***Name:* ……………………………….**

Department. : *Computer Engineering*

Class & Semester: *B.E (Final Year), Sem VIII*

Subject: *Distributed Computing Lab (DCL)*

Expt. No. 03

Title: *Write a program to demonstrate the steps of Remote Method Invocation (RMI) application using java.*

Date:

Subject In-charge Sign:

……………………………..

Experiment No. 03

AIM: Write a program to demonstrate the steps of Remote Method Invocation (RMI) application using java.

THEORY:

Remote Method Invocation (RMI)

Java's Remote Method Invocation (commonly referred to as RMI) is used for client and server models. RMI is the object oriented equivalent to RPC (Remote procedure call). The Java Remote Method Invocation (RMI) system allows an object running in one Java Virtual Machine (VM) to invoke methods on an object running in another Java VM. RMI provides for remote communication between programs written in the Java programming language.

RMI is defined to use only with the Java platform. If you need to call methods between different language environments, use CORBA. With CORBA a Java client can call C++ server and/or a C++ client can call a Java server. With RMI that cannot be done.

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.

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.

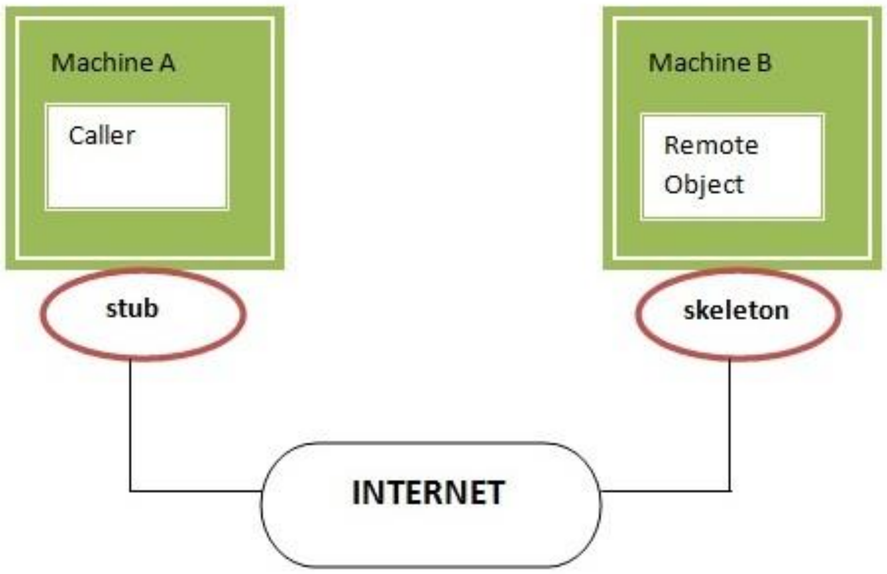


Figure: RMI Implementation

Steps to write the RMI program

1. **create the remote interface**

```
import java.rmi.*;
public interface Adder extends Remote
{
        public int add(int x,int y)throws RemoteException;
}
```

2. **Provide the implementation of the remote interface**

```
import java.rmi.*;
import java.rmi.server.*;
public class AdderRemote extends UnicastRemoteObject implements Adder
{
        AdderRemote()throws RemoteException{
        super();
}
public int add(int x,int y)
{       return x+y;}
}
```

3. **Create the stub and skeleton objects using the rmic tool.**

```
rmic AdderRemote
```

4. **Start the registry service by the rmiregistry tool**

```
rmiregistry 5000
```

5. **Create and run the server application**

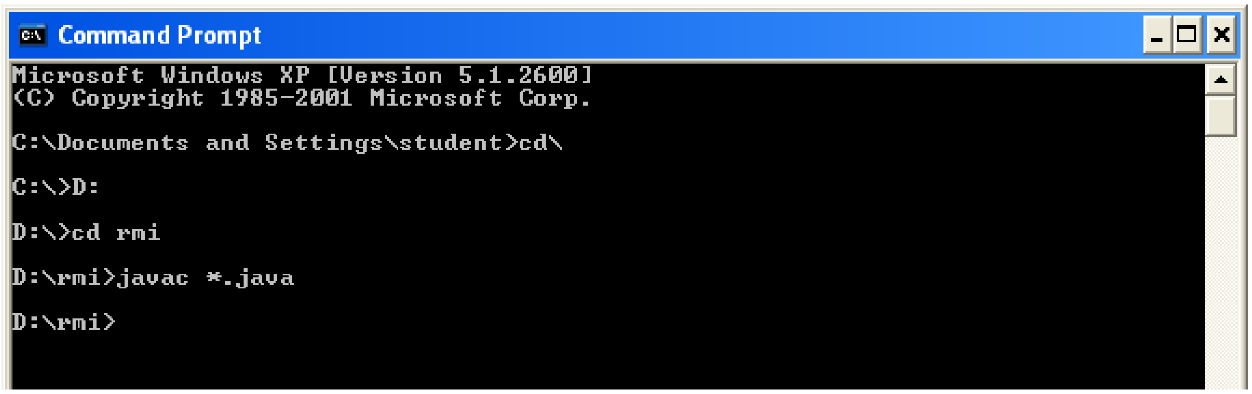
```
import java.rmi.*;
import java.rmi.registry.*;

public class MyServer
{
                        public static void main(String args[])
                        {
                            try
                            {
                                    Adder stub=new AdderRemote();
                                    Naming.rebind("rmi://localhost:5000/sonoo",stub);
                            }
                            catch(Exception e)
                            {       System.out.println(e);          }
                        }
}
```

