Tutoriel d'initiation à OpenORB

F. Bernardi, Université de Corse, 2002

OpenORB se trouve aux adresses suivantes:
`http://openorb.sourceforge.net`
`http://umn.dl.sourceforge.net/sourceforge/openorb/OpenORB-1.3.0.tgz`

1 Installation d'OpenORB 1.3.0

Récupérez l'archive à l'adresse suivante (version 1.3.0 à la date d'écriture de l'article) et sauvegardez la dans votre répertoire personnel en utilisant les commandes suivantes:

```
# cd ~
# wget http://umn.dl.sourceforge.net/sourceforge/openorb/OpenORB-1.3.0.tgz
```

Désarchivez l'archive dans votre répertoire personnel et renommez le dossier ainsi créé :

```
# cd ~
# tar xzf OpenORB-1.3.0.tgz
# mv OpenORB-1.3.0 OpenORB
```

Vérifiez que la variable d'environnement `JAVA_HOME` est positionnée:

```
# echo $JAVA_HOME
/usr/java/j2sdk1.4.1_01
```

Si elle n'est pas positionnée:

```
# export JAVA_HOME=<chemin vers la JRE Java>
```

Positionnez la variable `CLASSPATH` de manière à ce qu'elle pointe vers tous les fichiers `.jar` du répertoire `lib` du répertoire d'OpenORB:

```
# for i in ~/OpenORB/lib/*.jar
> do
> export CLASSPATH=$CLASSPATH:$i
> done
```

Vérifiez la valeur de cette variable:

```
# echo $CLASSPATH
```

Entrez les commande suivante:

```
# cd ~/OpenORB/lib
# su
# java -jar OpenORB/lib/openorb-1.3.0.jar
# exit
```

Vérifiez par ailleurs que le nom de la machine apparaît dans le fichier /etc/hosts.
Ceci permet de spécifier au compilateur et à l'interpréteur Java que l'on va utiliser OpenORB et non pas l'ORB integré au JDK. La commande `su` permet de basculer vers l'utilisateur "root".
Ouvrez un terminal, placez vous dans un répertoire quelconque et lancez la commande du compilateur IDL:

```
# java org.openorb.compiler.IdlCompiler
```

Si tout s'est bien passé, une liste d'options doit s'afficher. Dans le cas contraire, répétez les étapes précédentes en vérifiant les valeurs.

2 Le compilateur IDL vers Java

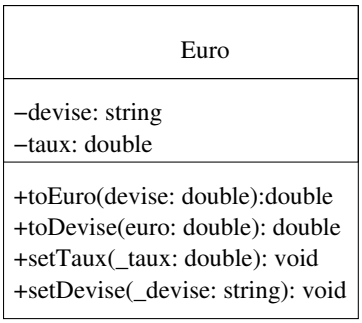
Le compilateur IDL vers Java est appelé par la commande suivante:
`# java org.openorb.compiler.IdlCompiler <fichier.idl>`
Les options principales de ce compilateur sont les suivantes:

- `-boa`: générer les skeleton pour utiliser le BOA;
- `-d <rep>`: spécifier un répertoire de destination. Par défaut, le répertoire est nommé "`generated`";
- `-noskeleton`: ne pas générer de skeleton;
- `-nostub`: ne pas générer de stub;
- `-package <package_name>`: spécifier un package de destination;
- `-jdk1.4`: générer des classes pour le JDK 1.4
- `-notie`: ne pas générer les classes pour l'approche par délégation

Par défaut, les fichiers sont générés pour l'utilisation du POA.

3 Exemple: un convertisseur euro simple avec POA

Le but de cet exemple est de créer une implémentation CORBA d'un objet `Euro` défini par le diagramme UML suivant:



Commencez par créer un répertoire de travail appelé CORBA dans votre répertoire personnel. Créez à l'intérieur un répertoire `conv_euro`.

Première étape: écrire la description IDL du convertisseur et effectuer la projection en Java.

Créez un fichier `conv_euro.idl` contenant le code suivant:

```
module Convertisseur {
  interface Euro {
    double toEuro(in double devise);
```

```
    double toDevise(in double euro);
    void setTaux(in double _taux);
    void setDevise(in string _devise);
  };
};
```

Effectuez la projection en Java en utilisant le POA (Portable Object Adapter):
`# java org.openorb.compiler.IdlCompiler conv_euro.idl -notie`
Cette commande crée un répertoire `generated` contenant un répertoire `Convertisseur` correspondant au module défini en IDL. Ce répertoire contient les fichiers générés suivants:

- `Euro.java`: définition de l'interface `Euro`;
- `EuroHelper.java`: méthodes pour faciliter l'utilisation des objets `Euro` (lire un objet, le convertir en utilisant la méthode `narrow`...);
- `EuroHolder.java`: méthodes pour prendre en charge le passage de paramètres avec les méthodes de l'objet `Euro`.
- `EuroOperations.java`: interface présentant les méthodes de l'objet `Euro` (Ce dernier hérite de cette interface);
- `EuroPOA.java`: skeleton pour l'objet `Euro`;
- `_EuroStub.java`: stub pour l'objet `Euro`;

Notez que tous ces fichiers font partie du package `convertisseur` créé par la projection du module IDL `Convertisseur`.

