Write a program to implement CORBA mechanism by using C ++ program at one end and Java program on the other.

Parallel And Distributed Systems Lab (Delhi Technological University)

EXPERIMENT 8

AIM: Write a program to implement CORBA mechanism by using C++ program at one end and Java program on the other.

THEORY:

Introduction:
CORBA, the Common Object Request Broker Architecture, is a powerful tool for distributed programming. CORBA is a mechanism in software for normalizing the method-call semantics between application objects that reside either in the same address space (application) or remote address space (same host, or remote host on a network). Version 1.0 was released in October 1991. CORBA uses an interface definition language (IDL) to specify the interfaces that objects will present to the outside world. CORBA then specifies a mapping from IDL to a specific implementation language like C++or Java. Standard mappings exist for Ada, C ,C++, Lisp, Ruby, Smalltalk, Java, COBOL, PL/Iand Python. There are also nonstandard mappings for Perl, Visual Basic, Erlang, and Tcl implemented by object request brokers(ORBs) written for those languages. The CORBA specification dictates that there shall be an ORB through which the Application interacts with other objects. In practice, the application simply initializes the ORB, and accesses an internal Object Adapter which maintains such issues as reference counting, object (and reference) instantiation policies, object lifetime policies, etc. The Object Adapter is used to register instances of the generated code classes. Generated code classes are the result of compiling the user IDL code, which translates the high-level interface definition into an OS-and language-specific class base for use by the user application. This step is necessary in order to enforce the CORBA semantics and provide a clean user process for interfacing with the CORBA infrastructure.

Description:
Data communication from client to server is accomplished through a well defined object-oriented interface. The object request broker (ORB) determines the location of the target object, sends a request to that object, and returns any response back to the caller.

CODE:

```
//Developing the Server Program
#include <iostream>
#include "OB/CORBA.h"
#include <OB/Cosnaming.h>
#include "crypt.h"
#include "cryptimpl.h"

using namespace std;

int main(int argc, char** argv)
{
        // Declare ORB and servant object
        CORBA::ORB_var orb;
        CryptographicImpl* CrypImpl = NULL;

        try {
                // Initialize the ORB.
                orb = CORBA::ORB_init(argc, argv);
                // Get a reference to the root POA
                CORBA::Object_var rootPOAObj =
                        orb->resolve_initial_references("RootPOA");
                // Narrow it to the correct type
                PortableServer::POA_var rootPOA =
                        PortableServer::POA::_narrow(rootPOAObj.in());

                // Create POA policies
                CORBA::PolicyList policies;
                policies.length(1);

                policies[0] =
                        rootPOA->create_thread_policy
                        (PortableServer::SINGLE_THREAD_MODEL);

                // Get the POA manager object
                PortableServer::POAManager_var manager = rootPOA->the_POAManager();

                // Create a new POA with specified policies
                PortableServer::POA_var myPOA = rootPOA->create_POA
                                                ("myPOA", manager, policies);

                // Free policies
                CORBA::ULong len = policies.length();
                for (CORBA::ULong i = 0; i < len; i++)
                        policies[i]->destroy();

                // Get a reference to the Naming Service root_context
                CORBA::Object_var rootContextObj =
                        orb->resolve_initial_references("NameService");
                // Narrow to the correct type
                CosNaming::NamingContext_var nc =
                        CosNaming::NamingContext::_narrow(rootContextObj.in());

                // Create a reference to the servant
                CrypImpl = new CryptographicImpl(orb);
                // Activate object
                PortableServer::ObjectId_var myObjID =
                        myPOA->activate_object(CrypImpl);
                // Get a CORBA reference with the POA through the servant
                CORBA::Object_var o = myPOA->servant_to_reference(CrypImpl);
                // The reference is converted to a character string
                CORBA::String_var s = orb->object_to_string(o);
                cout << "The IOR of the object is: " << s.in() << endl;

                CosNaming::Name name;
                name.length(1);
                name[0].id = (const char *) "CryptographicService";
                name[0].kind = (const char *) "";
                // Bind the object into the name service
                nc->rebind(name,o);

                // Activate the POA
                manager->activate();
                cout << "The server is ready.
                        Awaiting for incoming requests..." << endl;
                // Start the ORB
                orb->run();

        } catch(const CORBA::Exception& e) {
                // Handles CORBA exceptions
                cerr << e << endl;
        }

        // Decrement reference count
        if (CrypImpl)
```

```
        CrypImpl->_remove_ref();

 // End CORBA
 if (!CORBA::is_nil(orb)){
     try{
        orb->destroy();
        cout << "Ending CORBA..." << endl;
     } catch (const CORBA::Exception& e)
     {
         cout << "orb->destroy() failed:" << e << endl;
         return 1;
     }
 }
 return 0;
}
```

