TP CORBA

Mouhamed sane
Agrotic l2
INE N00473E20181
15 Mars 2021

1) Introduction:

Dans le cadre d'une ascension fulgurante des TIC, il devient capital d'avoir des solutions de middleware qui permettent des échanges d'information, indépendamment des plateformes matérielles, des langages de programmation et des systèmes d'exploitation. D'où l'importance de **corba**.

Dans nôtres travail nous allons présenter les différentes étapes pour mettre en place un programme simple de client-serveur sous corbra.

En premières lieu nous allons écrire et compilé le fichier **idl** ensuite codé la partie **serveur** et la partie **client** et enfin données les résultats.

2) Les différentes étapes de la création de système client-serveur sous corba

Côté serveur

Étape 1
Créer un nouveau projet Java avec Eclipse et appelez-le: CorbaCalculServer-> Cliquez sur Terminer une fois terminé.

Étape 2
Sous le projet CalculServer, créer un nouveau fichier appelé Calcul.idl. Copiez le code ci-dessous dans le nouveau fichier.

```
module CalculApp
{
  interface Calcul

   {
    long add(in long a,in long b);
    long sous(in long a,in long b);
    long div(in long a,in long b);
    long mult(in long a,in long b);
    oneway void shutdown();
   };
```

```
};
```

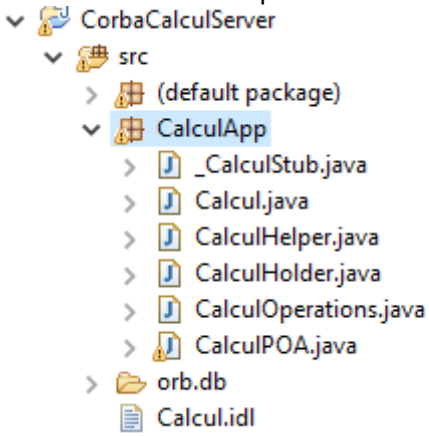
Étape 3
Ouvrez votre console CMD, changer le répertoire (cd) dans le dossier src à l'emplacement du projet.
Étape 4
Compilez le fichier idl en utilisant la commande idlj comme ci-dessous:

```
idlj -fall Calcul.idl
```

Vous verrez que dans le dossier src, un nouveau répertoire appelé CalculApp est créé avec un certain nombre de fichiers Java.

Étape 5
Rafraîchir la liste des éléments du projet créé en faisant click droit sur le répertoire source de votre projet sur Eclipse, et cliquer sur “Refresh”.
Entrer dans le répertoire CalculApp et afficher son contenu.

Voici le contenu du répertoire.



Étape 6 :
Sous le projet CorbaCalculServer, créer une nouvelle classe appelée CalculObj.
Copiez le code ci-dessous dans la nouvelle classe.

```
import CalculApp.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;
import java.util.Properties;
class CalculObj extends CalculPOA {
```

```
private ORB orb;
public void setORB(ORB orb_val) {
 orb = orb_val;
}           // implement add() method
public int add(int a, int b) {
 int r=a+b;
 return r;
}         // implement sous() method
public int sous(int a, int b){
         int r=a-b;
         return r;
}       //implement mult() method
public int mult(int a, int b){
         int r=a*b;
         return r;
}     // implement div() method
public int div(int a, int b){
         if(b!=0){
          int r =a/b;
          return r ;
         } return 0;
}
    // implement shutdown() method
public void shutdown() {
 orb.shutdown(false);
}
}
```

Étape 7

Sous le projet CalculServer, créer une nouvelle classe appelée StartServer. Copier le code ci-dessous dans la nouvelle classe.

```
import CalculApp.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;
import java.util.Properties;
public class StartServer {
public static void main(String args[]) {
```

```
try{            // create and initialize the ORB
//// get reference to rootpoa &amp;amp; activate the
POAManager

ORB orb = ORB.init(args, null);

POA rootpoa =
POAHelper.narrow(orb.resolve_initial_references("Root
POA"));

rootpoa.the_POAManager().activate();

 // create servant and register it with the ORB
CalculObj addobj = new CalculObj();
addobj.setORB(orb);

 // get object reference from the servant
org.omg.CORBA.Object ref =
rootpoa.servant_to_reference(addobj);

Calcul href = CalculHelper.narrow(ref);

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

NamingContextExt ncRef =
NamingContextExtHelper.narrow(objRef);

 NameComponent path[] = ncRef.to_name( "ABC" );

 ncRef.rebind(path, href);

 System.out.println("Calcul Server ready and waiting
...");

 // wait for invocations from clients

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

