Système Distribué avec CORBA 3

Yassamine Seladji

yassamine.seladji@gmail.com

17 mars 2020

L'architecture des composants CORBA.

- du bus logiciel ORB.
- des souches : stub et squelette.
- du POA (Portable Object Adapter).
- de l'interface IDL.
- des IOR (Interoperable Object Reference).
- Des services.

La mise en place d'une application CORBA

- La définition du contrat IDL.
- La pré-compilation du contrat IDL.
- L'implémentation du serveur.
- L'implémentation du client.
- L'exécution répartie de l'application.

L'implémentation par héritage.

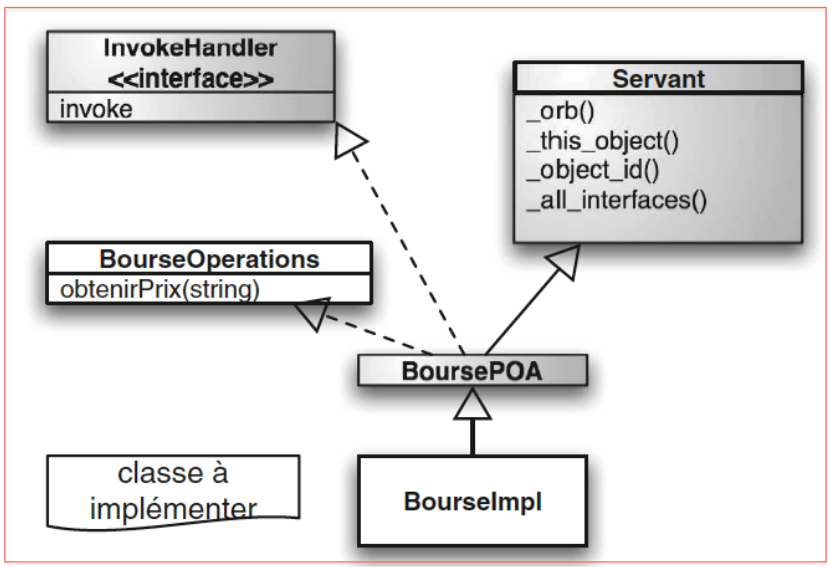
exemple : bourse.idl

```
module bourse {
    interface Bourse {
        double obtenirPrix (in String symbole);
    };
};
```

L'implémentation par héritage.

exemple : bourse.idl

```
module bourse {
    interface Bourse {
        double obtenirPrix (in String symbole);
    };
};
```

