COEN 283 (Operating Systems)
Programming Assignment #3
Harshkumar Pandya

Q 41.
Program a simulation of the banker's algorithm. Your program should cycle through each of the bank clients asking for a request and evaluating whether it is safe or unsafe. Output a log of requests and decisions to a file.

Soln.
Here, I have implemented the Banker's Algorithm for one/multiple resources. In my code, we first input from the clients, the instances of resources they have already acquired and the amount of units they further need to finish their execution. 'Available Vector' is computed using the given information by the program. Then using the Available Vector and comparing it with Request matrix using the function 'ArrayCompare'.
If the requests can be fulfilled the Client is 'marked' using the CheckSum Vector, so the client need not be visited during further iterations. Also the order/sequence in which the clients are serviced is saved in the array 'Order'. Log of requests and the final decision (if the state is Safe/Unsafe) is saved in a text file - "Prof Assn3 - Banker's Algorithm.txt".

PROGRAM:
```
import java.util.*;
import java.io.FileOutputStream;
import java.io.PrintWriter;
public class Banker
{
	public static void main(String[] args) throws Exception
	{
		PrintWriter writer = new PrintWriter("Prog Assn3 - Banker's Algorithm.txt",
		"UTF-8");
		//Create file to save requests and decisions
		int Clients,Resources;
		//Initializing Variables
		Scanner in = new Scanner(System.in);
		System.out.println("Enter the number of Clients.");
		Clients = in.nextInt();
		System.out.println("Enter the number of Resources.");
		Resources = in.nextInt();
		System.out.println("Maxi Sources Available - Existing Resource Vector .");
		//Input-Maximum Resources
		int[] Exist = new int[Resources];
		int[] Available = new int[Resources];
		int[][] Current = new int[Clients][Resources];
		int[][] Request = new int[Clients][Resources];
		int[] CheckSum = new int[Clients];
```

```
//Bookkeeping Vector to mark visited processes
int[][] Request1 = new int[Clients][Resources];
int[] Available1 = new int[Resources];
int[] Order=new int[Clients];
int temp;
int c=0;
//Input Max Resources

for(int i=0;i<Resources;i++)
{
        System.out.println("Enter the instances available for - Resource
"+(i+1));
        Exist[i] = in.nextInt();
}
//Input units of already acquired resources and further requests
for(int i=0;i<Clients;i++)
{
        for(int j=0;j<Resources;j++)
        {
                System.out.println("For - Client "+(i+1));
                System.out.println("Enter the instances of Resource "+(j+1) +"
        acquired.");
                Current[i][j] = in.nextInt();
                System.out.println("Enter the instances of Resource "+(j+1) +"
        requested.");
                Request[i][j] = in.nextInt();
                Request1[i][j]=Request[i][j];
        }
}
//Computing Available Vector
//Available=Existing-Current
System.out.print("Hence the Available Vector is : ");
for(int i=0;i<Resources;i++)
{
        temp=0;
        for(int j=0;j<Clients;j++)
        temp+=Current[j][i];
        Available[i]=Exist[i]-temp;
        Available1[i]=Available[i];
        System.out.print(" " + Available[i] +" ");
}
System.out.println("");
boolean a;
//To poll all Clients and see if it is safe
for(int i=0;i<Clients;i++)
{
```

```
for(int j=0;j<Clients;j++)
{
        if (CheckSum[j]!=99)
        {
                a=ArrayCompare(Request[j],Available);
                if(a)
                {
                        CheckSum[j]=99;
                        for(int k=0;k<Resources;k++)
                        {
                                Available[k]+=Current[j][k];
                                Request[j][k]=0;
                        }
                        //To record the order of the Clients are serviced
                        Order[c]=j+1;
                        c++;
                }
        }
}
}
//To verify if it is a SAFE STATE
boolean empty = true;
for (int i=0; i<Clients; i++)
{
        for(int j=0;j<Resources;j++)

        {
                if (Request[i][j]!=0)
                {
                        empty = false;
                        break;
                }
        }
}
//Writing Requests and Output Descision in file
try
{
        for(int i=0;i<Clients;i++)
        {
                writer.println("For Client - " + (i+1));
                for(int j=0;j<Resources;j++)
                {
                        writer.println("Acquired : " + Current[i][j] + " instances
                        of Resource " + (j+1));
                        writer.println("Requested : " + Request1[i][j] + "
                        instances of Resource " + (j+1));
                }
```

```
                        writer.println("");
                    }
                    writer.println("Available instances of ");
                    for(int i=0;i<Resources;i++)
                    {
                        writer.println("Resource : "+ (i+1) + " - "+ Available1[i]);
                    }
                    writer.println("");
                    if(empty)
                    {
                        writer.println("Order in which the Clients are serviced ");
                        for(int i=0;i<Clients;i++)
                        {
                            writer.print(" " + Order[i] + " ");
                        }
                        writer.println("");
                        writer.println("Safe State");
                    }
                    else
                    {
                        writer.println("UnSafe State");
                    }
                    writer.close();
                }
                catch(Exception e1)
                {
                System.out.println("Error during reading/writing");
                }
            }
            //Function to compare Request and Available Vectors
            public static boolean ArrayCompare(int[] a, int[] a2)
            {
                int length = a.length;
                for (int i=0; i<length; i++) // compare array values
                if (a[i] > a2[i])
                return false;
                return true;
            }
        }
```