```
 }
 System.out.println("HelloServer Exiting ...");}
}
```

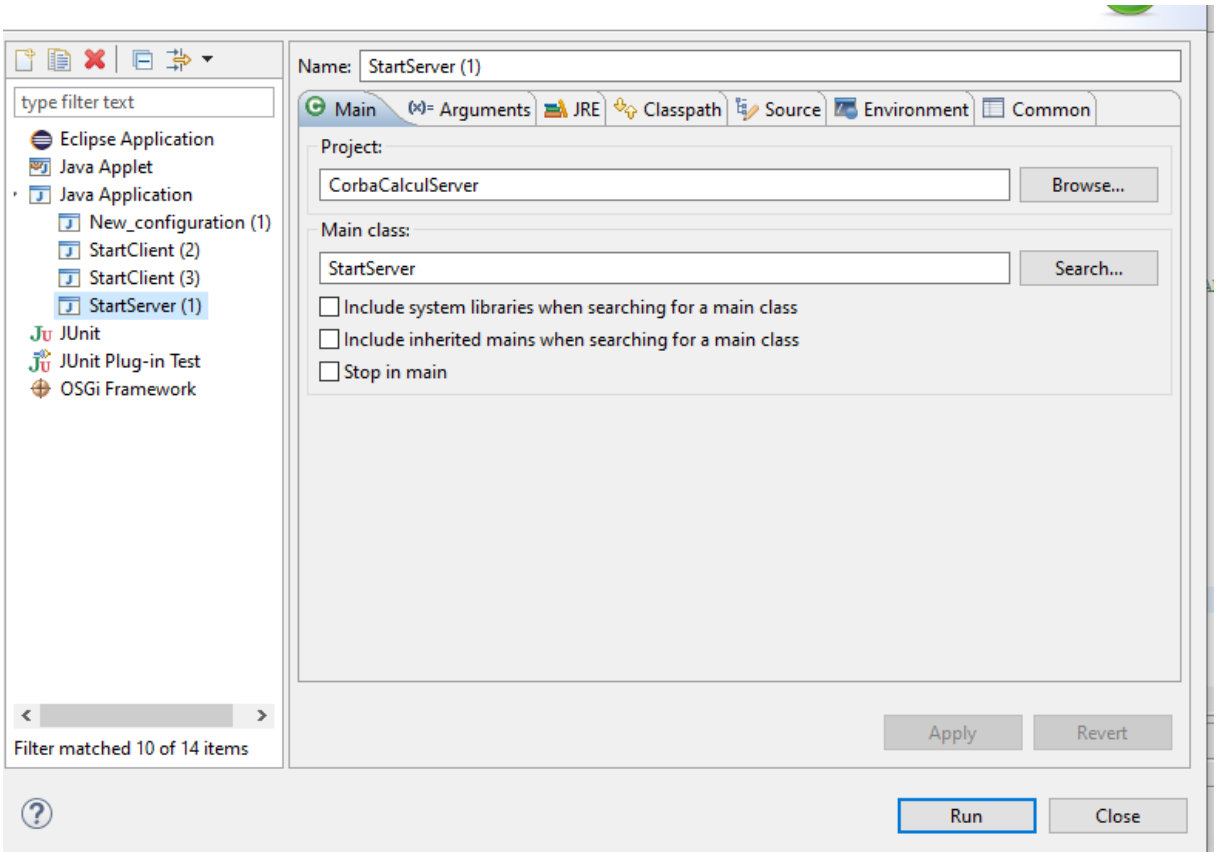
Étape 8

