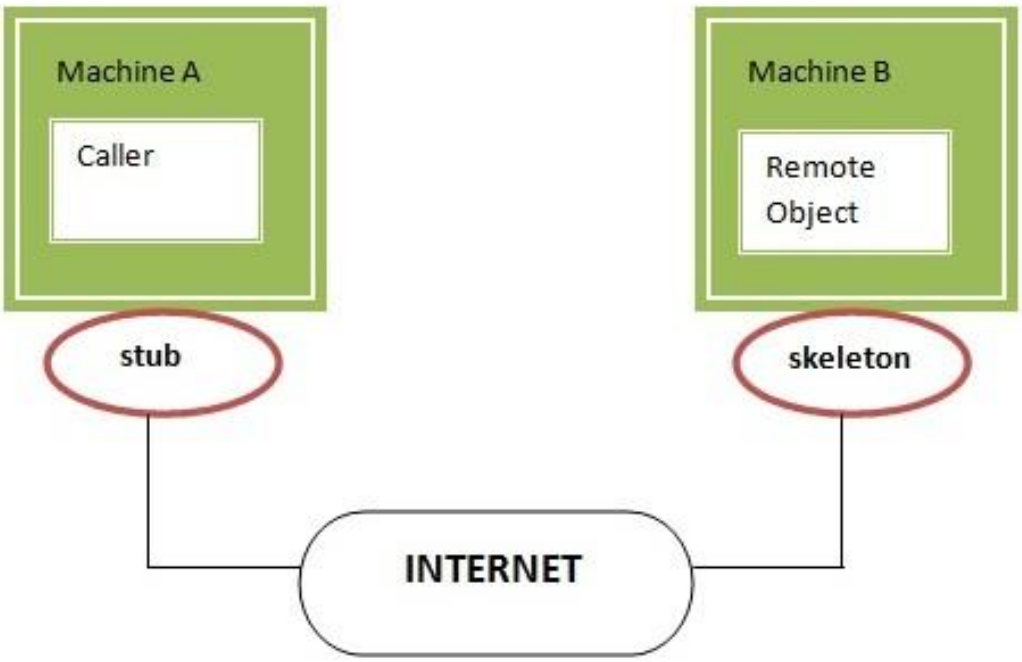
Chapter 12
RMI

Contents:

- Defining and implementing RMI Service Interface
- Creating an RMI Server and Client
- Running the RMI System

Introduction of RMI:

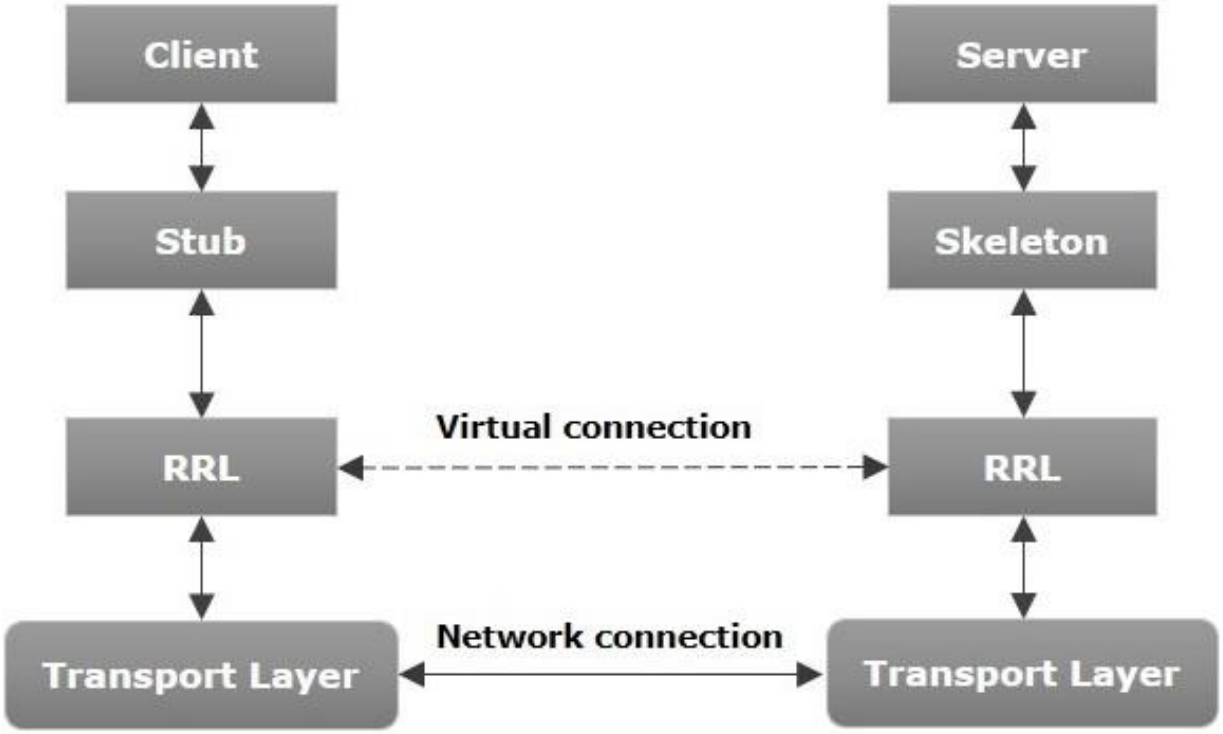
- The **RMI**(Remote Method Invocation) is an API that provides a mechanism to create distributed application in java.
- The RMI allows an object to invoke methods on an object running in another JVM.
- The RMI provides remote communication between the applications using two objects ***stub*** and ***skeleton***.
- It is provided in the package **java.rmi**



