**Practical 2**

**Aim:**

2.1 Write RMI application where client supplies two numbers and server response by summing it. (Use: Scanner class).

**Description:**

RMI (Remote Method Invocation)

The **RMI** (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.

The RMI provides remote communication between the applications using two objects *stub* and *skeleton*.

Understanding stub and skeleton

RMI uses stub and skeleton object for communication with the remote object.

A **remote object** is an object whose method can be invoked from another JVM. Let's understand the stub and skeleton objects:

stub

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

1. It initiates a connection with remote Virtual Machine (JVM),
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),
3. It waits for the result
4. It reads (unmarshals) the return value or exception, and
5. It finally, returns the value to the caller.

skeleton

The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

1. It reads the parameter for the remote method
2. It invokes the method on the actual remote object, and
3. It writes and transmits (marshals) the result to the caller.

In the Java 2 SDK, an stub protocol was introduced that eliminates the need for skeletons.



**Program code:**

**RemoteInterface:-**

**import** java**.**rmi**.\*;**

public interface RemoteInterface **extends** Remote

**{**

public int add**(**int x**,**int y**)throws** Exception**;**

**}**

**ServerImplements:-**

**import** java**.**rmi**.\*;**

**import** java**.**rmi**.**server**.\*;**

**import** java**.**lang**.**String**;**

public class ServerImplements **extends**

UnicastRemoteObject **implements** RemoteInterface

**{**

public ServerImplements**()throws** Exception

**{**

**super();**

**}**

public int add**(**int x**,**int y**)**

**{**

**return** **(**x**+**y**);**

**}**

**}**

**Server:-**

**import** java**.**rmi**.\*;**

**import** java**.**net**.\*;**

public class Server

**{**

public static void main**(**String args**[])**

**{**

**try**

**{**

ServerImplements s**=new** ServerImplements**();**

Naming**.**rebind**(**"SERVICE"**,**s**);**

System**.**out**.**println**(**"Server Started "**);**

**}**

**catch(**Exception e**)**

**{**

System**.**out**.**println**(**e**.**getMessage**());**

**}**

**}**

**}**

**Client:-**

**import** java**.**rmi**.\*;**

**import** java**.**io**.\*;**

public class Client

**{**

public static void main**(**String args**[])**

**{**

**try**

**{**

String ip**=**"rmi://localhost/SERVICE"**;**

RemoteInterface s**=(**RemoteInterface**)**Naming**.**lookup**(**ip**);**

System**.**out**.**println**(**"130050131070"**)**

System**.**out**.**println**(**"sum: "**+** s**.**add**(**1**,**3**));**

**}**

**catch(**Exception e**)**

**{**

System**.**out**.**println**(**e**.**getMessage**());**

e**.**printStackTrace**();**

**}**

**}**

**}**

**Input Output:**





