**Lab 2**

RMI Implementation

1. Objective

The objective of this lab is to implement a RMI system.

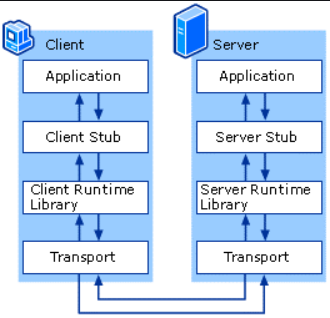
1. Software Used

The Java Development Kit (JDK) was used as a compiler, notepad was used to write the code and the windows command prompt was used to execute the code.

1. Introduction

Remote Procedure Call (RPC) is a protocol that one program can use to request a service from a program located in another computer on a network without having to understand the network's details. RPC is used to call other processes on the remote systems like a local system. A procedure call is also sometimes known as a function call or a subroutine call.

RPC has the following architecture



1. Code Implementation

Client code:

import java.rmi.Naming;

public class lab\_2\_RMI\_CLient {

public static void main(String args[]) throws Exception {

lab\_2\_RMI\_interface obj = (lab\_2\_RMI\_interface)Naming.lookup("//localhost/lab\_2\_RMI\_Server");

System.out.println(obj.getMessage());

}

}

Server Code:

import java.rmi.Naming;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.server.UnicastRemoteObject;

public class lab\_2\_RMI\_Server extends UnicastRemoteObject implements lab\_2\_RMI\_interface {

public static final String MESSAGE = "Hello World";

public lab\_2\_RMI\_Server() throws RemoteException {

super(0); }

public String getMessage() {

return MESSAGE;

}

public static void main(String args[]) throws Exception {

System.out.println("RMI server started");

try {

LocateRegistry.createRegistry(1099);

System.out.println("java RMI registry created.");

} catch (RemoteException e) { System.out.println("java RMI registry already exists.");

}

lab\_2\_RMI\_Server obj = new lab\_2\_RMI\_Server();

Naming.rebind("//localhost/lab\_2\_RMI\_Server", obj);

System.out.println("PeerServer bound in registry");

}

}

Interface Code:

import java.rmi.Remote;

import java.rmi.RemoteException;

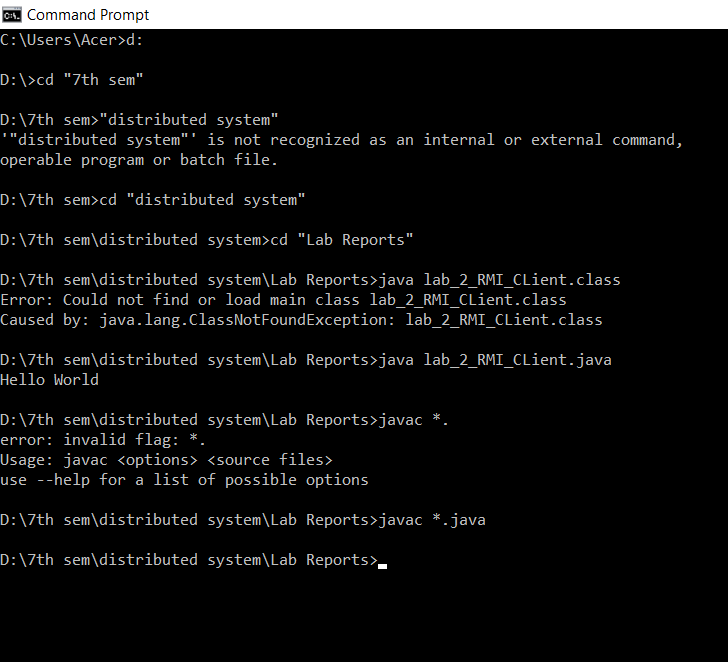
public interface lab\_2\_RMI\_interface extends Remote {

public String getMessage() throws RemoteException;

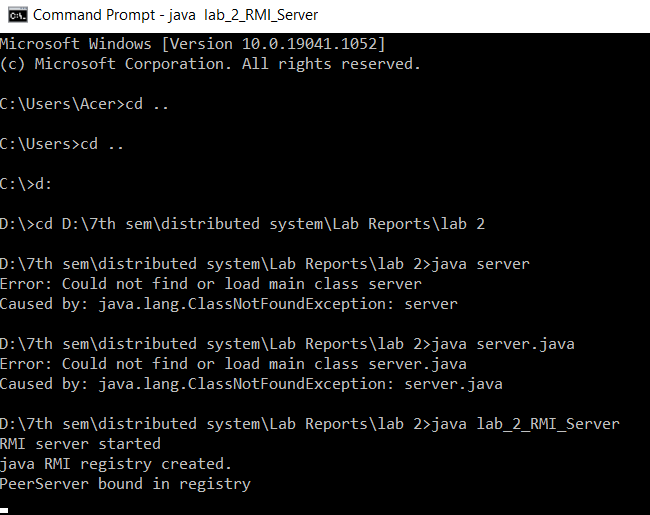
}

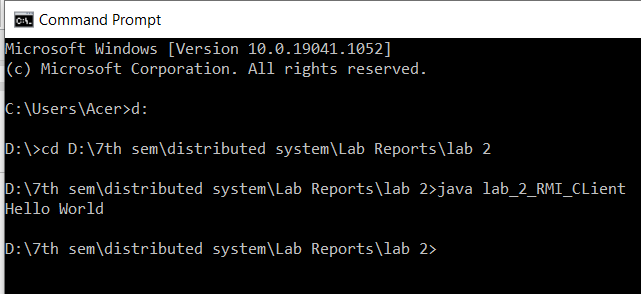
1. Result

All files compiled:



Executions:





6. Discussion

We implemented a simple RPC using java and observed the following limitations:

• It is not a standard as it can be implemented in different ways.

• There is no flexibility in RPC for hardware architecture. It is only interaction based.

• Context switching increases scheduling costs

1. Conclusion

Hence, in this lab we created a client and a server using java and interfaced between them using an interface.