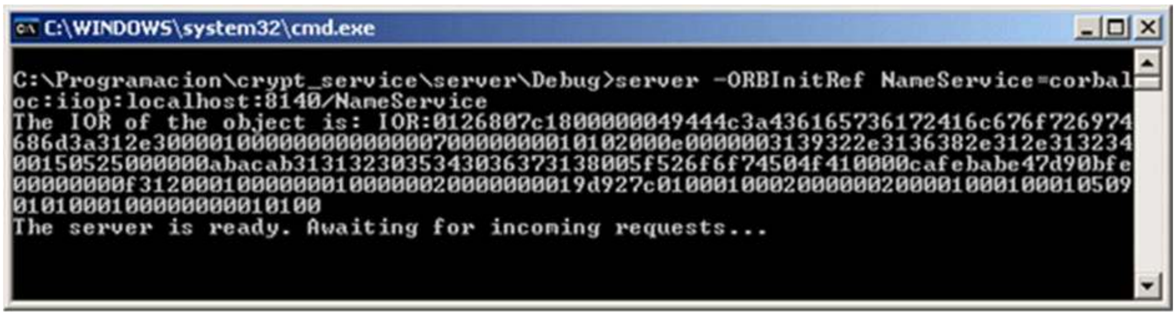
// Developing the Client Program

```
#include <iostream>
#include <string>
#include "OB/CORBA.h"
#include "OB/Cosnaming.h"
#include "crypt.h"

using namespace std;

int main(int argc, char** argv)
{

    // Declare ORB
    CORBA::ORB_var orb;

    try {

        // Initialize the ORB
        orb = CORBA::ORB_init(argc, argv);

        // Get a reference to the Naming Service
        CORBA::Object_var rootContextObj =
            orb->resolve_initial_references("NameService");
        CosNaming::NamingContext_var nc =
            CosNaming::NamingContext::_narrow(rootContextObj.in());

        CosNaming::Name name;
```

