ASSIGNMENT-2

Aim and Objective:

To develop any distributed application through implementing client-server communication programs based on Java RMI .

Tools / Environment:

JavaProgrammingEnvironment, JDK1.8or higher, MPILibrary(mpi.jar) , MPJExpress(mpj.jar)

CODES

server.java

```
package As2;
import java.io.*;
import java.net.*;

publicclass server {
	publicstaticvoid main(String[] args) {
		try
			{
			@SuppressWarnings("resource")
			ServerSocket server = new ServerSocket(3001);
			Socket s = server.accept();
			System.out.println("Connected To Server for Message 
Passing from DYPCOE...");
			DataOutputStream dos = new 
DataOutputStream(s.getOutputStream());
			dos.writeUTF(" Hi Ankur ...Welcome  to socket");
			} catch (Exception e) {
		}
	}
}
```

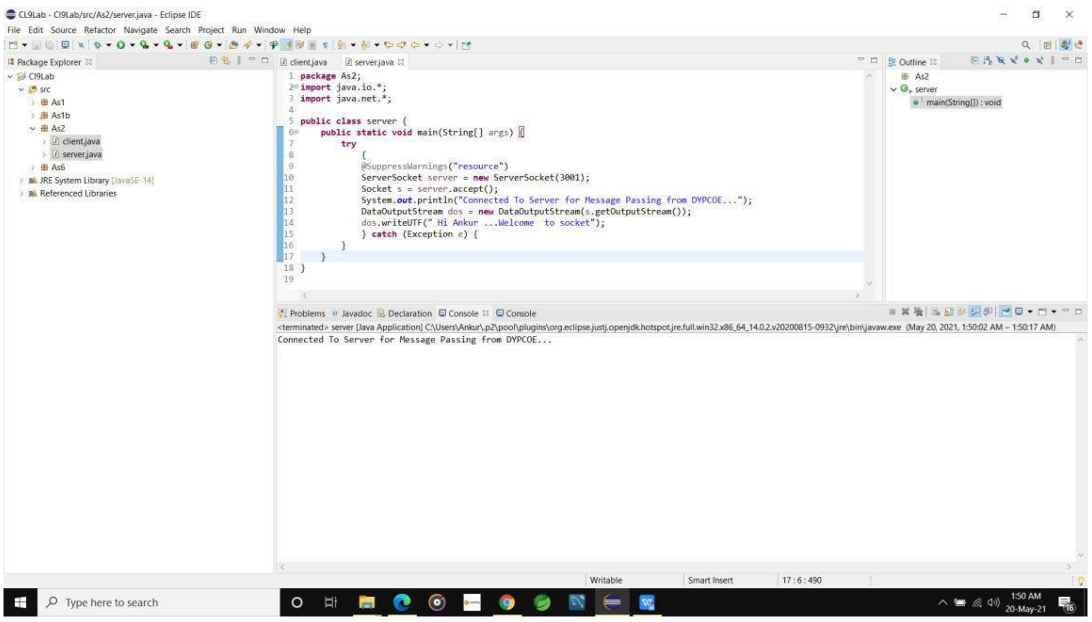
client.java

```
package As2;
import java.io.DataInput;
import java.io.DataInputStream;
import java.net.Socket;

publicclass client {

	publicstaticvoid main(String[] args) {
		try
			{
```

```
                    @SuppressWarnings("resource")
                    Socket s = new Socket("127.0.0.1", 3001);
                   DataInput dis = new DataInputStream(s.getInputStream());
                    String msg = dis.readUTF();
                    System.out.println(msg);
              } catch (Exception e) {
              }
       }
}
```