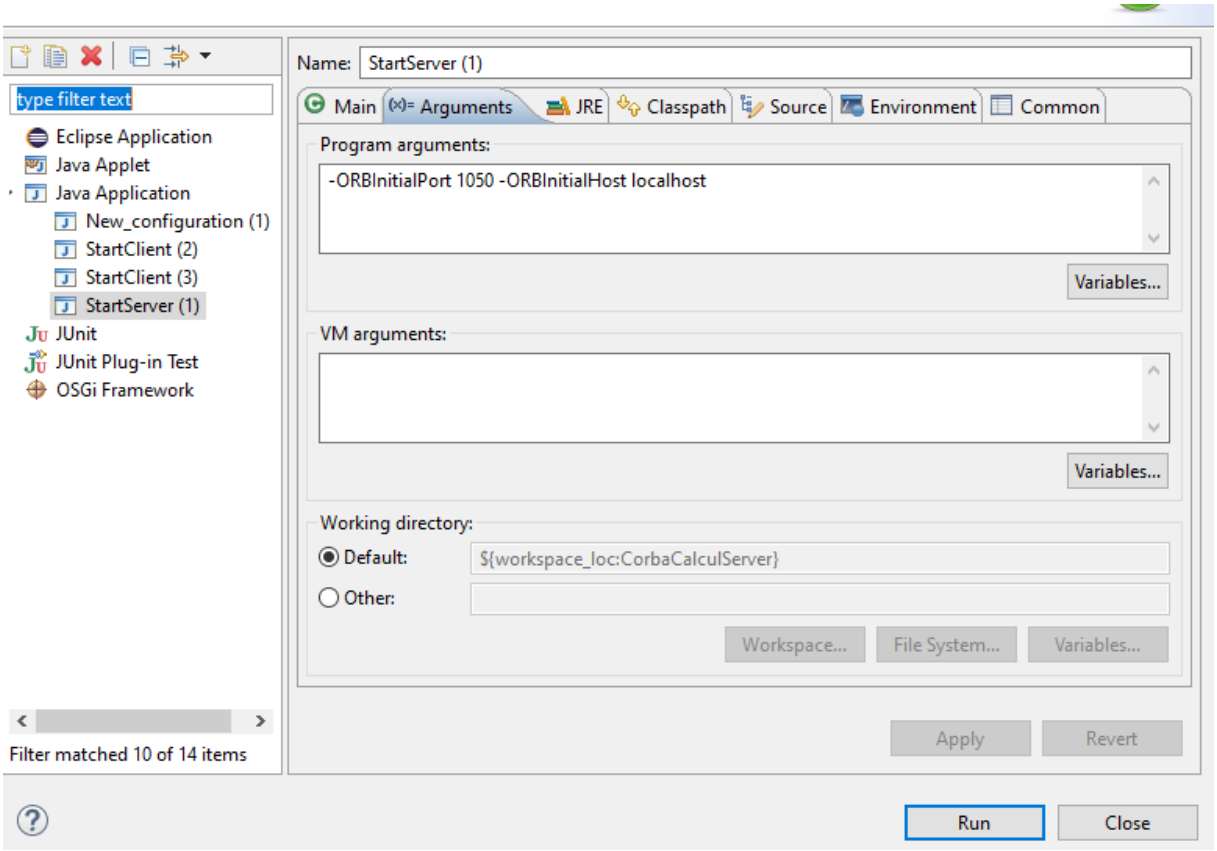
Ouvrir une nouvelle fenêtre de la console CMD, et taper la commande suivante pour démarrer l'ORB.

