ITL 3 PRACTICAL - 08

**AIM**

**Perform multiplication of two numbers using RMI.**

# 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:**

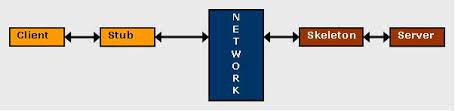
1. **It initiates a connection with remote Virtual Machine (JVM),**
2. **It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),**
3. **It waits for the result**
4. **It reads (unmarshals) the return value or exception, and**
5. **It finally, returns the value to the caller.**

### 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.**

****

## Understanding requirements for the distributed applications

**If any application performs these tasks, it can be distributed application.**

1. **The application need to locate the remote method**
2. **It need to provide the communication with the remote objects, and**
3. **The application need to load the class definitions for the objects.**

**The RMI application have all these features, so it is called the distributed application.**

**Server Interface**

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

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

**public interface servinterface extends Remote**

**{**

**public double mul(double a,double b) throws RemoteException;**

**}**

**Server Implementation**

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

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

**public class servimpl extends UnicastRemoteObject implements servinterface**

**{**

**public servimpl() throws RemoteException**

**{**

**super();**

**}**

**public double mul(double a,double b)**

**{**

**return(a\*b);**

**}**

**}**

**Server**

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

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

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

**public class server extends servimpl**

**{**

**public server() throws RemoteException**

**{**

**}**

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

**{**

**try**

**{**

**Registry reg = LocateRegistry.createRegistry(9999);**

**servimpl s = new servimpl();**

**reg.rebind("Server",s);**

**System.out.println("Server is ready!");**

**}**

**catch(Exception e)**

**{**

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

**}**

**}**

**}**

**Client**

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

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

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

**public class client**

**{**

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

**{**

**try**

**{**

**int a = Integer.parseInt(args[0]);**

**int b = Integer.parseInt(args[1]);**

**Registry reg=LocateRegistry.getRegistry("127.0.0.1",9999);**

**servinterface i = (servinterface)reg.lookup("Server");**

**System.out.println("First number: "+a);**

**System.out.println("Second number: "+b);**

**System.out.println("Mul:\t"+i.mul(a,b));**

**}**

**catch(Exception e)**

**{**

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

**}**

**}**

**}**

**Output:   Compiling and starting services:**

**Text

Description automatically generated**

**Server :**

**Text

Description automatically generated**

**Client:**

Text

Description automatically generated