Experiment No: 6

Step 1: Install Java and IDL Compiler

```
sudo apt update
sudo apt install default-jdk
```

Step 2: Create the IDL File

1. Create a directory for the project:

```
mkdir HelloApp
cd HelloApp
nano Hello.idl
```

```
module HelloApp {
   interface Hello {
       string sayHello();
   };
};
```

Step 3: Compile the IDL File

```
idlj -fall Hello.idl
```

This generates several files:

- Hello.java
- HelloHelper.java
- HelloHolder.java
- _HelloStub.java
- HelloPOA.java
- HelloOperations.java

Step 4: Create the Server

```
nano HelloServer.java
```

```
import HelloApp.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;

class HelloServant extends HelloPOA {
   public String sayHello() {
      return "\nHello world!!\n";
   }
}

public class HelloServer {
   public static void main(String[] args) {
      try {
         // Create and initialize the ORB
         ORB orb = ORB.init(args, null);

         // Get reference to root POA and activate the POAManager
         POA rootpoa = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
         rootpoa.the_POAManager().activate();

         // Create servant and register it with the ORB
         HelloServant helloImpl = new HelloServant();
         org.omg.CORBA.Object ref = rootpoa.servant_to_reference(helloImpl);
         Hello href = HelloHelper.narrow(ref);

         // Get the naming service and bind the object reference
         org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
         NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
```

```
            NameComponent path[] = ncRef.to_name("Hello");
            ncRef.rebind(path, href);

            System.out.println("HelloServer ready and waiting...");
            // Wait for invocations from clients
            orb.run();
        } catch (Exception e) {
            System.out.println("ERROR: " + e);
            e.printStackTrace(System.out);
        }
        System.out.println("HelloServer Exiting...");
    }
}
```

Step 5: Create the Client

```
nano HelloClient.java
```

```
import HelloApp.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextExt;
import org.omg.CosNaming.NamingContextExtHelper;

public class HelloClient {
    public static void main(String[] args) {
        try {
            // Create and initialize the ORB
            ORB orb = ORB.init(args, null);

            // Get the naming service reference
            org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
            NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
            // Resolve the object reference in the naming
            String name = "Hello";
            Hello helloImpl = HelloHelper.narrow(ncRef.resolve_str(name));


            System.out.println("Obtained a handle on the server object.");
            System.out.println(helloImpl.sayHello());


        } catch (Exception e) {
            System.out.println("ERROR: " + e);
            e.printStackTrace(System.out);
        }
    }
}
```

Step 6: Compile the Java Files

```
javac HelloServer.java HelloClient.java HelloApp/*.java
```

Step 7: Start the Name Server

```
tnameserv -ORBInitialPort 1050 &
```

Step 8: Run the Server

```
java HelloServer -ORBInitialPort 1050 -ORBInitialHost localhost
```

Step 9: Run the Client

```
java HelloClient -ORBInitialPort 1050 -ORBInitialHost localhost
```

Output:

On the server side:

```
HelloServer ready and waiting...
```

On the client side:

```
Obtained a handle on the server object.
Hello world!!
```