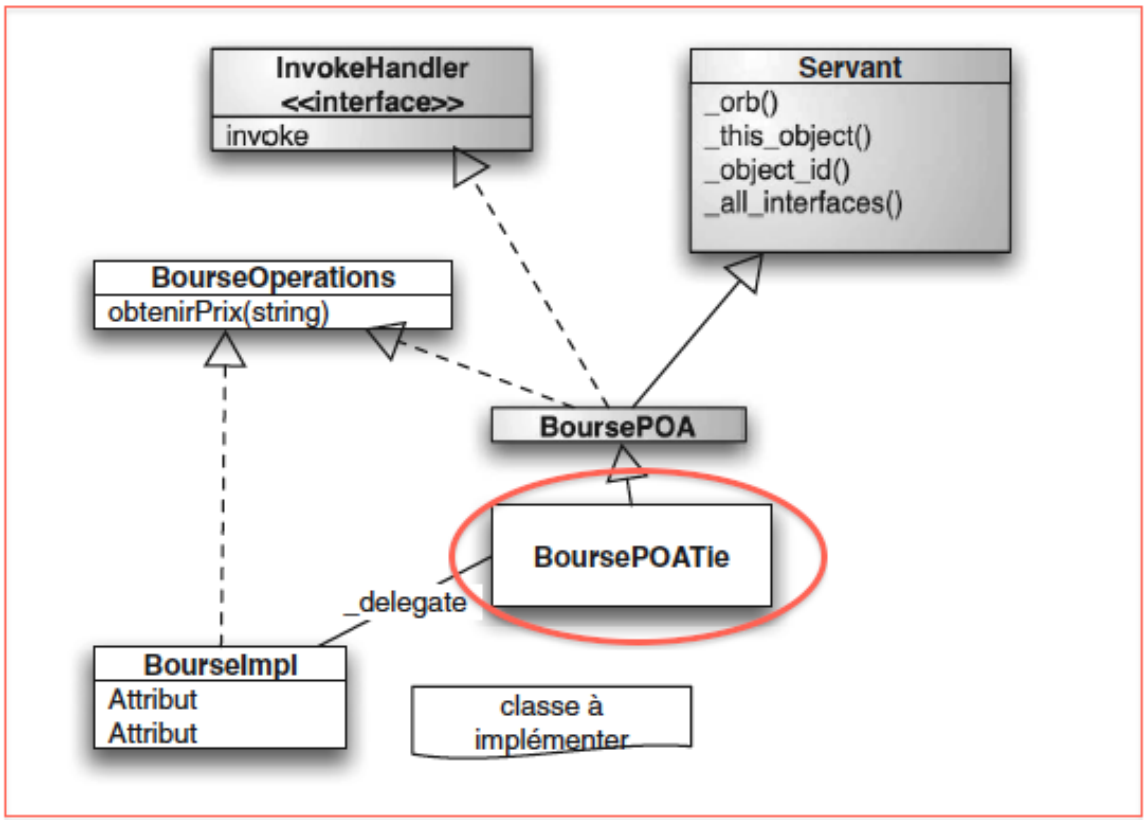


L'implémentation par délégation.

- idlj -allTIE bourse.idl

L'implémentation par délégation.

- idlj -allTIE bourse.idl



L'implémentation par délégation.

```
public class BourseImpl implements BourseOperations {

  public double obtenirPrix(String symbole) {
      return 45.0 ;
  }
}
```

L'implémentation par délégation.

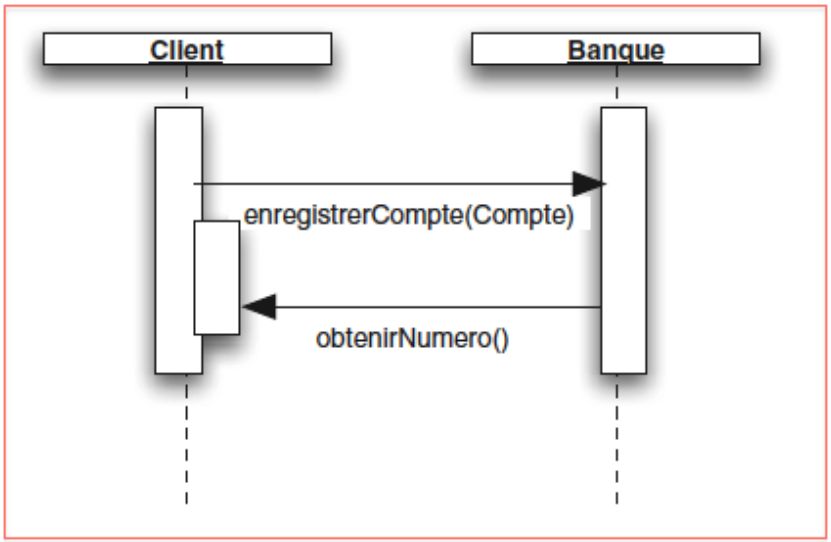
```
public class BourseImpl implements BourseOperations {

  public double obtenirPrix(String symbole) {
      return 45.0 ;
  }
}
```

```
ORB orb = ORB.init(args, props);
BourseImpl bourse = new BourseImpl();

org.omg.CORBA.Object a = orb.resolve_initial_references("RootPOA");
POA rootpoa = POAHelper.narrow(a);
rootpoa.the_POAManager().activate();

// Par delegation
BoursePOATie tie = new BoursePOATie(bourse, rootpoa);
Bourse href = tie._this(orb);
String ior = orb.object_to_string(href);

OutputStream file = new FileOutputStream ("bourse.ior");
DataOutputStream out = new DataOutputStream(file);
out.writeBytes(ior);
out.close();
System.out.println("serveur prêt");
orb.run();
```

