BITS- Assignment 1- Distributed Computing (S1-20_SSZG526)

Sl No	Name	Registration ID	Course
1	Rahul K Yadav	2020MT12284	M. Tech Data Analytics

Work Integrated Learning Programmes

BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Subject ID: SSZG526

20 Nov 2020

Table of Contents

1. Assignment and Description

Assignment: Write a program to implement the Chandy–Misra–Haas Algorithm for the OR model

OR Model Description:

- a process can make a request for multiple resources simultaneously
- request is satisfied if any one of the requested resources is granted
- requested resources may exist at different locations
- if all requests in the WFG are OR requests, then the nodes are called OR nodes
- presence of a knot indicates a deadlock
- with every blocked process, there is an associated set of processes called *dependent set*
- *a blocked process becomes active on receiving a grant message from any one of the processes in its dependent set*
- *a process is permanently blocked if it never receives a grant message from any of the processes in its dependent set*
- deadlock detection in the OR model is equivalent to finding knots in the graph
- note: there can be a deadlocked process that is not a part of a knot
- in an OR model, a blocked process P is deadlocked if it is either in a knot or it can only reach processes on a knot

2. Algorithm Pseudo Code

Initiate a deadlock detection for a blocked process P_i:

- send query(i, i, j) to all processes P_j in the dependent set DS_i of P_i
- $num_i(i) = |DS_i|$
- $wait_i(i)$ = true

When a blocked process P_k receives a query(i, j, k):

- ***if*** this is the **engaging query** for process P_i then

send query(i, k, m) to all P_m in its dependent set DS_k

$num_k(i) = |DS_k|$

$wait_k(i)$ = true

- ***else*** if $wait_k(i)$ = true then

send a reply(i, k, j) to P_j

When a process P_k receives a reply(i, j, k):

- ***if*** $wait_k(i)$ = true then

$num_k(i) = num_k(i) - 1$

- ***if*** $num_k(i) = 0$ then

if i = k then **declare a deadlock**

else send reply(i, k, m) to the process P_m which sent

the engaging query

3. Pseudo Code to JAVA Code

```
package chandymisrahass_or_algorithm;

import java.util.*;
class QueryMessage
{


    public  int initiator=0;
    public  int sender=0;
    public  int receiver=0;
    public QueryMessage(int i,int j,int k)
    {
        initiator=i;
        sender=j;
        receiver=k;

    }
    public String toString()
    {
        return "("+initiator+","+sender+","+receiver+")";
    }
 }

class ReplyMessage
{


    public  int initiator=0;
    public  int sender=0;
    public  int receiver=0;
    public ReplyMessage(int i,int j,int k)
    {
```

```
        initiator=i;
        sender=j;
        receiver=k;

    }
    public String toString()
    {
        return "("+initiator+","+sender+","+receiver+")";
    }
 }
public class chandymisrahass_or {
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int graph[][];
        boolean isDeadlock=false;
        boolean wait;
        System.out.println("Enter the number of processes - Please enter Integer
value::");
        int n=sc.nextInt();

        graph=new int[n][n];
        System.out.println("Enter the wait for graph: ");
        System.out.println("Since there are: " + n + "Processes so enter : " +n*n + "
times::");
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n;j++)
            {
                graph[i][j]=sc.nextInt();

            }
        }

        System.out.println("the wait for graph is:");
        new chandymisrahass_or().Display(graph);
        System.out.println("Enter the process initiating the diffusion computation");
        int init=sc.nextInt();

        //Initiate Probe
        System.out.println("Initiating deadlock detection ");
        List<QueryMessage> mess_list=new ArrayList<QueryMessage>();
        int num=0;

        List<ReplyMessage> mess_list1=new ArrayList<ReplyMessage>();
        int num1=0;

        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n;j++)
            {
                if(graph[i][j]==1)
                {
                    QueryMessage m=new QueryMessage(init,i,j);
                    mess_list.add(m);
```

```
                    num+=1;
                    wait = true;
                }
                else {
                    ReplyMessage m=new ReplyMessage(i,init,j);
                    mess_list1.add(m);

                }
            }
        }
        System.out.println(mess_list);
        if (wait = true)
             num-=1;

        for(int i=0;i<num;i++)
        {
            for(int j=0;j<num;j++)
            {

             if(mess_list.get(i).initiator==mess_list.get(j).receiver)
                    isDeadlock=true;
            }
        }
        sc.close(); // Close scanner
        if(isDeadlock)
            System.out.println("The Deadlock has been detected..."); //Deadlock
detected
        else
            System.out.println("No Deadlock has been detected...");


       }

    void Display(int[][] mat)
    {
        int n=mat[0].length;
        int m=mat.length;
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<n;j++)
            {
                System.out.print(mat[i][j]+" ");
            }
            System.out.println();
        }
    }
}
```

