TP 4 : Appel de procédure distante

Un appel de procédure distante (RPC) permet à un programme informatique de faire exécuter un sous-programme ou une procédure dans un autre espace d'adressage (généralement sur un autre ordinateur d'un réseau partagé) sans que le programmeur ne code explicitement les détails de cette interaction à distance.

Dans ce TP, vous utiliserez le package Java RMI pour la gestion des objets distribués. Vous allez écrire des applications distribuées Java à Java, dans lesquelles les méthodes des objets Java distants peuvent être appelées à partir d'autres machines virtuelles Java, éventuellement sur des hôtes différents.

Objectifs

Après avoir terminé ce TP, vous pourrez:

- Programmer une interface pour les objets distants.
- Implémenter une application serveur.
- Enregistrement et utilisation d'un objet distant.
- Développer des applications distribuées Java à Java.

Exercice1 : Gestion de compte

Les opérations de gestion de compte nécessitent l'implémentation les trois méthodes suivantes : void debiter(double montant), void crediter(double montant) et double lire-solde().

1. Programmer l'interface qui rend chacune de ces méthodes accessibles à distance depuis la machine cliente.

```
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface CompteInterface extends java.rmi.Remote
{
 void debiter(double montant)
 throws java.rmi.RemoteException;
 void crediter(double montant)
 throws java.rmi.RemoteException;
 double lire_solde()
 throws java.rmi.RemoteException;
};
```

2. Déduire la classe qui matérialise le service qui offre les opérations debiter(), crediter() et lire-solde().

```
import java.rmi.*;
import java.rmi.server.*;
public class Compte extends UnicastRemoteObject implements
CompteInterface
{
 private double solde;
 public Compte(double s) throws java.rmi.RemoteException
 {
 super();
 solde=s;
```

```
}

public void crediter(double montant)
throws java.rmi.RemoteException
{
solde=solde+montant;
}
public void debiter(double montant)
throws java.rmi.RemoteException
{
solde=solde-montant;
}
public double lire_solde()
throws java.rmi.RemoteException
{
return solde;
}
}
```

3. Programmer la classe serveur.java pour permettre l'enregistrement du service auprès de RMI Registry.

```
import java.rmi.*;
import java.rmi.server.*;
public class Serveur {
public static void main(String[] args)
{
 try {
 System.out.println("Serveur : Construction de l'implémentation");
 Compte cpt= new Compte(15.50);
 System.out.println("Objet Compte enregistré dans RMIregistry");

 Naming.rebind("//localhost/CompteCourant", cpt);

 System.out.println("Attente des invocations des clients ");
 }
 catch (Exception e) {
 System.out.println("Erreur de liaison de l'objet Compte");
 System.out.println(e.toString());
 }
 } // fin du main
} // fin de la classe
```

4. Programmer la classe Client.java

```
import java.io.*;
import java.rmi.*;
class Client
{
public static void main (String [] argv) throws IOException
{
 if(argv.length != 2){
 System.out.println("Usage : java Client <nombre> <operation>");
 System.exit(1);
 }
// operation = 1: credit, 2: dedit
```

```
System.setSecurityManager(new RMISecurityManager());

double valeur = Double.parseDouble(argv[0]);
int operation = Integer.parseInt(argv[1]);

try {
CompteInterface cpt= (CompteInterface) Naming.lookup("//localhost/CompteCourant");
if (operation==1) cpt.crediter(valeur);
if (operation ==2) cpt.debiter(valeur);
System.out.println ("Votre solde courant = " +
cpt.lire_solde() + " euros");


}catch (Exception e) {
System.out.println("Erreur d'acces a un objet distant");
System.out.println(e.toString());
}
}
}
```

5. Nous devons démarrer le service de noms **rmiregistry** qui va accueillir les références des objets distribués. Lancez **startServer.bat** ou ouvrez une fenêtre de commande et exécutez: ***start emiregistry***

   Mais pour cela, il doit utiliser une politique de sécurité définie dans un fichier, par exemple fichier.policy.
   fichier.policy
   grant {
   permission java.security.AllPermission;
   };
   A la lecture de fichier.policy, on déduit que rmiregistry accepte une requête de n'importe quel programme client, sans restriction sur les numéros de port. L'activation du service de noms nécessite de spécifier le fichier de sécurité utilisé.
   Machine_serveur>start rmiregistry -J-Djava.security.policy=fichier.policy

Exercice 2 : Opérations arithmétiques.

