

Reverse Server

```
import ReverseModule.Reverse;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
class ReverseServer
{
    public static void main(String[] args)  {
        try {
            // initialize the ORB
            org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);
            // initialize the BOA/POA
            POA rootPOA = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
            rootPOA.the_POAManager().activate();
            // creating the calculator object
            ReverseImpl rvr = new ReverseImpl();
            // get the object reference from the servant class
            org.omg.CORBA.Object ref = rootPOA.servant_to_reference(rvr);
            System.out.println("Step1");
            Reverse h_ref = ReverseModule.ReverseHelper.narrow(ref);
            System.out.println("Step2");
            org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
            System.out.println("Step3");
            NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
            System.out.println("Step4");
            String name = "Reverse";
            NameComponent path[] = ncRef.to_name(name);
            ncRef.rebind(path,h_ref);
            System.out.println("Reverse Server reading and waiting....");
            orb.run();
        }
        catch(Exception e)
```

```
{  
    e.printStackTrace();  
}  
}  
}
```

Reverse Client

```
import ReverseModule.*;  
import org.omg.CosNaming.*;  
import org.omg.CosNaming.NamingContextPackage.*;  
import org.omg.CORBA.*;  
import java.io.*;  
class ReverseClient{  
    public static void main(String args[])  {  
        Reverse ReverseImpl=null;  
        try {  
            // initialize the ORB  
            org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);  
            org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");  
            NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);  
            String name = "Reverse";  
            ReverseImpl = ReverseHelper.narrow(ncRef.resolve_str(name));  
            System.out.println("Enter String=");  
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  
            String str= br.readLine();  
            String tempStr= ReverseImpl.reverse_string(str);  
            System.out.println(tempStr);  
        }  
        catch(Exception e){  
            e.printStackTrace();  
        }  
    }  
}
```

Reverse Implementation

```
import ReverseModule.ReversePOA;
import java.lang.String;
class ReverseImpl extends ReversePOA{
    ReverseImpl() {
        super();
        System.out.println("Reverse Object Created");
    }
    public String reverse_string(String name) {
        StringBuffer str=new StringBuffer(name);
        str.reverse();
        return ("Server Send "+str);
    }
}
```

Output

The image displays three sequential screenshots of a Windows command prompt window, showing the process of compiling and running a CORBA Reverse module.

First Screenshot: Shows the compilation of the ReverseModule.idl file. The command is `F:\BE\8th\DS\Practical\IDL CORBA>idlj -fall ReverseModule.idl`. This is followed by two `javac` commands to compile the Java files. The first `javac *.java` command produces a warning: `Note: .\ReverseModule\ReversePOA.java uses unchecked or unsafe operations. Note: Recompile with -Xlint:unchecked for details.` The second `javac *.java ReverseModule/*.java` command also produces the same warning.

Second Screenshot: Shows the execution of the ReverseServer. The command is `F:\BE\8th\DS\Practical\IDL CORBA> java ReverseServer -ORBInitialPort 1050& -ORBInitialHost localhost&`. The output shows the server starting and waiting for connections: `Reverse Object Created`, `Step1`, `Step2`, `Step3`, `Step4`, and `Reverse Server reading and waiting....`

Third Screenshot: Shows the execution of the ReverseClient. The command is `F:\BE\8th\DS\Practical\IDL CORBA> java ReverseClient -ORBInitialPort 1050 -ORBInitialHost localhost`. The output shows the client sending a string: `Enter String=`, `Hello I am Akshay Bhore`, and the server's response: `Server Send roHb yahskA ma I olleH`. The client then sends another string: `Enter String=`, `This is a Testing for CORBA`, and the server's response: `Server Send ABROC rof gnitseT a si siHT`. The prompt ends with `F:\BE\8th\DS\Practical\IDL CORBA>`.