Deuxième étape: Définir la classe de l'objet que nous voulons mettre à disposition des clients.

L'implémentation de l'objet serveur sera construit par spécialisation de la classe `EuroPOA`. Créez une nouvelle classe `EuroPOAImpl` (et donc un nouveau fichier `EuroPOAImpl.java`) contenant le code suivant:

```
// Implémentation de l'objet Euro avec le POA
import Convertisseur.*;
public class EuroPOAImpl extends EuroPOA {
  private String devise;
  private double taux = 6.55957;
  public double toEuro(double devise) {
    return devise / taux;
  }
  public double toDevise(double euro) {
    return euro * taux;
  }
  public void setTaux(double _taux) {
    taux = _taux;
  }
  public void setDevise(String _devise) {
```

```
    devise = _devise;
  }
}
```

Troisième étape: réalisation du serveur CORBA.

La réalisation du serveur CORBA correspond à la définition d'une application principale qui contiendra un objet `Euro`. Créez une nouvelle classe `ServerPOA` (et donc un nouveau fichier `ServerPOA.java`) contenant le code suivant:

```
// Serveur utilisant le POA
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
import org.omg.CORBA.ORBPackage.*;
import java.io.*;
public class ServerPOA {
  public static void main(String args[]) {
    // Initialisation de l'ORB
    ORB orb = ORB.init(args, null);
    // Initialisation du POA
    org.omg.CORBA.Object objPoa = null;
    POA rootPOA = null;
    try {
      objPoa = orb.resolve_initial_references("RootPOA");
    } catch (InvalidName e1) {}
    // Opération de narrow sur l'objet obtenu
    rootPOA = POAHelper.narrow(objPoa);
    // Instanciation d'un objet Euro
    EuroPOAImpl euro = new EuroPOAImpl();
    try { // Activation du servant
      byte[] servantId = rootPOA.activate_object(euro);
      // Récupération de la référence du servant
      org.omg.CORBA.Object obj = rootPOA.id_to_reference(servantId);
      // Exportation de la référence
      String reference = orb.object_to_string(obj);
    try {
      FileOutputStream file = new FileOutputStream("ObjectId");
      PrintStream pfile = new PrintStream(file);
      pfile.println(reference);
    } catch (IOException e2) {}
      // Activation du POA par son manager
      rootPOA.the_POAManager().activate();
      System.out.println("Serveur en attente...");
      // Lancement de l'ORB
      orb.run();
    } catch (Exception e3) {
    System.out.println("Exception !");
```

```
    e3.printStackTrace();
    }
  }
}
```

Quatrième étape: réalisation du client CORBA.

De manière à récupérer la référence sur l'objet utilisé, le client va utiliser le fichier `ObjectId` créé par le serveur CORBA. Créez une nouvelle classe `Client` (et donc un nouveau fichier `Client.java`) contenant le code suivant:

```
// Client CORBA
import Convertisseur.*;
public class Client {
  public static void main(String args[]) {
    // Initialisation de l'ORB
    org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);
    // Récupération de la référence de l'objet Euro
    Euro euro = null;
    org.omg.CORBA.Object obj = null;
    try {
      java.io.FileInputStream file = new java.io.FileInputStream("ObjectId");
      java.io.DataInputStream myInput = new java.io.DataInputStream(file);
      String stringTarget = myInput.readLine();
      obj = orb.string_to_object(stringTarget);
    } catch (java.io.IOException e1) {}
    // Opération de narrow sur la référence de l'objet
    euro = EuroHelper.narrow(obj);
    // Utilisation de l'objet euro
    try {
      System.out.println("1 franc = " + euro.toEuro(1) + " euros");
      System.out.println("1 euro = " + euro.toDevise(1) + " francs");
    }
    catch (org.omg.CORBA.SystemException e3) {
    System.out.println("Exception CORBA !");
    System.out.println(e3.getMessage());
    }
  }
}
```

Cinquième étape: compilation et exécution de l'application.

Sauvegardez les fichiers `Client.java`, `EuroPOAImpl.java` et `ServerPOA.java` dans le répertoire `~/CORBA/conv_euro` et lancez la compilation par les commandes suivantes:

```
# cd ~/CORBA/conv_euro
# export CLASSPATH=$CLASSPATH:./generated/:.
# cd generated/Convertisseur
# javac *.java
```

```
# cd ../..
# javac Client.java ; javac EuroPOAImpl.java ; javac ServerPOA.java
```

Ouvrez un nouveau terminal et placez vous dans `~/CORBA/conv_euro`. Dans ce nouveau terminal lancez les commandes:

```
# for i in ~/OpenORB/lib/*.jar
> do
> export CLASSPATH=$CLASSPATH:$i
> done
# cd ~/CORBA/conv_euro
# export CLASSPATH=$CLASSPATH:./generated/:.
# java ServerPOA
```

et dans l'ancien terminal qui a servi à compiler les fichiers:

```
# cd ~/CORBA/conv_euro
# java Client
```