```
package Banker;
import java.util.Scanner;

public class Bankers_Algo {


   public static void main(String args[])
   {
      int n,m,available[],max[][],allocation[][],need[][],work[];
      boolean finish[];
      Scanner reader = new Scanner(System.in);

      System.out.print("Enter number of processes: ");
      n = reader.nextInt();
      System.out.print("Enter number of resources: ");
      m = reader.nextInt();

      available = new int[m];
      max = new int[n][m];
      allocation = new int[n][m];
      need = new int[n][m];
      finish = new boolean[n];
      work = new int[m];

      System.out.println("Enter the available resources: ");
      for(int i=0;i<m;i++){
         available[i] = reader.nextInt();
         work[i] = available[i];
      }

      System.out.println("Enter the Max matrix: ");
      acceptInput(max,n,m);

      System.out.println("Enter the allocation matrix: ");
      acceptInput(allocation,n,m);

      for(int i=0;i<n;i++){
         for(int j=0;j<m;j++){
            need[i][j] = max[i][j] - allocation[i][j];
         }
      }

      for(int i=0;i<n;i++){
         finish[i] = false;
      }

      int safeseq[] = new int[n];
      int count = 0;

      while(count < n)
      {
         boolean flag = false;

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

```
        {

            int j;
            if(finish[i] == false)
            {
                for(j=0;j<m;j++)
                {
                    if(need[i][j]>work[j])
                    {
                        break;
                    }
                }
                if(j == m)
                {
                    safeseq[count++] = i;
                    finish[i] = true;
                    flag = true;

                    for(j=0;j<m;j++)
                    {
                        work[j] = work[j] + allocation[i][j];
                    }
                }
            }
        }
        if(flag == false)
        {
            break;
        }

    }
    if(count < n)
    {
        System.out.println("System is unsafe");
    }

    else
    {
        System.out.println("Safe sequence is: ");
        for(int i=0;i<n;i++)
        {
            System.out.print("P"+(safeseq[i] + 1 )+"\t");
        }
    }
}

public static void acceptInput(int matrix[][], int rows, int cols)
{
    Scanner reader = new Scanner(System.in);
    for(int i=0;i<rows;i++)
    {
        for(int j=0;j<cols;j++)
        {
            int a = reader.nextInt();
```

```
            matrix[i][j] = a;
        }

    }

  }
}
```

```
Enter number of processes: 5
Enter number of resources: 4
Enter the available resources:
1 5 2 0
Enter the Max matrix:
0 0 1 2
1 7 5 0
2 3 5 6
0 6 5 2
0 6 5 6
Enter the allocation matrix:
0 0 1 2
1 0 0 0
1 3 5 4
0 6 3 2
0 0 1 4
Safe sequence is:
P1 P3 P4 P5 P2
```

```
Enter number of processes: 5
Enter number of resources: 3
Enter the available resources:
3
3
2
Enter the Max matrix:
7 5 3
3 2 2
9 0 2
4 2 2
5 3 3
Enter the allocation matrix:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Safe sequence is:
P2 P4 P5 P1 P3
```