PROJECT 2: FUNCTIONAL CALCULATOR USING CORBA IDL

CORBA stands for Common Object Request Broker Architecture.

The following project has been developed using CORBA IDL.

IDL (Interface Definition Language) is the language that describes the interface for the server.

Once the client knows that interface, effective communication takes place between the client and the server.

The project has been implemented in the following steps:

STEP1:

The project starts with defining an IDL file which contains the required interfaces, methods, parameters, events, arguments, exceptions etc. within a module. The IDL file is saved as "Calc.idl"

STEP2:

Map the IDL file into its equivalent JAVA file using the following commands:

idlj –fclient Calc.idl

idlj –fserver Calc.idl

The above commands lead to the formation of the following files within the package:

a)Calc.java

This file contains the JAVA version of the IDL interface.

b)_CalcStub.java

This file is the Client Stub.

c)CalcPOA.java

This file is the Server Skeleton.

d)CalcHelper.java

This Class provides the narrow() to cast the CORBA object references to the proper data types.

e)CalcHolder.java

This class is used to handle and provide functionalities to in/out/in-out statements.

f)CalcOperations.java

This Class contains all the methods defined in the IDL interface in its JAVA version.

STEP3:

Implement the Server.

The server side has two classes: Servant class and the Server class. Servant Class contains the implementations of the IDL interface. Server Class contains the main().

Server Class performs the following functionalities:

- A CORBA server needs a local ORB object. The server instantiates the ORB and registers its servant objects so that the ORB can find the server on client invocation.

- The ORB then obtains an initial reference to the Name Service. Reference to the root POA (Portable Object Adapter) is achieved and the POA manager is activated so that it becomes portable and can process requests.
- The server instantiates the servant objects to perform IDL interface defined operations.
- Common Object Services (COS) Naming Service is used to make the servant object's operations available to the clients.
- Obtain the Initial Naming Context. Pass the string "NameService" so that the ORB returns a naming context object.
- Use the narrow() to cast the object reference to its proper type.
- Register the servant with the Name Server by passing the string "Calc".
- Use the run() to wait for the client requests.

STEP4:

Implement the Client.

Client class contains the main() and performs the following functionalities:

- A CORBA client needs a local ORB object. Hence the client needs to create and initialize the ORB.
- The client then uses the COS Naming Service to locate the Server.
- The client needs to get an Initial Naming Context by calling the resolve_initial_references().
- Narrow the object reference to its proper type by using the narrow().
- Invoke the required methods on the servant class as inputted by the client.

STEP5:

This part describes how I actually made a working calculator.

- Accept a string from the user in the form "+64".
- Separate the string into operator and two operands using charAt() and storing them in three different characters.
- Convert the operator into its ASCII value and the operators into their integer values using Character.getNumericValue().
- Pass the ASCII value of the operator and the two operands to the calculate().
- Store the returned result in a variable and display the result.
- If the client wants to continue then accept another string else invoke the exit()/shutdown().

STEP6:

Running the Calculator Application.

- Compile the server using the following command: javac CalcServer.java
- Compile the client using the following command: javac CalcClient.java
- Start the Name Service using the following command:
  start orbd –ORBInitialPort 1050 –ORBInitialHost localhost
- Start the server using the following command:
  java CalcServer –ORBInitialPort 1050 –ORBInitialHost localhost
- Start the Client using the following command:
  java CalcClient –ORBInitialPort 1050 –ORBInitialHost localhost