**Practical No. 4**

**Aim:** Write a program to generate stubs to perform any specific operation

**Theory:**

A method stub or simply stub in [software development](https://en.wikipedia.org/wiki/Software_development) is a piece of code used to stand in for some other programming functionality. A stub may [simulate](https://en.wikipedia.org/wiki/Simulation) the behaviour of existing code (such as a [procedure](https://en.wikipedia.org/wiki/Subroutine) on a remote machine) or be a temporary substitute for yet-to-be-developed code. Stubs are therefore most useful in [porting](https://en.wikipedia.org/wiki/Porting), [distributed computing](https://en.wikipedia.org/wiki/Distributed_computing) as well as general software development and [testing](https://en.wikipedia.org/wiki/Software_testing).

**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

**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.

**Implementation:**

1. Compile all the java files

   javac \*.java

1. Create stub and skeleton object by rmic tool

   rmic AdderRemote

1. Start rmi registry in one command prompt

   rmiregistry 5000

1. Start the server in another command prompt

  java MyServer

1. Start the client application in another command prompt

   java MyClient

**Program:**

1. **Adder.java**

import java.rmi.\*;

public interface Adder extends Remote

{

public int add(int x,int y)throws RemoteException;

}

1. **AdderRemote.java**

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;}

}

1. **MyClient.java**

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);

}

}

}

1. **MyServer.java**

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)

{

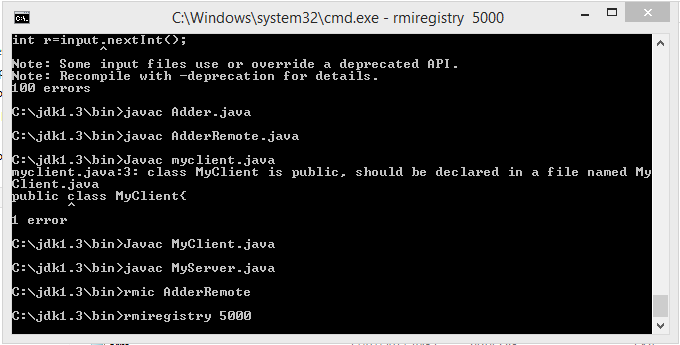
System.out.println(e);

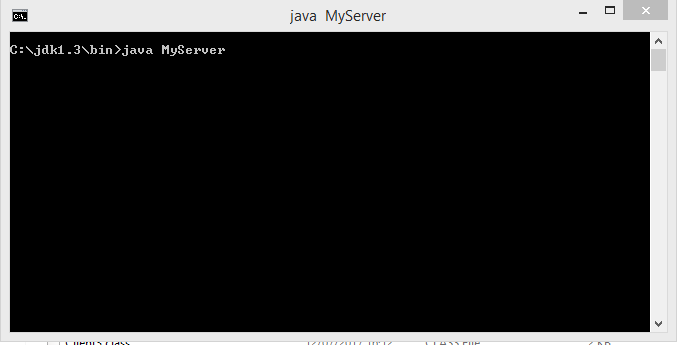
}

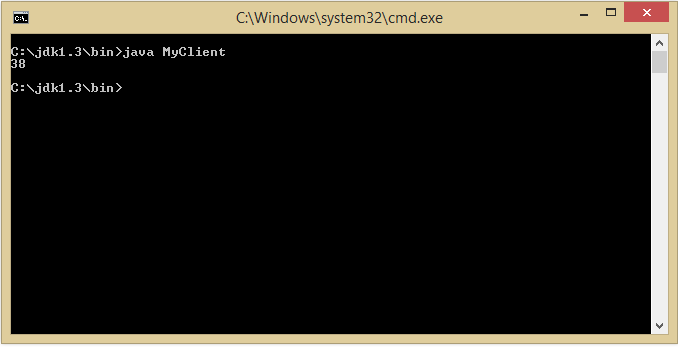
}

}

**Output:**

****

****

****

**Conclusion:** A simple program to generate a stub for performing addition of two numbers has been designed and implemented on a client server architecture.