Machine A can invoke Method of Machine B

Architecture of RMI:

- Inside the server program, a remote object is created and reference of that object is made available for the client (using the registry).
- The client program requests the remote objects on the server and tries to invoke its methods.



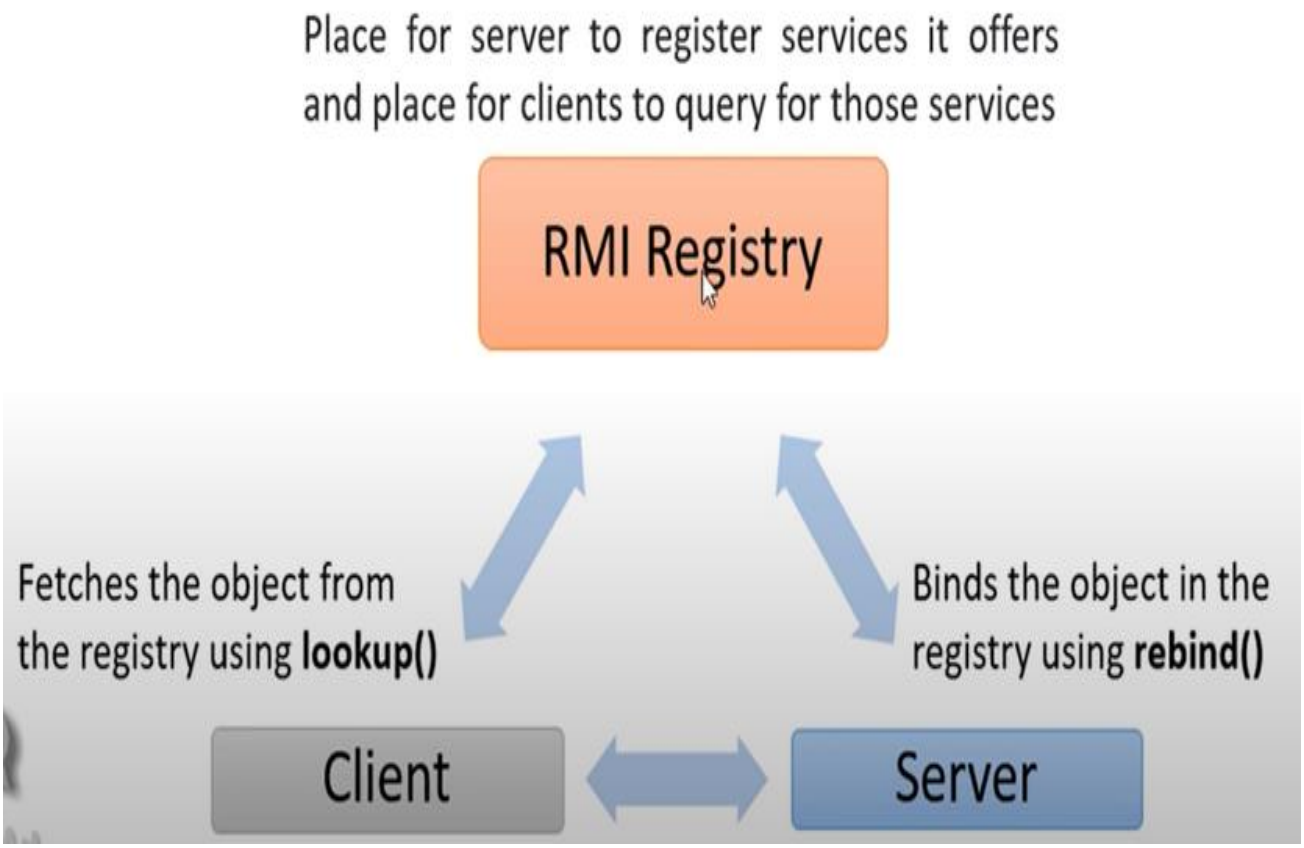
- **Transport Layer** − This layer connects the client and the server. It manages the existing connection and also sets up new connections.
- **Stub** − A stub is a representation (proxy) of the remote object at client. It resides in the client system; it acts as a gateway for the client program.
- **Skeleton** − This is the object which resides on the server side. **stub** communicates with this skeleton to pass request to the remote object.
- **RRL(Remote Reference Layer)** − It is the layer which manages the references made by the client to the remote object.