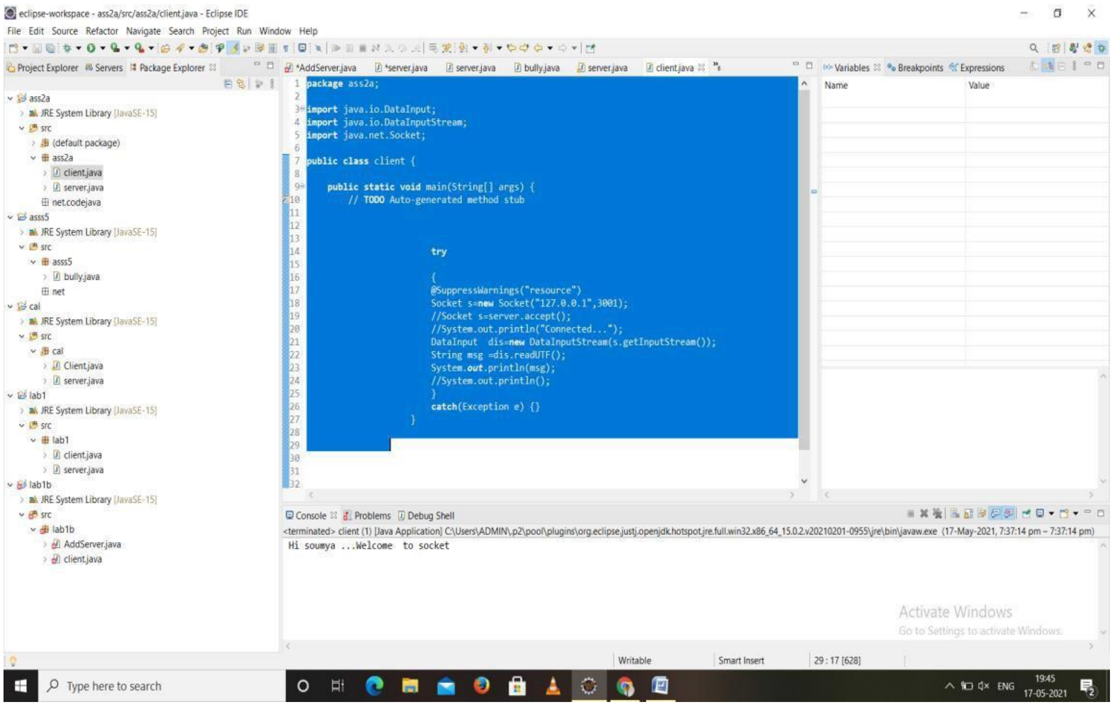
Outputs

Using Eclipse :



```
package As2;
import java.io.*;
import java.net.*;

public class server {
    public static void main(String[] args) {
        try
            {
            @SuppressWarnings("resource")
            ServerSocket server = new ServerSocket(3001);
            Socket s = server.accept();
            System.out.println("Connected To Server for Message Passing from DYPCOE...");
            DataOutputStream dos = new DataOutputStream(s.getOutputStream());
            dos.writeUTF(" Hi Ankur ...Welcome  to socket");
            } catch (Exception e) {
        }
    }
}
```

