Client.java

```
import HelloModule.*;

import org.omg.CosNaming.*;

import org.omg.CosNaming.NamingContextPackage.*;

import org.omg.CORBA.*;

import org.omg.CORBA.ORB.*;

import java.util.Scanner;

public class Client {

    public static void main(String[] args) {

        Hello HelloImpl = null;

        try {

            // create and initialize ORB

            org.omg.CORBA.ORB orb =
org.omg.CORBA.ORB.init(args,null);



            //obtaining the ORB object references for initial
services

            org.omg.CORBA.Object objRef =
orb.resolve_initial_references("NameService");



            //Naming ContextExt contains set of name
bindings of Interoperable Naming services

            NamingContextExt ncRef =
NamingContextExtHelper.narrow(objRef);



            //We have binded the name Hello from server so
using same name for lookup

            String name = "Hello";



            //Getting reference of server name hello and then
we are narrowing it down to Hello type

            HelloImpl =
HelloHelper.narrow(ncRef.resolve_str(name));



            //Taking user Input

            System.out.println("Enter your name: ");

            Scanner sc = new Scanner(System.in);

            String userName = sc.nextLine();



            //Invoking the print_hello

System.out.println(HelloImpl.print_hello(userName));

        } catch (Exception e) {

            System.out.println(e);

        }

    }

}
```

HelloImpl.java

```
module HelloModule{

    interface Hello{

        string print_hello(in string s);

    };

};
```

Server.java

```
import HelloModule.Hello;

import org.omg.CosNaming.*;

import org.omg.CosNaming.NamingContextPackage.*;

import org.omg.CORBA.*;

import org.omg.PortableServer.*;



public class Server {

    public static void main(String[] args) {

        try {

            // create and initialize ORB

            org.omg.CORBA.ORB orb =
org.omg.CORBA.ORB.init(args,null);

            //Getting reference of ROOTPOA

            POA rootPOA =
POAHelper.narrow(orb.resolve_initial_references("RootPO
A"));

            //Activating ROOTPOA

            rootPOA.the_POAManager().activate();

            //Create Object of Interface implementation which
will act as servant

            HelloImpl helloImpl = new HelloImpl();



            //Registering the servant object reference in the
rootPOA

            org.omg.CORBA.Object ref =
rootPOA.servant_to_reference(helloImpl);
```

```
        //narrowing the ROOTPOA reference object to
propertype which in this case is of type Hello

        System.out.println("Step 1");

        Hello h_ref = HelloModule.HelloHelper.narrow(ref);



        //obtaining the ORB object references for initial
services

        System.out.println("Step 2");

        org.omg.CORBA.Object objRef =
orb.resolve_initial_references("NameService");



        //Afain narrowing the ORB object reference to
NamingContext type to bin it with server

        System.out.println("Step 3");

        NamingContextExt ncRef =
NamingContextExtHelper.narrow(objRef);



        //passing path and the servant object to the
naming service, binding the servant object to the "Hello"

        System.out.println("Step 4");

        String name = "Hello";

        NameComponent path[] = ncRef.to_name(name);

        ncRef.rebind(path,h_ref);



        //Enbaling ORB to run on main thread and waiting
till invocation comes for ORB. Since it is in main method
after invocation it will wait again

        System.out.println("Server Ready....");

        orb.run();

      } catch (Exception e) {

        System.out.println(e);

      }

   }

}
```

HelloModule.idl

```
module HelloModule{

   interface Hello{

      string print_hello(in string s);

   };};
```

RUN.SH

```
#!/bin/bash


# Compile IDL file

idlj -fall HelloModule.idl


# Compile Java files

javac *.java HelloModule/*.java


# Check if orbd is running and kill if necessary

pkill orbd


sleep 1


# Start orbd

orbd -ORBInitialPort 1050&


sleep 2


# Start the server

gnome-terminal --tab --title="Server" --command="bash -c
'java Server -ORBInitialPort 1050 -ORBInitialHost
localhost; exec bash'"


sleep 2


# Open new terminal and run the client

gnome-terminal --tab --title="Client" --command="bash -c
'java Client -ORBInitialPort 1050 -ORBInitialHost localhost;
exec bash'"
```

```
     client                          object implementati(server)

 >     <     <     <              <  >         >          <>       >

Interface   Dynamic IDL       ORB2

repository Invocation stubs interface

              Static     Dynamic object                Implementati

              IDL Selector skeleton adaptation      <Reposition

              O        R       B     C     O      R      E
```