RMI Registry:

- RMI registry is a namespace on which all server objects are placed.
- Each time the server creates an object, it registers this object with the RMIregistry (using bind() or rebind() methods).
- These are registered using a unique name known as **bind name**.
- To invoke a remote object, the client needs a reference of that object. At that time, the client fetches the object

from the registry using its bind name (using **lookup()** method).

The following illustration explains the entire process –



Executing RMI Application:

- Compile the Remote Interface
- Compile the Implementation class
- Compile the Server program
- Compile the Client program

Step to write the RMI Program:

1. Create the remote interface

```
import java.rmi.*;

public interface HelloInterface extends Remote {
        public String say() throws RemoteException;
}
```

2. Create the implementation Class (remote object)

```
//This is IMPLEMENTER class

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

public class Hello extends UnicastRemoteObject
implements HelloInterface {
```

```
        private String message;

    public Hello(String msg) throws RemoteException {
            message = msg;
        }

        public String say() throws RemoteException {
            return message;
        }
    }
```

3. Create, define, and run the Server application

```
    import java.io.*;
    import java.rmi.*;
    import java.rmi.registry.LocateRegistry;
    import java.rmi.registry.Registry;

    public class HelloServer {
        public static void main(String[] argv) {
            try {
                // Create an instance of the remote object
                Hello robj = new Hello("Hello, world!");

            // Start the RMI registry on port 1099
            Registry registry =
    LocateRegistry.createRegistry(1099);

            // Bind the remote object to the registry
            registry.rebind("Hello Service", robj);
```

```
                System.out.println("Hello Server is ready.");
            } catch (Exception e) {
                System.out.println("Hello Server failed: " +
e);
            }
        }
```

4. Create, define, and run the Client application

```
import java.rmi.*;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.io.*;

public class HelloClient {
        public static void main(String[] argv) {
            try {
                // Get a reference to the RMI registry
          Registry registry =
LocateRegistry.getRegistry("localhost", 1099);

          // Look up the remote object from the registry
                HelloInterface hello = (HelloInterface)
registry.lookup("Hello Service");
                System.out.println(hello.say());
            } catch (Exception e) {
                System.out.println("HelloClient exception: "
+ e);
            }
        }}
```

Executing:

Step 1: Compile all the java files

Command Prompt

```
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Aayush Chaudhary>cd desktop\RMI

C:\Users\Aayush Chaudhary\Desktop\RMI>javac *.java

C:\Users\Aayush Chaudhary\Desktop\RMI>
```

Step 2: Start the rmi Registry using the command rmiregistry

```
C:\Users\Aayush Chaudhary\Desktop\RMI>rmiregistry
WARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by sun.rmi.registry.RegistryImpl
WARNING: Please consider reporting this to the maintainers of sun.rmi.registry.RegistryImpl
WARNING: System::setSecurityManager will be removed in a future release
```

Step 3: Run the Server Class

Command Prompt - java HelloServer

```
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Aayush Chaudhary>cd desktop\RMI

C:\Users\Aayush Chaudhary\Desktop\RMI>java HelloServer
Hello Server is ready.
```

Step 4: Run the Client Class

Command Prompt

```
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Aayush Chaudhary>cd desktop\RMI

C:\Users\Aayush Chaudhary\Desktop\RMI>java HelloClient
Hello, world!
```