4. Process Execution

Case 1 : Deadlock is detected

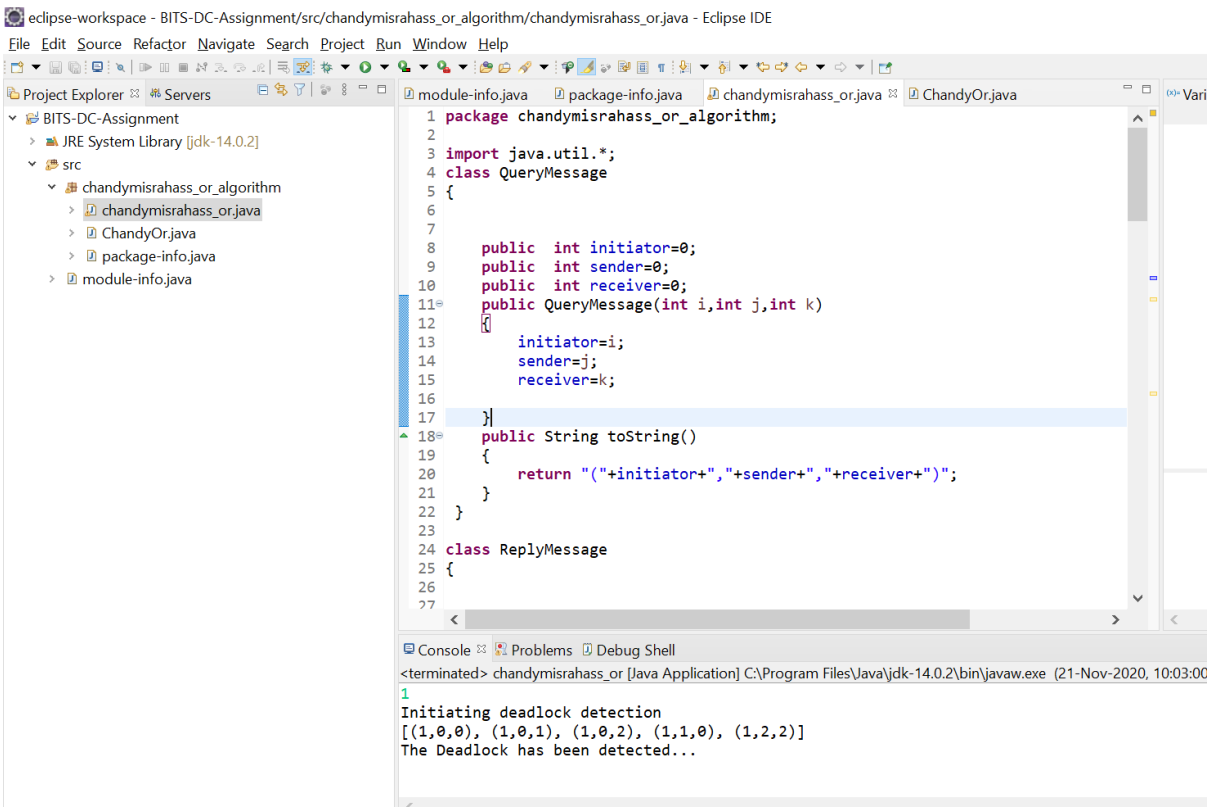
```
Enter the number of processes
```

```
 4
Enter the wait for graph:

1
1
1
0
0
0
0
1
0
0
1
1
1
0
0
0
the wait for graph is:
1 1 1 0
0 0 0 1
0 0 1 1
1 0 0 0
Enter the process initiating probe
3
Initiating probe…
[(3,0,0), (3,0,1), (3,0,2), (3,1,3), (3,2,2), (3,2,3), (3,3,0)]
The Deadlock has been detected…
```

Case 2: No Deadlock

```
Enter the number of processes
3
Enter the wait for graph:
0
1
0
0
0
1
1
0
0
the wait for graph is:
0 1 0
0 0 1
1 0 0
Enter the process initiating probe
3
Initiating probe...
[(3,0,1), (3,1,2), (3,2,0)]
No Deadlock has been detected...
```



On Command prompt Execution using JAR executable file it will look like this

```
C:\Program Files\Java\jdk-14.0.2\bin> java -jar C:\Users\rahul\cmh-Or_Algorithm.jar
Enter the number of processes - only Integer values
4
Enter the wait for graph:
please enter wait upto:16times
1
0
0
1
1
0
0
0
0
0
1
1
1
0
0
0
the wait for graph is:
1 0 0 1
1 0 0 0
0 0 1 1
1 0 0 0
Enter the process initiating probe
2
Initiating probe...
[(2,0,0), (2,0,3), (2,1,0), (2,2,2), (2,2,3), (2,3,0)]
The Deadlock has been detected...

C:\Program Files\Java\jdk-14.0.2\bin>
```

5. Readme file, JAVA Source Code File, executable JAR file

Below are the Readme file, Source Code and JAR file to download.


Readme.txt


cmh-Or_Algorithm.jar


chandymisrahass_or.java

Note: The program has some **limitations** as it is not **handling the exceptions** as idea is to only demonstrate the Deadlock detection, **user has to provide the values in Integers and within the boundaries as per the Messages instructions appearing during execution**