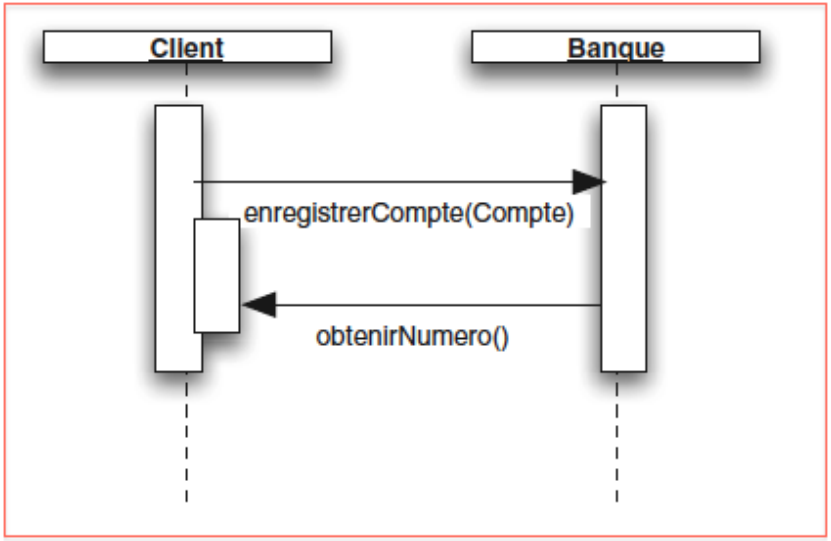
Le passage des paramètres.

- Passage des paramètres par références.
- Passage des paramètres par valeur.

Passage d'arguments par référence.



Passage d'arguments par référence.




```
module finance {
interface Compte {
double obtenirSolde( in string symbol );
long obtenirNumero();
};
interface Banque {
Compte creerCompte( in long numero );
void enregistrerCompte(in Compte compte);
};
};
```

Passage d'arguments par référence.

Coté serveur :

- La pré-compilation du fichier banque.idl :
  idlj -fall banque.idl

Passage d'arguments par référence.

Coté serveur :

- La pré-compilation du fichier banque.idl :
  idlj -fall banque.idl
- l'implémentation de *CompteImpl* :

```
public class CompteImpl extends ComptePOA {
    public int obtenirNumero() {
        return 1234;}
    public double obtenirSolde(String symbol) { return 1000.0; }
}
```

Passage d'arguments par référence.

Coté serveur :

- La pré-compilation du fichier banque.idl :
  idlj -fall banque.idl
- l'implémentation de *CompteImpl* :

```
public class CompteImpl extends ComptePOA {
    public int obtenirNumero() {
        return 1234;}
    public double obtenirSolde(String symbol) { return 1000.0; }
}
```

- l'implémentation de *banqueImpl* :

```
public class BanqueImpl extends BanquePOA {
    private Vector comptes= new Vector(); // Vector est synchronisé
    public Compte creerCompte(int numero) {
        try{
// Création de l'objet Compte
    CompteImpl compte = new CompteImpl();

    Compte referenceCompte =
        CompteHelper.narrow(_poa().servant_to_reference(compte));
    return referenceCompte ;
        }catch(Exception e) {...return null ;}
}
public void enregistrerCompte(Compte compte) {
    comptes.add(compte);
```

Passage d'arguments par référence.

Coté client :

- Initialiser l'ORB :

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

Passage d'arguments par référence.

Coté client :

- Initialiser l'ORB :

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

- Récupérer le POA :

```
POA rootpoa =
    POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
rootpoa.the_POAManager().activate();
```

Passage d'arguments par référence.

Coté client :

- Initialiser l'ORB :

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

- Récupérer le POA :

```
POA rootpoa =
    POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
rootpoa.the_POAManager().activate();
```

- Obtenir la référence de l'objet distant **Banque** :

```
Banque banque=BanqueHelper.narrow(
orb.string_to_object(
    "corbaname:iiop:1.2@localhost:1050#BanqueHasard"));
```

Passage d'arguments par référence.

