**Distributed Computing – Assignment 1**

Written By: Shruti Nagendra

BITS ID: 2020MT12072

Email: 2020MT12072@wilp.bits-pilani.ac.in

M-Tech, System Software

Contents

[Introduction 3](#_Toc56629980)

[System Requirements 3](#_Toc56629981)

[Solution Description 3](#_Toc56629982)

[Class Description 3](#_Toc56629983)

[Running the program 3](#_Toc56629984)

[Inputs 4](#_Toc56629985)

[Output 5](#_Toc56629986)

[Known Issues 6](#_Toc56629987)

# Introduction

This document explains the process to run the code that demonstrates Chandy-Haas-Misra OR Model.

# System Requirements

Operating System – Windows 10/iOS/ Linux – Latest Versions

Java Version – 1.6 or higher should be installed. Follow this [link](https://www.java.com/en/download/help/index_installing.html) to install Java and set it up for your OS.

# Solution Description

Implementation Language – Java

### Class Description

1. ChandyMisraHaasOrDemo.java – Main class. Begin program execution from here. This class gathers user input on number of processes, dependent processes for each of the process and initializes the deadlock detection algorithm
2. Process.java – Class that represents a distributed process. It has all the data structures and the data needed to run the algorithm.
3. Message.java – Class that represents the query and reply messages that are passed between the processes.
4. DistributedProcess.java – Interface that contains the definition of methods implemented by Process.java class.
5. MessageType.java – Enumeration type that has definitions for the message types. This is a class that contains constant values.
6. WaitForGraph.java – Class the represents the entire wait for graph.

# Running the program

There are 3 ways to run the program –

1. Using Batch/Shell Script to run the ChandyMisraHaasORAlgorithm.jar file.
   1. On Windows Systems – Ensure that Java environment is setup correctly. Once that is done, double click RunDeadlockDetection.bat
   2. On Mac/Linux Systems – Ensure that Java environment is setup correctly. Once that is done, open the RunDeadlockDetection.sh and edit the path to the place where the jar is located. Run the shell script on the terminal.
2. Run the jar file directly – Ensure that Java environment is setup correctly. Open a cmd prompt/terminal and type the following command –

java -jar ChandyMisraHaasORAlgorithm.jar and press Enter.

1. Using source code – Place the src folder at any place in the computer. Open a command prompt/terminal and navigate inside src folder. Compile all the files using the following command -

javac \*.java

Run the program using the command –

java ChandyMisraHaasOrDemo

Alternatively, you can use an IDE like Eclipse or IntelliJ, import the source into a Java Project and run the main class from the IDE.

# Inputs

1. Enter the number of processes – Total number of processes in the wait for graph. Please enter a number only. On entering a non-numeric character, an exception is thrown.
2. Enter dependent process – Each process can have ‘n’ dependent processes, where ‘n’ is the total number of processes in the WFG, as entered in the previous step. A separate prompt will be issued ‘n’ times for each of the ‘n’ processes. Please enter the dependent processes one by one.

Allowed set of values is from -1 and from 0 to n-1. If you try to enter any value greater than 1, then you will be prompted to enter the correct value. This check will take place every time a wrong value is entered.

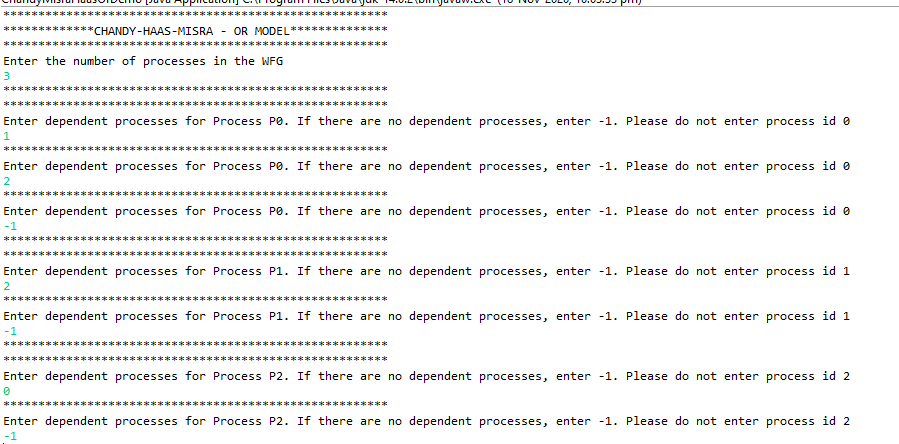
In case you do not wish to add any dependent

You are not allowed to enter the same value as the current process. For example, if you are entering dependent processes for process id 0, then 0 is not a valid dependent process.

Please enter only numeric characters. Entering non-numeric characters will result in an unhandled exception.

1. Enter the Process Id that should initiate deadlock detection – This process will be the first one to initiate the deadlock detection algorithm.

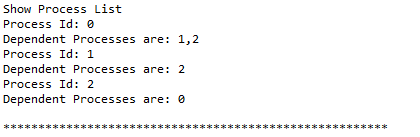
Sample Inputs



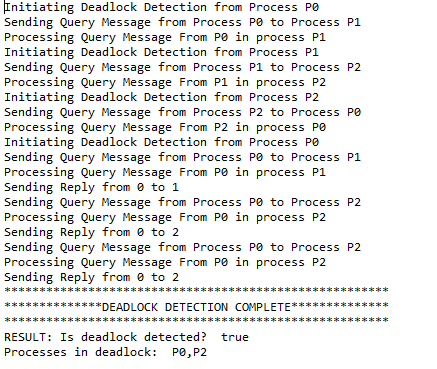


# Output

Once all the dependencies are added, a list showing the dependencies are added.



The output of the deadlock detection algorithm is as shown below –



# Known Issues

1. Entering non-numeric input results in an exception. Always enter numeric input to obtain correct output.