```
start orbd -ORBInitialPort 1050
```

Étape 9

Sous Eclipse maintenant, cliquer sur le projet CalculServer, puis cliquer sur **Run** -> **Run Configurations** comme indiqué ci-dessous:



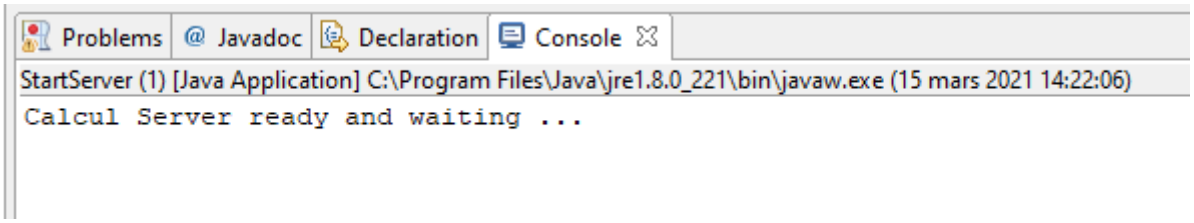


Étape 10

Cliquez sur **Arguments** et tapez les arguments suivants dans les arguments du programme. Cliquez sur **Appliquer** ou “Apply” une fois terminé:

-ORBInitialPort 1050 -ORBInitialHost localhost

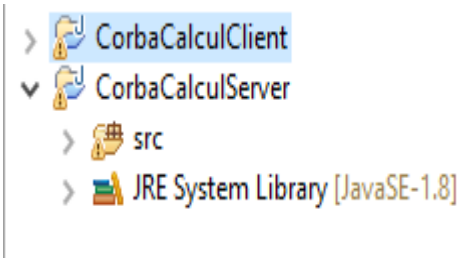
Voici le résultat obtenu si tout se passe bien :



Côté client

Étape 1

Créer un nouveau projet Java avec Eclipse et appeler-le: CorbaCalculClient-> Cliquer sur “Finish” une fois terminé.



Étape 2

Cliquer avec le bouton droit sur Package CalculApp sous l'ancien projet coté serveur (CorbaCalculServer), puis cliquer sur Copier. Ensuite, faire un clic droit sur src sous CorbaCalculClient, puis cliquer sur Coller.

Étape 3

Sous le projet CorbaCalculClient, créer une nouvelle classe appelée StartClient. Copier le code ci-dessous dans la nouvelle classe.

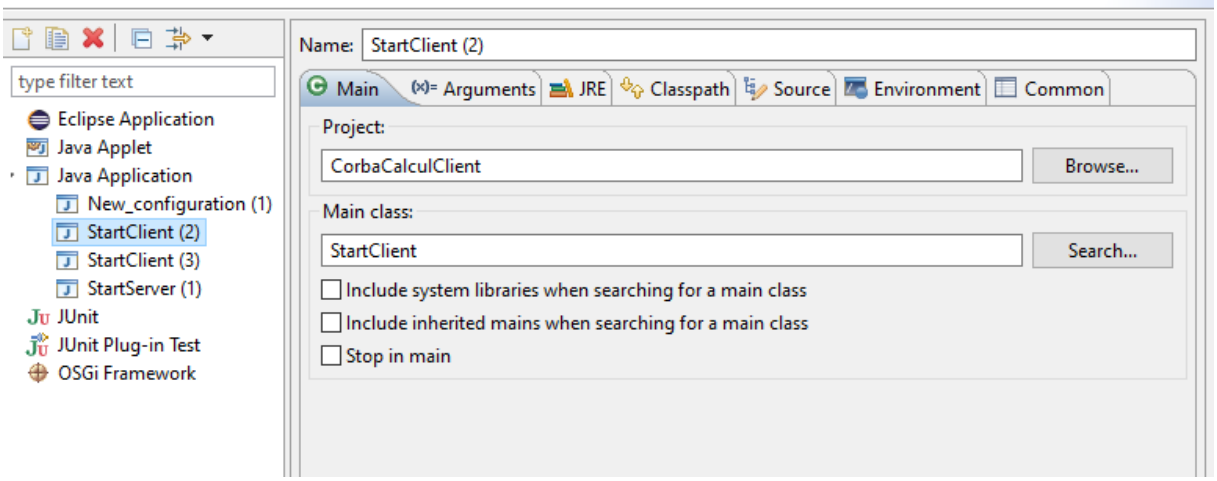
```
import CalculApp.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;import java.io.*;
import java.util.*;
public class StartClient {



