# Experiment 04: JAVA REMOTE METHOD INVOCATION

**Learning Objective:** Student should be able to perform remote method invocation in Java

**Tools:** Java Development Kit, Text Editor

## Theory

RMI stands for Remote Method Invocation. It is a mechanism that allows an object residing in a system (JVM) to access/invoke an object running on another JVM.

RMI is used to build distributed applications; it provides remote communication between Java Programs. It is provided in the package java.rmi.

### Working of an RMI Application

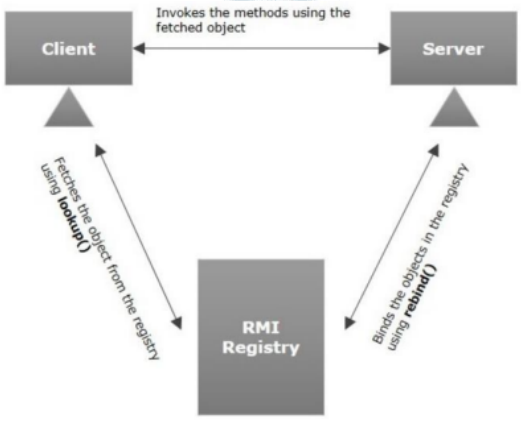
The following points summarize how an RMI application works −

● When the client makes a call to the remote object, it is received by the stub which eventuallypassesthisrequest to the RRL.

● When the client-side RRL receives the request, it invokes a method called invoke() of the object remoteRef. It passes the request to the RRL on the server side.

● The RRL on the server side passes the request to the Skeleton (proxy on the server) which finally invokes the required object on the server.

● The result is passed all the way back to the client.



**IMPLEMENTATION**

**RMI_inteface.java**

```
package pkg_RMI;
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface RMI_interface {
	public void displayMessage()  throws RemoteException;
}
```

**RMI_Server.java**

```
package pkg_RMI;

import java.rmi.server.UnicastRemoteObject;
import java.rmi.AlreadyBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;

public class RMI_Server extends UnicastRemoteObject implements RMI_interface {
	public static int i = 0;
	public RMI_Server() throws RemoteException {
	super();
	}

	public static void main(String[] args) throws RemoteException, AlreadyBoundException
{
	try {
	Registry registry = LocateRegistry.createRegistry(1878);
	registry.bind("hello", new RMI_Server());
	System.out.println("The RMI_Server is running and ready...");
	} catch (Exception e) {
	System.out.println("The RMI_Server is not running...");
	}
	}

	@Override
	public void displayMessage() throws RemoteException {
	System.out.println("-----------------------------------------------");
	i++;
	System.out.println("No. of calls : "+i);
	System.out.println("-----------------------------------------------");
	}
}
```

**RMI_Client.java**

```
package pkg_RMI;

import java.net.MalformedURLException;
import java.rmi.RemoteException;
import java.rmi.NotBoundException;
import java.rmi.Naming;

public class RMI_Client {
        public static void main(String[] args) throws MalformedURLException,
RemoteException, NotBoundException {

        try {
        RMI_interface helloAPI = (RMI_interface)
Naming.lookup("rmi://localhost:1878/hello");
        helloAPI.displayMessage();
        } catch (Exception e) {
        System.out.println("The RMI APP is not running...");
        e.printStackTrace();
        }
        }

}
```

**OUTPUT**

```
[admin@archlinux SE4]$ javac pkg_RMI/RMI_Server.java
^[[A[admin@archlinux SE4]$ java pkg_RMI/RMI_Server
The RMI_Server is running and ready...
-----------------------------------------------
No. of calls : 0
-----------------------------------------------
-----------------------------------------------
No. of calls : 1
-----------------------------------------------
-----------------------------------------------
No. of calls : 2
-----------------------------------------------
```

```
[admin@archlinux SE4]$ javac pkg_RMI/RMI_Client.java
[admin@archlinux SE4]$ java pkg_RMI/RMI_Client
[admin@archlinux SE4]$ java pkg_RMI/RMI_Client
[admin@archlinux SE4]$ java pkg_RMI/RMI_Client
[admin@archlinux SE4]$ java pkg_RMI/RMI_Client
[admin@archlinux SE4]$
```

**Conclusion:** Java RMI Demo program was implemented to count calls in the server application from a client application

For Faculty Use

| Correction Parameters | Formative Assessment [40%] | Timely completion of Practical [ 40%] | Attendance / Learning Attitude [20%] | |
|---|---|---|---|---|
| Marks Obtained | | | | |