OUTPUT

```
Enter the number of Clients.
5
Enter the number of Resources.
4
Maximum Sources Available - Existing Resource Vector .
Enter the instances available for - Resource 1
6
Enter the instances available for - Resource 2
3
Enter the instances available for - Resource 3
4
Enter the instances available for - Resource 4
2
For - Client 1
Enter the instances of Resource 1 acquired.
3
Enter the instances of Resource 1 requested.
1
For - Client 1
Enter the instances of Resource 2 acquired.
0
Enter the instances of Resource 2 requested.
1
For - Client 1
Enter the instances of Resource 3 acquired.
1
Enter the instances of Resource 3 requested.
0
For - Client 1
Enter the instances of Resource 4 acquired.
1
Enter the instances of Resource 4 requested.
0
For - Client 2
Enter the instances of Resource 1 acquired.
0
Enter the instances of Resource 1 requested.
0
For - Client 2
Enter the instances of Resource 2 acquired.
1
Enter the instances of Resource 2 requested.
```

```
1
For - Client 2
Enter the instances of Resource 3 acquired.
0
Enter the instances of Resource 3 requested.
1
For - Client 2
Enter the instances of Resource 4 acquired.
0
Enter the instances of Resource 4 requested.
2
For - Client 3
Enter the instances of Resource 1 acquired.
1
Enter the instances of Resource 1 requested.
3
For - Client 3
Enter the instances of Resource 2 acquired.
1
Enter the instances of Resource 2 requested.
1
For - Client 3
Enter the instances of Resource 3 acquired.
1
Enter the instances of Resource 3 requested.
0
For - Client 3
Enter the instances of Resource 4 acquired.
0
Enter the instances of Resource 4 requested.
0
For - Client 4
Enter the instances of Resource 1 acquired.
1
Enter the instances of Resource 1 requested.
0
For - Client 4
Enter the instances of Resource 2 acquired.
1
Enter the instances of Resource 2 requested.
0
For - Client 4
Enter the instances of Resource 3 acquired.
0
Enter the instances of Resource 3 requested.
1
For - Client 4
```

```
Enter the instances of Resource 4 acquired.
1
Enter the instances of Resource 4 requested.
0
For - Client 5
Enter the instances of Resource 1 acquired.
0

Enter the instances of Resource 1 requested.
2
For - Client 5
Enter the instances of Resource 2 acquired.
0
Enter the instances of Resource 2 requested.
1
For - Client 5
Enter the instances of Resource 3 acquired.
0
Enter the instances of Resource 3 requested.
1
For - Client 5
Enter the instances of Resource 4 acquired.
0
Enter the instances of Resource 4 requested.
0
Hence the Available Vector is :
1 0 2 0
```

Contents of the File Containing Output – **Prog Assn3 - Banker's Algorithm.txt**

```
For Client - 1
Acquired : 3 instances of Resource 1
Requested : 1 instances of Resource 1
Acquired : 0 instances of Resource 2
Requested : 1 instances of Resource 2
Acquired : 1 instances of Resource 3
Requested : 0 instances of Resource 3
Acquired : 1 instances of Resource 4
Requested : 0 instances of Resource 4
For Client - 2
Acquired : 0 instances of Resource 1
Requested : 0 instances of Resource 1
Acquired : 1 instances of Resource 2
Requested : 1 instances of Resource 2
Acquired : 0 instances of Resource 3
Requested : 1 instances of Resource 3
```

```
Acquired : 0 instances of Resource 4
Requested : 2 instances of Resource 4
For Client - 3
Acquired : 1 instances of Resource 1
Requested : 3 instances of Resource 1
Acquired : 1 instances of Resource 2
Requested : 1 instances of Resource 2
Acquired : 1 instances of Resource 3
Requested : 0 instances of Resource 3
Acquired : 0 instances of Resource 4
Requested : 0 instances of Resource 4

For Client - 4
Acquired : 1 instances of Resource 1
Requested : 0 instances of Resource 1
Acquired : 1 instances of Resource 2
Requested : 0 instances of Resource 2
Acquired : 0 instances of Resource 3
Requested : 1 instances of Resource 3
Acquired : 1 instances of Resource 4
Requested : 0 instances of Resource 4
For Client - 5
Acquired : 0 instances of Resource 1
Requested : 2 instances of Resource 1
Acquired : 0 instances of Resource 2
Requested : 1 instances of Resource 2
Acquired : 0 instances of Resource 3
Requested : 1 instances of Resource 3
Acquired : 0 instances of Resource 4
Requested : 0 instances of Resource 4
Available instances of
Resource : 1 - 1
Resource : 2 - 0
Resource : 3 - 2
Resource : 4 - 0
Order in which the Clients are serviced
4 5 1 2 3
Safe State
```