 public static void main(String[] args) {
 try {
ORB orb = ORB.init(args, null);

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

 NamingContextExt ncRef =
NamingContextExtHelper.narrow(objRef);

 Calcul addobj = (Calcul)
CalculHelper.narrow(ncRef.resolve_str("ABC"));

 Scanner c=new Scanner(System.in);

 System.out.println("Welcome to the Cacul system:");
 System.out.println("1:for addition");
 System.out.println("2:for substraction");
 System.out.println("3:for division ");
 System.out.println("4:for multiplication");

 for(;;){
```

```
        String choix =c.nextLine();
        int ch=Integer.parseInt(choix);
        if(ch==1){
          System.out.println("Enter a:");String aa =
c.nextLine();
          System.out.println("Enter b:");String bb =
c.nextLine();
          int a=Integer.parseInt(aa);int
b=Integer.parseInt(bb);
          int ra=addobj.add(a,b);
          System.out.println("The result for Addition
is : "+ra);
          System.out.println("----------------------
-----------");
                  }
     if(ch==2){
       System.out.println("Enter a:");String aa =
c.nextLine();
       System.out.println("Enter b:");String bb =
c.nextLine();
       int a=Integer.parseInt(aa);int
b=Integer.parseInt(bb);
          int rs=addobj.sous(a,b);
          System.out.println("The result for
Soustraction is : "+rs);
          System.out.println("----------------------
-----------");
                  }

     if(ch==3){

           System.out.println("Enter a:");String aa =
c.nextLine();
        System.out.println("Enter b:");String bb =
c.nextLine();
        int a=Integer.parseInt(aa);int
b=Integer.parseInt(bb);
        int rd=addobj.div(a,b);
        System.out.println("The result for Division
is : "+rd);
        System.out.println("----------------------
----------");
             }
```