6. **Create and run the client application**

```
import java.rmi.*;
public class MyClient
{
        public static void main(String args[])
        {
                try
                {
                        Adder stub=(Adder)Naming.lookup("rmi://localhost:5000/sonoo");
                        System.out.println(stub.add(34,4));
                }
                catch(Exception e)
                {       System.out.println(e);          }
        }
}
```

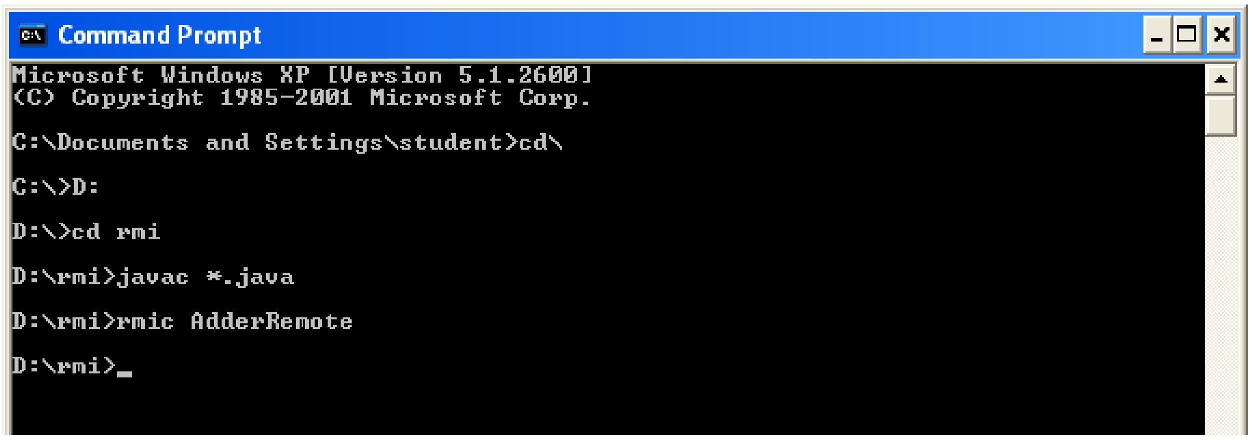
Compiling and Running the System

1. Compile client and server


Command Prompt

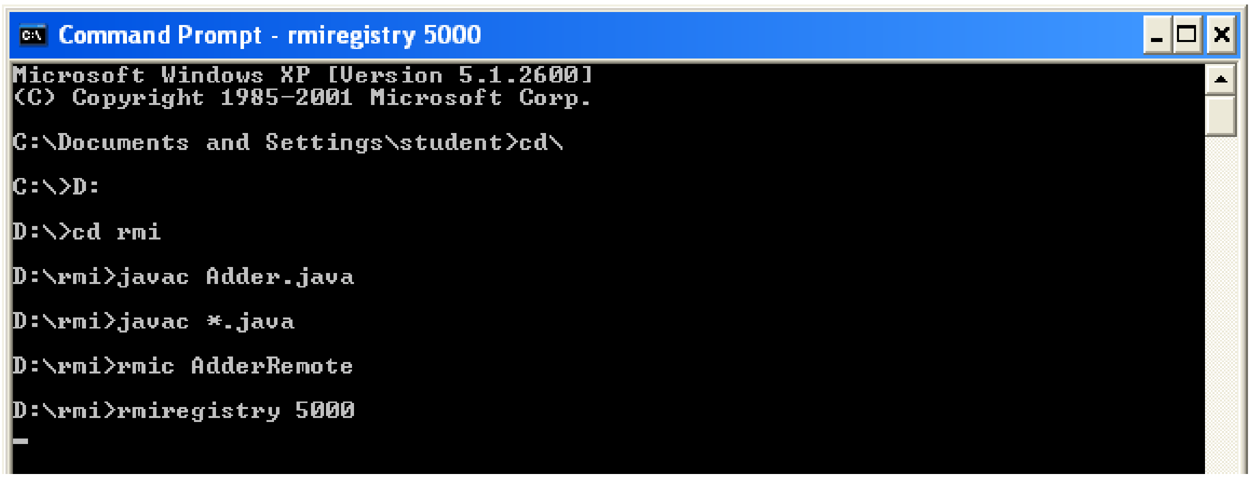
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\student>cd\

