System.out.println("Resource "+(j+1)+" "+(a[j]-z[j]));

}

System.out.println("Amount of Resources Required for each processes : ");

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

{

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

{

req[i][j]=ma[i][j]-al[i][j];

}

}

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

{

System.out.print("\tResources "+(i+1));

}

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

{

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

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

{

System.out.print("\t\t"+req[i][j]);

}

}

int b=0,o=0;

while(o<p)

{

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

{

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

{

if(process[i]==0)

{

if((req[i][j])<=(a[j]-z[j]))

{

b=b+1;

}

else

break;

}

}

if(b==n)

{

process[i]=1;

ord[s]=i+1;

finish[s]=1;

s++;

o++;

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

{

//rem[j]=(a[j]-z[j])-req[i][j];

z[j]=z[j]+req[i][j];

}

}

b=0;

}

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

{

if(finish[i]==1)

{

int k=ord[i]-1;

finish[i]=0;

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

{

z[j]=z[j]-ma[k][j];

}

}

}

}

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

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

System.out.println(ord[i] + " Process completed");

}

}

**Output:**

Enter the no. of Resources:

3

Enter the no. of instances of each Resources:

10

5

7

Enter the no. of processes:

5

Enter the max number of Resources required for each processes :

7 5 3

3 2 2

9 0 2

2 2 2

4 3 3

Enter the resources allocated for each processes :

0 1 0

2 0 0

3 0 2

2 1 1

0 0 2

Resources Left Out:

Resource 1 3

Resource 2 3

Resource 3 2

Amount of Resources Required for each processes :

Resources 1 Resources 2 Resources 3

Processes 1 7 4 3

Processes 2 1 2 2

Processes 3 6 0 0

Processes 4 0 1 1

Processes 5 4 3 1

Execution Order

2 Process completed

4 Process completed

1 Process completed

3 Process completed

5 Process completed