```
    if(ch==4){
        System.out.println("Enter a:");String aa =
c.nextLine();
         System.out.println("Enter b:");String bb =
c.nextLine();
         int a=Integer.parseInt(aa);int
b=Integer.parseInt(bb);
         int rm=addobj.mult(a,b);
         System.out.println("The result for
Multiplication is : "+rm);
         System.out.println("----------------------
-----------");
             }
 } }
 catch (Exception e) {
 System.out.println("Hello Client exception: " + e);
 e.printStackTrace();
 }
 }
}
```

Étape 4

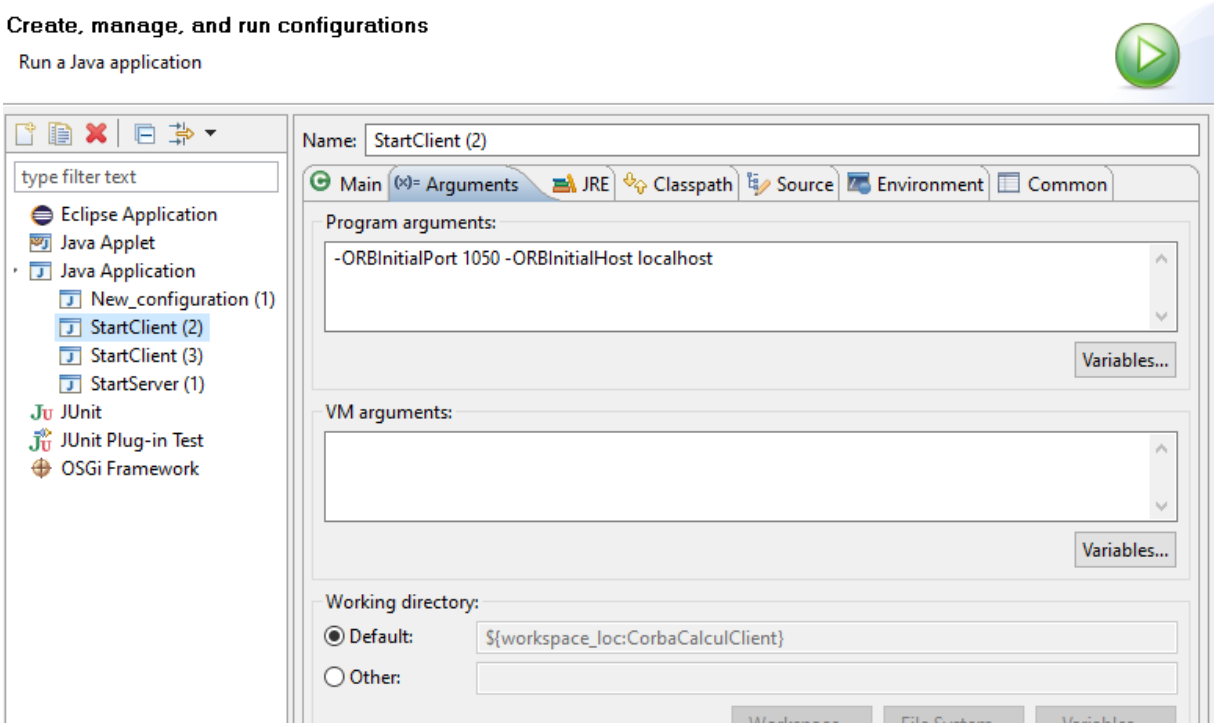
Cliquez sur **Run** -> **Run Configuration**, tout en sélectionnant le StartClient pour le CorbaAdditionClient.



Étape 5

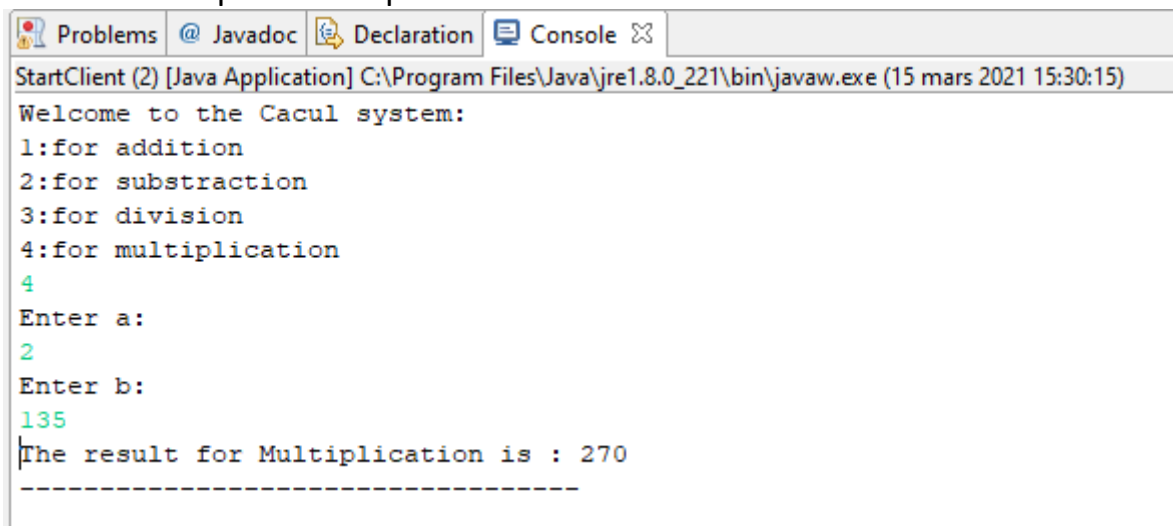
Cliquer sur Arguments, puis sous les arguments du programme, taper ce qui suit:

-ORBInitialPort 1050 -ORBInitialHost localhost



Cliquez sur Appliquer, puis sur Exécuter. C'est tout pour le client.

Voici un exemple de test pour le client



3) <u>Conclusion :</u>

Ce travail nous à parmi de mieux comprendre comment fonctionne corba, l'importance de l'utilisation du langage IDL et du gestionnaire de requêtes ORB sur les applications sous corba. Il nous a permis également de voir la flexibilité de corba en faisans communiqué un client et un serveur qui sont dans des projets différents .