C:\>D:

D:\>cd rmi

D:\rmi>javac *.java

D:\rmi>
```

2. Generate the client stub and the server skeleton.


Command Prompt

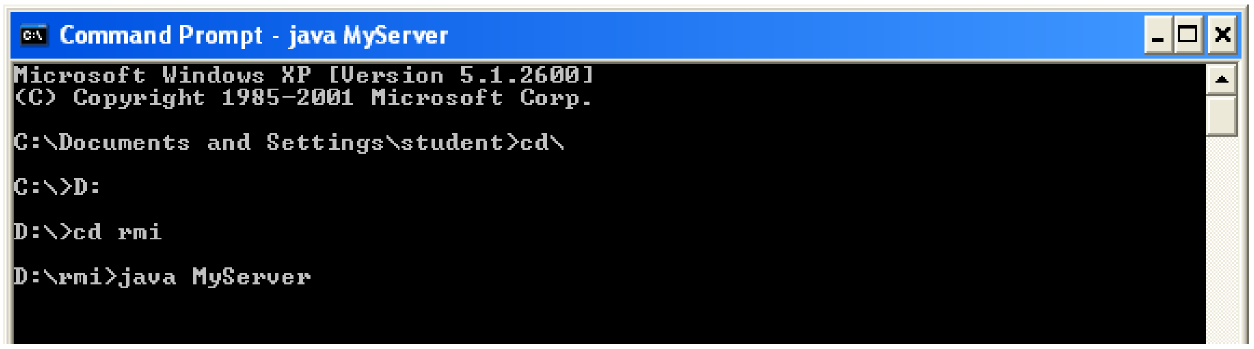
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\student>cd\

C:\>D:

D:\>cd rmi

D:\rmi>javac *.java

D:\rmi>rmic AdderRemote

D:\rmi>_
```

3. Start the RMI registry.


Command Prompt - rmiregistry 5000

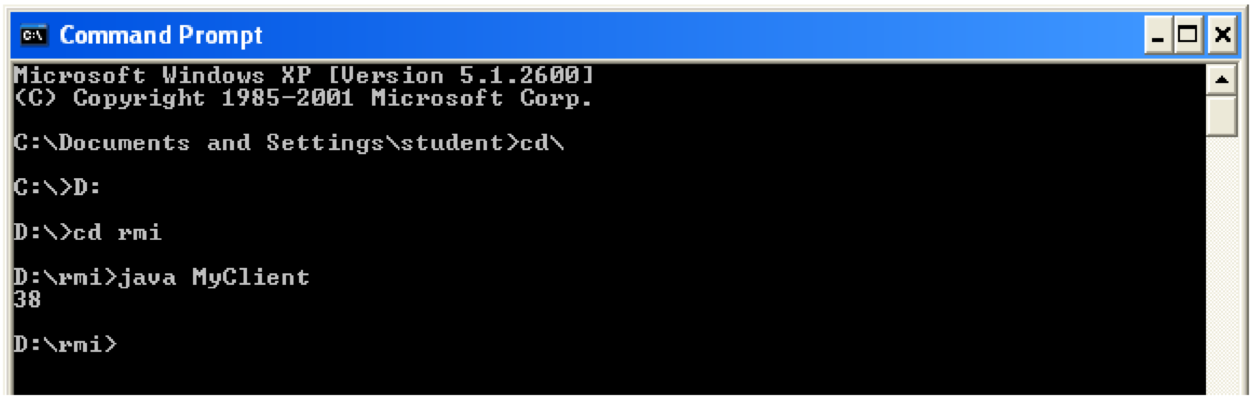
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\student>cd\

C:\>D:

D:\>cd rmi

D:\rmi>javac Adder.java

D:\rmi>javac *.java

D:\rmi>rmic AdderRemote

D:\rmi>rmiregistry 5000
_
```

4. Start the server


Command Prompt - java MyServer

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\student>cd\

C:\>D:

D:\>cd rmi

D:\rmi>java MyServer
```

5. Start the client.


Command Prompt

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\student>cd\

C:\>D:

D:\>cd rmi

D:\rmi>java MyClient
38

D:\rmi>
```

CONCLUSION:

Thus we have studied and demonstrated the steps of Remote Method Invocation (RMI) in java.