**Practical Assignment 8:**

**Write a JAVA program to implement Banker's Algorithm**

import java.util.Scanner;

public class Bankers{

private int need[][],allocate[][],max[][],avail[][],np,nr;

private void input(){

Scanner sc=new Scanner(System.in);

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

np=sc.nextInt(); //no. of process

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

nr=sc.nextInt(); //no. of resources

need=new int[np][nr]; //initializing arrays

max=new int[np][nr];

allocate=new int[np][nr];

avail=new int[1][nr];

System.out.println("Enter allocation matrix -->");

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

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

allocate[i][j]=sc.nextInt(); //allocation matrix

System.out.println("Enter max matrix -->");

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

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

max[i][j]=sc.nextInt(); //max matrix

System.out.println("Enter available matrix -->");

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

avail[0][j]=sc.nextInt(); //available matrix

sc.close();

}

private int[][] calc\_need(){

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

for(int j=0;j<nr;j++) //calculating need matrix

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

return need;

}

private boolean check(int i){

//checking if all resources for ith process can be allocated

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

if(avail[0][j]<need[i][j])

return false;

return true;

}

public void isSafe(){

input();

calc\_need();

boolean done[]=new boolean[np];

int j=0;

while(j<np)

{

//until all process allocated

boolean allocated=false;

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

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

//trying to allocate

for(int k=0;k<nr;k++)

avail[0][k]=avail[0][k]-need[i][k]+max[i][k];

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

allocated=done[i]=true;

j++;

}

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

}

if(j==np) //if all processes are allocated

System.out.println("\nSafely allocated");

else

System.out.println("All proceess cant be allocated safely");

}

public static void main(String[] args) {

new Bankers().isSafe();

}

}

**OUTPUT:**

Enter no. of processes: 5

Enter no. of Resources : 3

Enter allocation matrix -->

0 1 0

2 0 0

3 0 2

2 1 1

0 0 2

Enter max matrix -->

7 5 3

3 2 2

9 0 2

2 2 2

4 3 3

Enter available matrix -->

3 3 2

Allocated process : 1

Allocated process : 3

Allocated process : 4

Allocated process : 0

Allocated process : 2

Safely allocated