RMI (Remote method Invocation)

⇒ The RMI 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 on another jvm.

⇒ RMI provides remote communication between the application using two object stub and skeleton.

Stub and skeleton

- RMI uses stub and skeleton object for communication with remote object.
- A remote object is an object whose method can be invoked from another JVM.

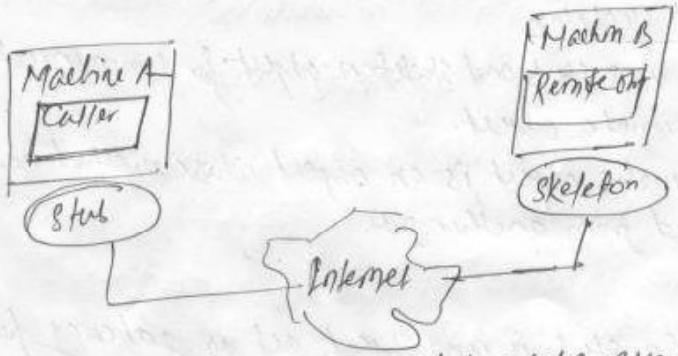
Stub

- The stub is an object, act as gateway for the client side. All the outgoing request are routed through it. It resides at the client side and represent the remote object. When the caller invokes method on stub object, it does the following tasks.
  1. It initiates a connection with Remote virtual machine
  2. It writes and transmits (marshal) the parameter to remote virtual machine.
  3. It waits for the result
  4. It reads the return value or exception

Skeleton

↳ The skeleton is an object, act as gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incomming request, it does following task

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



following 6 steps are required to create RMI program

① Create the remote interface
② Provide the implementation of remote interface.
③ Compile the implementation class and create the stub and skeleton object using the rmitool.
④ Start the registry service by rmiregistry tool.
⑤ Create and start the remote application
⑥ Create and start the client application.

The Remote Interface

↳ Remote interface should be implemented by both client and server side.

① Interface

```
package com.mykong.rmiinterface;
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface RMIInterface extends Remote {
    public String helloTo(String name) throws
                                    RemoteException;
}
```

→ The interface must always be public and extend Remote.

→ ~~RemoteException in there~~

→ All methods declared in interface must list RemoteException in their throws clause.

② Server

→ Server must extends UnicastRemoteObject and implements the interface

→ The main method will bind the server on localhost with name MyServer.

```
package com.mykon.rmiserver;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import com.mylong.rmiinkeface.RMIInterface;
```

```
public class ServerOperation extends UnicastRemoteObject
                implement RMIInterface {
    private static final long serialVersionUID = 1L;
    protected ServerOperation() throws RemoteException {
        super();
    }
    @Override
    public String helloTo(String name) throws RemoteException {
        System.err.println(" " name + " is trying to contact");
        return "Server sys hello to " + name;
    }
    public static void main(String[] args) {
        try {
            Naming.rebind("//localhost/Myserver",
                    new ServerOperation());
            System.err.println("Server ready");
        }
        catch(Exception e) {
            System.err.println("server exception: " + e.toString);
            e.printStackTrace();
        }
    }
}
```

The client

```
package com.mycong.rmiclient;
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import javax.swing.JOptionPane;
import com.mycong.rmiinterface.RMIInterface;
public class ClientOperation {
    private static RMIInterface look_up;
    public static void main(String[] args)
            throws MalformedURLException, RemoteException,
            NotBoundException {
        look_up = (RMIInterface) Naming.lookup("//localhost/Myserver");
        String txt = JOptionPane.showInputDialog("what is ur name");
        String response = look_up.helloTo(txt);
        JOptionPane.showMessageDialog(null, response);
    }
}
```

How to run

① Goto src folder

Javac src/com/mycong/rmiinterface/RMIInterface.java
" /rmiserver/ ServerOperation.java
" /rmiclient/ ClientOperation.java

② cd src

③ start ~~rmi registry~~ rmiregistry