TP1 : Java RMI

PAR : Programmation d'Applications Réparties
Troisième année

1 Un premier exemple : conversions d'unités

L'objectif est d'écrire une application simpliste (et faire ainsi vos premiers pas avec les Java RMI). Il s'agit d'un service de conversion entre degrés Celsius et Fahrenheit. On rappelle les formules suivantes :

$$x \deg^c = (x * \frac{9}{5} + 32) \deg^f \qquad y \deg^f = ((y - 32) * \frac{5}{9}) \deg^c$$

▷ **Question 1.** Écrivez dans l'ordre :
- L'interface `ConvInterface` dans un fichier `ConvInterface.java`
- La classe `Conv` implémentant le service dans le fichier `Conv.java`
- Le serveur dans un fichier `ConvServer.java`
- Le client dans un fichier `ConvClient.java`

▷ **Question 2.** Comment compiler votre projet ? Testez votre service d'abord en local puis en plaçant le serveur sur un hôte distant et remplissez le tableau suivant :

°C	0	10	100	180	1000					
°F						0	10	100	500	1000

Indications :
- Pour pouvoir utiliser plusieurs instances de `rmiregistry` sur une seule machine (par exemple lorsque plusieurs utilisateurs sont sur la même machine), il faut changer le numéro de port utilisé (utilisez un numéro de port original et non 4321).
- Pour cela, lancez-le ainsi : `rmiregistry 4321 &`
- Ensuite, modifiez l'appel à `Namming.lookup` dans le client et celui à `Namming.rebind` dans le serveur pour utiliser l'adresse suivante `"rmi://localhost:4321/Conv1"`

2 Un service de dictionnaire

Il s'agit maintenant d'implémenter un service de dictionnaire. On souhaite retrouver à la fois si un mot fait partie du dictionnaire ou non et sa définition. On utilisera pour cela l'interface java.util.Map et son implémentation TreeMap.
- Insérer un mot : `data.put("mot", "définition") ;`
- Retrouver une définition : `data.get("mot") ;`

▷ **Question 3.** Écrivez un service (interface, implémentation, serveur et client) permettant d'insérer un mot, d'en supprimer et de rechercher des définitions du dictionnaire à distance.
▷ L'insertion d'un mot déjà présent dans le dictionnaire et la recherche d'un mot absent doivent provoquer respectivement les exceptions ExisteDeja et PasTrouve que vous définirez.

3 Dictionnaires multiples et fabriques

On souhaite maintenant offrir la possibilité d'utiliser des dictionnaires stockés sur disques pour initier le contenu. Le format de fichier utilisé consistera simplement à placer chaque mot et sa définition sur une ligne séparée, en les séparant par des espaces. Vous trouverez des fichiers d'exemple sur la page `http://www.loria.fr/~quinson/teach-PAR.html`.

Exemple de dictionnaire

```
1 RPC (Remote Procedure Call) Système inventé dans les années 80 pour le calcul distribué
2 RMI (Remote Method Invocation) Variante des RPC pour les objets, utilisée en Java
3 NFS (Network File System) Procédé permettant l'usage d'un disque dur à distance (basé sur RPC)
```

Voici une fonction permettant de charger un fichier utilisant un tel format dans une structure Map.

Dict.java

```
import java.io.*;
import java.lang.*;
import java.util.*;

public class Dict {
 private Map data;
 public Dict(String fich) throws java.io.IOException {
  BufferedReader in = new BufferedReader(new InputStreamReader(new FileInputStream(fich)));
  String text;
  while( (text = in.readLine()) != null) {
   data.put(text.substring(0,text.indexOf(" ")), (text.substring(text.indexOf(" "))).trim());
}}}
```

▷ **Question 4.** Ajoutez ce constructeur à la classe implémentant les dictionnaires du deuxième exercice, et écrivez une fabrique de dictionnaires (interface et classe) permettant de spécifier le fichier utilisé. Voici un exemple de client utilisant cette fonctionnalité.

DictFabClient.java

```
import java.rmi.* ;
public class DictFabClient {
  public static void main (String[]argv){
    System.setSecurityManager (new RMISecurityManager ()) ;
    try {
      DictFabInterface fab = (DictFabInterface) Naming.lookup("rmi://localhost:4321/DictFab1") ;

      DictInterface dict   = (DictInterface) fab.newDict("data/dict.little");

      System.out.println ("Recherche RPC: "+dict.rechercher("RPC")) ;

      dict = (DictInterface) fab.newDict("data/dict.en");

      System.out.println ("Recherche aqueduct: "+dict.rechercher("aqueduct")) ;
      dict.supprimer("corsair");
    } catch (Exception e) {
      System.out.println ("Erreur client : " + e) ;
    }
  }
}

```

4 Serveur de discussion et callbacks

L'objectif de cet exercice est d'écrire un serveur de groupe de discussion. Une fois que les clients s'y sont connectés, ils peuvent envoyer des chaînes de caractères que le serveur se charge de faire suivre aux autres clients.

▷ **Question 5.** Écrivez un service de discussion où les clients peuvent envoyer des chaînes de caractères (lues au clavier) au serveur, qui les affiche sur sa sortie standard.

▷ **Question 6.** Ajoutez au client une méthode `Remote` lui permettant de recevoir des chaînes de caractères d'un hôte distant et de les afficher à l'écran.

▷ **Question 7.** Les clients doivent maintenant se connecter au serveur au début de la session et fournir un nom d'utilisateur. De même, ils doivent se déconnecter en fin de session.

▷ **Question 8.** Modifiez le serveur pour qu'il fasse suivre les messages reçus à chaque client connecté.

▷ **Question 9.** Testez votre service, d'abord en local puis en plaçant le serveur sur une machine distante.

▷ **Question 10.** Pourquoi votre client ne marche-t-il pas avec le serveur de votre voisin ? Que faut il changer pour y remédier ?