Coté client :

- Initialiser l'ORB :

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

- Récupérer le POA :

```
POA rootpoa =
    POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
rootpoa.the_POAManager().activate();
```

- Obtenir la référence de l'objet distant **Banque** :

```
Banque banque=BanqueHelper.narrow(
orb.string_to_object(
    "corbaname:iiop:1.2@localhost:1050#BanqueHasard"));
```

- La création et l'utilisation de l'objet **Compte** :

```
CompteImpl compte= new CompteImpl();
rootpoa.activate_object(compte);
Compte ref = CompteHelper.narrow(rootpoa.servant_to_reference(compte));
    banque.enregistrerCompte(ref);
```

Passage d'arguments par valeur.

Coté serveur :

- Dans l'IDL, utiliser le type ValueType

```
module finance {
valuetype Compte {
    long obtenirNumero();
    factory init(in long numero, in double solde);
};
interface Banque {
    Compte creerCompte( in long numero );
};};
```

Passage d'arguments par valeur.

Coté serveur :

- Dans l'IDL, utiliser le type ValueType

```
module finance {
valuetype Compte {
    long obtenirNumero();
    factory init(in long numero, in double solde);
};
interface Banque {
    Compte creerCompte( in long numero );
};};
```

- Pré-compiler l'IDL avec idlj :

Passage d'arguments par valeur.

Coté serveur :

- Dans l'IDL, utiliser le type ValueType

```
module finance {
valuetype Compte {
    long obtenirNumero();
    factory init(in long numero, in double solde);
};
interface Banque {
    Compte creerCompte( in long numero );
};};
```

- Pré-compiler l'IDL avec idlj :
  - La classe *Comple* qui implémente *org.omg.CORBA.portable.StreamableValue*.

Passage d'arguments par valeur.

Coté serveur :

- Dans l'IDL, utiliser le type ValueType

```
module finance {
valuetype Compte {
    long obtenirNumero();
    factory init(in long numero, in double solde);
};
interface Banque {
    Compte creerCompte( in long numero );
};};
```

- Pré-compiler l'IDL avec idlj :
  - La classe *Comple* qui implémente *org.omg.CORBA.portable.StreamableValue*.
  - La classe *CompteDefaultFactory* qui implémente *CompteValueFactory*.

Passage d'arguments par valeur.

Coté serveur :

- implémenter la classe CompteImpl qui hérite de Compte.

Passage d'arguments par valeur.

Coté serveur :

- implémenter la classe CompteImpl qui hérite de Compte.
- implémenter la classe BanqueImpl qui hérite de BanquePOA.

```
public class BanqueImpl extends BanquePOA {
public Compte creerCompte(int numero) {
    CompteDefaultFactory factory= new CompteDefaultFactory();
    return factory.init(numero, 0.0);
} }
```

Passage d'arguments par valeur.

Coté client :

- La même implémentation que pour le passage par référence.

Passage d'arguments par valeur.

Coté client :

- La même implémentation que pour le passage par référence.

Remarque :

- CORBA n'utilise pas la sérialisation Java.
- Ajouter les méthodes de sérialisation pour CORBA.

Passage d'arguments par valeur.

Coté client :

- La même implémentation que pour le passage par référence.

Remarque :

- CORBA n'utilise pas la sérialisation Java.
- Ajouter les méthodes de sérialisation pour CORBA.
-

```
public void _read (org.omg.CORBA.portable.InputStream istream) {
    numero=istream.read_long();} // sérialisation du numéro seul
public void _write (org.omg.CORBA.portable.OutputStream ostream) {
    ostream.write_long(numero);}
```

Conclusion.

- CORBA permet de mettre en place une communication logique entre deux applications.
- Les applications peuvent être implémenter en des langages différents.
- CORBA est constitué de plusieurs composant.
- La mise en place de CORBA est simplifiée par l'OMG.