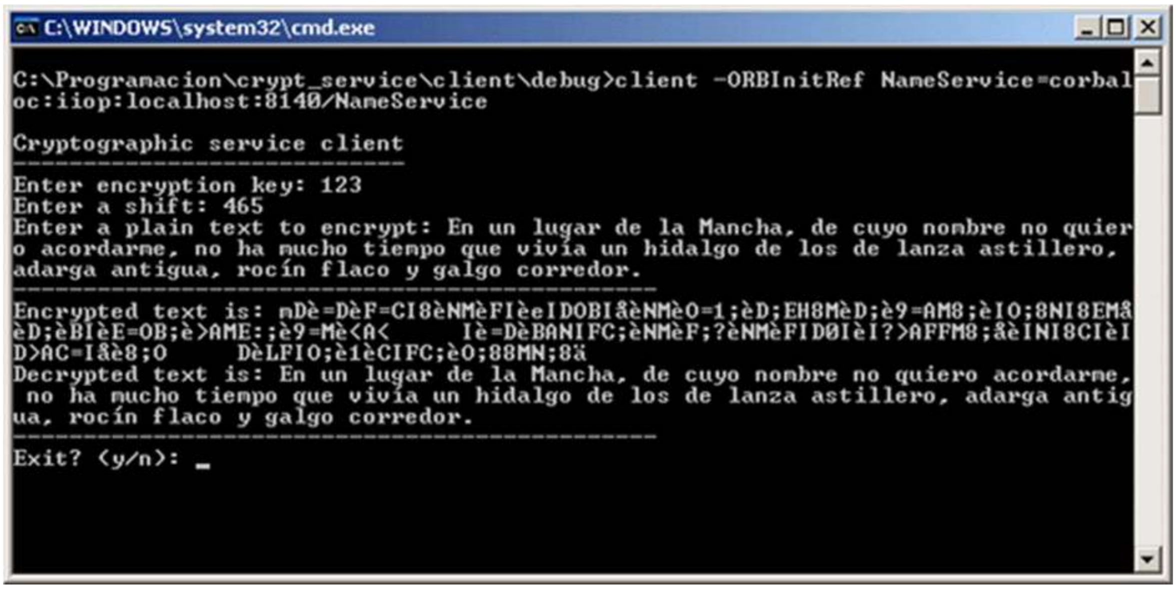
```
name.length(1);
name[0].id = (const char *) "CryptographicService";
name[0].kind = (const char *) "";
// Invoke the root context to retrieve the object reference
CORBA::Object_var managerObj = nc->resolve(name);
// Narrow the previous object to obtain the correct type
::CaesarAlgorithm_var manager =
        ::CaesarAlgorithm::_narrow(managerObj.in());

string info_in,exit,dummy;
CORBA::String_var info_out;
::CaesarAlgorithm::charsequence_var inseq;
unsigned long key,shift;

try{
        do{
                cout << "\nCryptographic service client" << endl;
                cout << "---------------------------" << endl;

                do{ // Get the cryptographic key
                        if (cin.fail())
                        {
                                cin.clear();
                                cin >> dummy;
                        }
                        cout << "Enter encryption key: ";
                        cin >> key;

                } while (cin.fail());

                do{ // Get the shift
                        if (cin.fail())
                        {
                                cin.clear();
                                cin >> dummy;
                        }
                        cout << "Enter a shift: ";
                        cin >> shift;

                } while (cin.fail());

                // Used for debug pourposes
                //key = 9876453;
                //shift = 938372;
                getline(cin,dummy); // Get the text to encrypt
                cout << "Enter a plain text to encrypt: ";
                getline(cin,info_in);

                // Invoke first remote method
                inseq = manager->encrypt
                    (info_in.c_str(),key,shift);
                cout << "----------------------------------------"
                     << endl;
                cout << "Encrypted text is: "
                    << inseq->get_buffer() << endl;
                // Invoke second remote method
                info_out = manager->decrypt(inseq.in(),key,shift);
                cout << "Decrypted text is: "
                    << info_out.in() << endl;
                cout << "-----------------------------------------"
                     << endl;
                cout << "Exit? (y/n): ";
                cin >> exit;
            } while (exit!="y");

            // Shutdown server message
            manager->shutdown();

        } catch(const std::exception& std_e){
            cerr << std_e.what() << endl;
         }
    }catch(const CORBA::Exception& e) {
        // Handles CORBA exceptions
        cerr << e << endl;
}
// End CORBA
    if (!CORBA::is_nil(orb)){
        try{
            orb->destroy();
            cout << "Ending CORBA..." << endl;
        } catch(const CORBA::Exception& e)
        {
            cout << "orb->destroy failed:" << e << endl;
            return 1;
        }
}
```

```
  return 0;
}
```

OUTPUT:



```
C:\Programacion\crypt_service\server\Debug>server -ORBInitRef NameService=corbal
oc:iiop:localhost:8140/NameService
The IOR of the object is: IOR:0126807c1800000049444c3a436165736172416c676f726974
686d3a312e300001000000000000007000000000101020000e0000003139322e3136382e312e313234
00150525000000abacab3131323035343036373138005f526f6f74504f410000cafebabe47d90bfe
00000000f3120001000000010000002000000019d927c01000100020000000200001000100010509
01010001000000000010100
The server is ready. Awaiting for incoming requests...
```



```
C:\Programacion\crypt_service\client\debug>client -ORBInitRef NameService=corbal
oc:iiop:localhost:8140/NameService

Cryptographic service client
------------------------------------
Enter encryption key: 123
Enter a shift: 465
Enter a plain text to encrypt: En un lugar de la Mancha, de cuyo nombre no quier
o acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza astillero,
adarga antigua, rocín flaco y galgo corredor.
------------------------------------------------------------
Encrypted text is: mDè=DèF=CI8èNMèFIèeIDOBIåèNMèO=1;èD;EH8MèD;è9=AM8;èIO;8NI8EMå
èD;èBIèE=OB;è>AME:;è9=Mè<A<      Iè=DèBANIFC;èNMèF;?èNMèFID0IèI?>AFFM8;åèINI8CIèI
D>AC=Iåè8;O      DèLFIO;è1èCIFC;èO;88MN;8ä
Decrypted text is: En un lugar de la Mancha, de cuyo nombre no quiero acordarme,
 no ha mucho tiempo que vivía un hidalgo de los de lanza astillero, adarga antig
ua, rocín flaco y galgo corredor.
------------------------------------------------------------
Exit? (y/n): _
```