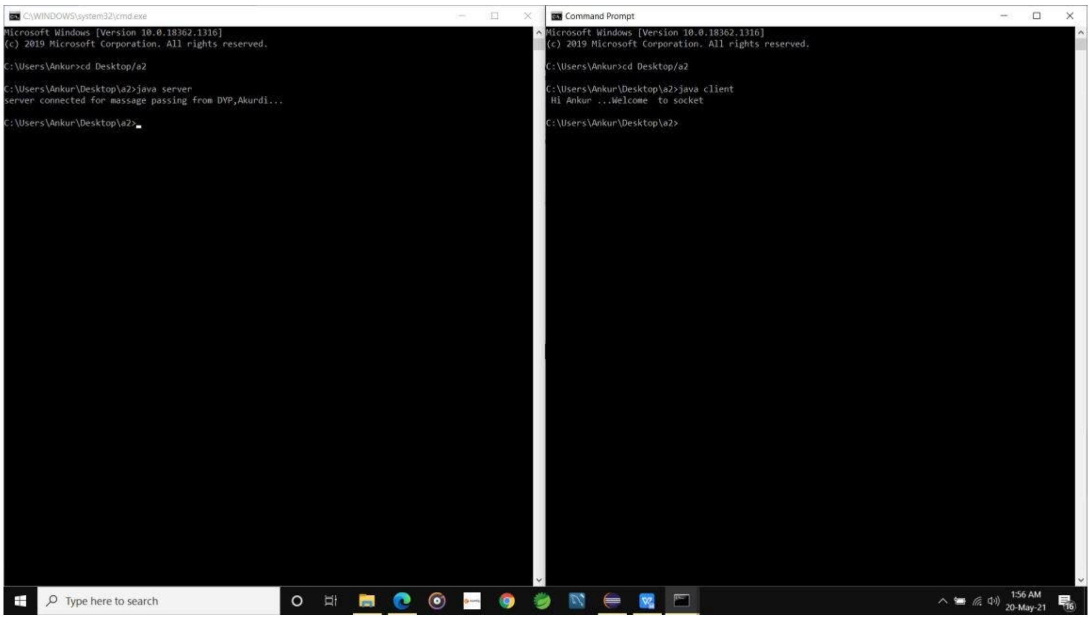
```
<terminated> server [Java Application] C:\Users\Ankur\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20200815-0932\jre\bin\javaw.exe (May 20, 2021, 1:50:02 AM – 1:50:17 AM)
Connected To Server for Message Passing from DYPCOE...
```



```java
package ass2a;

import java.io.DataInput;
import java.io.DataInputStream;
import java.net.Socket;

public class client {

    public static void main(String[] args) {
        // TODO Auto-generated method stub



                    try

                    {
                    @SuppressWarnings("resource")
                    Socket s=new Socket("127.0.0.1",3001);
                    //Socket s=server.accept();
                    //System.out.println("Connected...");
                    DataInput  dis=new DataInputStream(s.getInputStream());
                    String msg =dis.readUTF();
                    System.out.println(msg);
                    //System.out.println();
                    }
                    catch(Exception e) {}
                }
```

```
<terminated> client (1) [Java Application] C:\Users\ADMIN\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0955\jre\bin\javaw.exe (17-May-2021, 7:37:14 pm – 7:37:14 pm)
 Hi soumya ...Welcome  to socket
```

Using JAVA CMD:



```
Microsoft Windows [Version 10.0.18362.1316]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Ankur>cd Desktop/a2

C:\Users\Ankur\Desktop\a2>java server
server connected for massage passing from DYP,Akurdi...

C:\Users\Ankur\Desktop\a2>
```

```
Microsoft Windows [Version 10.0.18362.1316]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Ankur>cd Desktop/a2

C:\Users\Ankur\Desktop\a2>java client
 Hi Ankur ...Welcome  to socket

C:\Users\Ankur\Desktop\a2>
```

Related Theory:

Message passing is a popularly ren owned mechanism to implement parallel is min applications; it is also called MPI. The MPI interface for Java has a technique for identifying the user and helping in lower startup overhead. It also helps in collective communication and could be executed on both **shared memory and distributed systems**. MPJ is a familiar Java API for MPI implementation. Mpi Java is the near flexible Java binding for MPJ standards.
Currently developers can produce more efficient and effective parallel applications using message passing.

Abasic prerequisite for message passing is a good communication API. Java comes with various readymade packages for communication, notably an interface to BSD sockets, and the Remote Method Invocation(RMI) mechanism. The parallel computing world is mainly concerned with
`symmetric' communication, occurring in group so interacting peers. This symmetric model of communication is captured in the successful Message Passing Interface standard(MPI).

Message-PassingInterface Basics:

Every MPI program must contain the preprocessor directive:

```
#include<mpi.h>
```

The `mpi.h` file contains the definitions and declarations necessary for compiling an MPI program.

`MPI_Init` initializes the execution environment for MPI. It is a "sharenothing" modality in which the outcome of any one of thec oncurrent processes can in no way be influenced by the intermediate results of any of the other processes. Command has to be called before any other MPI call is made, and it is an error to call it more than a single time within the program. **`MPI_Finalize`** cleans up all the extra neousmess that was first put into place by `MPI_Init`.
Multiple Data (SPMD). *The multicore configuration is meant for users who plan to write and execute parallel Java applications using MPJ Express on their desktops or laptops which contains shared memory and multicore processors. In this configuration, users can write their message passing parallel application using MPJ Express and it will be ported automatically on multicore processors. We except that users can first develop applications on their laptops and desktops using multicore configuration, and then take the same code to distributed memory platforms*

Conclusion:

There has been a large amount of interest in parallel programming using Java.mpj is an MPI binding with Java along with the support for multicore architecture so that user can develop the code on it's own laptop or desktop. This is an effort to develop and run parallel programs according to MPI standard.