M1 Programmation avancée et répartie en Java : TP 8

Jean Fortin et Frédéric Gava
gava@u-pec.fr ou jean.fortin@gmail.com

1 Exercice 1 : Hello World en RMI

1. Dans un dossier tp8 (dans un src habituel), créer un fichier `HelloInterface.java`, qui définit l'interface suivante :

```
package tp8;
import java.rmi.*;
public interface HelloInterface extends Remote {
  public String say() throws RemoteException;
}
```

2. Dans un fichier `Hello.java`, définir la classe suivante :

```
package tp8;
import java.rmi.*;
import java.rmi.server.*;

public class Hello extends UnicastRemoteObject implements HelloInterface {
  private String message;
  public Hello (String msg) throws RemoteException {
    message = msg;
  }
  public String say() throws RemoteException {
    return message;
  }
}
```

3. Compiler les deux fichiers ci-dessus. Depuis le dossier contenant src et class,

```
javac -d build src/tp8/*.java
```

4. Compiler le rmi avec la commande

```
rmic -classpath build -d build tp8.Hello
```

5. Créer l'application serveur `HelloServer.java`

```
package tp8;
import java.rmi.Naming;

public class HelloServer
{
 public static void main (String[] argv) {
  try {
```

```
    Naming.rebind ("Hello", new Hello ("Hello World!"));
    System.out.println("Server is connected and ready for operation.");
   }
   catch (Exception e) {
    System.out.println ("Server not connected: " + e);
   }
  }
}
```

6. Créer l'application client, HelloClient.java :

```
package tp8;
import java.rmi.Naming;

public class HelloClient {
 public static void main (String[] argv) {
  try {
   HelloInterface hello =(HelloInterface) Naming.lookup ("//127.0.0.1/Hello");
   System.out.println (hello.say());
  }
  catch (Exception e){
   System.out.println ("HelloClient exception: " + e);}
  }
}
```

7. Compiler les fichiers
8. Ouvrir un nouveau terminal, se placer dans le dossier "class", et lancer le programme "rmiregistry".
9. Ouvrir un nouveau terminal, se placer dans le dossier racine, et lancer le serveur.

   `java -cp build tp8.HelloServer`

10. Ouvrir un nouveau terminal, se placer dans le dossier racine, et lancer le client.

    `java -cp build tp8.HelloClient`

11. (Si tout marche bien) Modifier le client pour qu'il se connecte au serveur de votre voisin, et tester (vous pouvez personnaliser les messages).
12. Analyser les programmes pour en comprendre le fonctionnement.

2 Exercice 2 : Une base de licenciés en RMI

Dans cet exercice, on va définir un service de gestion d'une base de licenciés d'une fédération sportive. Pour cela, on utilisera deux classes avec des méthodes distantes, la classe `Licencie` et la classe `Base`.

1. Définir une classe `Licencie`, ayant comme champs privés un identifiant (entier), un nom (String), et une ligue (String). Les variables privées sont initialisées dans le constructeur. On définira des méthodes getNom, setNom, getLigue, setLigue et getID.
2. Définir une classe `Base`, contenant un ensemble de licenciés, sous la forme d'un `HashMap <Integer,Licencie>` (l'entier sert d'identifiant unique pour le `Licencie`), initialement vide, avec des méthodes :

- `int ajouteLicencie(String nom, String ligue)`, qui ajoute un licencié dans la base, et renvoie son identifiant
- `void effaceLicencie(int id)`, qui efface un licencié de la base
- `Licencie getLicencie(int id)`, qui renvoit le licencié à partir de son id.

3. Définir les interfaces correspondants aux services pour les classes `Licencie` et `Base`. (on incluera toutes les méthodes ci-dessus).
4. Définir les classes `LicencieExporte` et `BaseExportee`
5. Définir la classe `Serveur`, qui enregistre le service de la base sous le nom "BaseLicencies".
6. Définir la classe `Client`, qui se connecte à la base, ajoute 2 licenciés, puis modifie le nom du premier, et vérifie que la modification a bien été effectuée.
7. Tester.

3 Exercice 3 : JDBC - Sqlite

Exercice à effectuer sous Linux. Copier le fichier jdbc.java de la page de cours. Suvre le liens vers tutorialspoint. Et essayez les exemples proposés. Essayer de faire vous même une petite BD et d'y accèder comme dans l'exemple "jdbc.java" avec Sqlite.

4 Exercice 4 : JDBC - Oracle

Créer une application graphique permettant d'exécuter une requête SELECT sur une base de donnée oracle et d'afficher le résultat. Pour cela, utilisez le driver sun du fichier "jdbc.java".