Description :
Dan cet exercice, vous déclarez l'interface distante du serveur pour toutes les opérations de calcule en étendant l'interface distante. Définissez les opérations de calcule de base telles que la somme, la différence, le produit et la division en étendant **UnicastRemoteObject**.
Une interface est le point d'interaction avec un logiciel ou du matériel informatique, ou avec des périphériques tels qu'un écran d'ordinateur ou un clavier. Certaines interfaces informatiques, telles que les écrans tactiles, peuvent envoyer et recevoir des données, tandis que d'autres, comme une souris ou un microphone, ne peuvent envoyer que des données.

Programme :
1. Créez une interface nommant **Display Int permettant** au client de lister les services (méthodes) offerts par le serveur. Chaque méthode distante dans l'interface lève une exception **java.rmi.remote**.
2. Définissez la classe qui implémente l'interface de l'objet distant en élargissant la classe Objet distant **Java.rmi.unicast**.
3. Créez le programme serveur, qui contient la méthode utilisée pour lier le nom à l'objet serveur distant.

4. Écrivez le code client. Le programme client consiste généralement en un simple code Java. Dans le programme client, nous devons utiliser la méthode de recherche disponible dans **java.rmi**.
5. Nommez la classe pour localiser un objet distant.
6. Compilez tous les fichiers java.
7. Créez un **stub** et un **skeleton** à l'aide du nom de fichier du compilateur **rmic Impl**.
8. Démarrez le registre RMI pour nous permettre d'enregistrer l'objet serveur. Le registre RMI est lancé à l'aide de la commande.
9. Exécutez le programme du serveur.
10. Exécutez le programme client dans une invite de commande séparée à l'aide de la commande.

Exercice 3 : Programme Hello World distribué

Écrivez un programme Hello World distribué en deux parties, une application serveur RMI et une applet client. L'applet effectue un appel de méthode à distance au serveur à partir duquel il a été téléchargé pour extraire le message "Hello World!". Lorsque l'applet s'exécute et que le serveur est accessible, le message est affiché sur le client.

Etape de programmation:

1. Définir une interface d'accès:

```
interface publique Hello étend java.rmi.Remote {
String sayHello () soulève java.rmi.RemoteException;
}
```

2. Ecrire une classe implémentant le serveur

Pour écrire un objet distant, vous écrivez une classe qui implémente une ou plusieurs interfaces d'accès distant.

La classe d'implémentation doit:

- Spécifiez la ou les interfaces distantes à implémenter.
- Définissez le constructeur de l'objet distant.
- Fournir des implémentations pour les méthodes pouvant être appelées à distance.
- Créer et installer un gestionnaire de sécurité.
- Créez une ou plusieurs instances d'un objet distant.
- Enregistrez au moins l'un des objets distants auprès du registre d'objets distants RMI, à des fins d'amorçage.

3. Ecrivez une applet client qui utilise le service distant. La partie applet de l'exemple Hello World distribué appelle à distance la méthode **sayHello**() de HelloServer afin d'obtenir la chaîne "Hello World!", Qui est affichée.

Exercice 4 : Enregistreur d'événements distribute (event logger)

Écrivez un programme d'enregistreur d'événements distribué comportant deux éléments: une application serveur RMI et un applet client. L'applet fait un appel de méthode à distance au serveur pour noter un événement. La méthode a un argument de type String qui est la description de l'événement. Sinon, le client peut appeler une méthode distante sur le serveur pour envoyer des messages à la liste du serveur. L'applet a deux parties. La premiere est utilisé pour créer des journaux et le second pour obtenir une liste.

Etape de programmation:

1. Définir une interface d'accès distante:

```
public interface eEventLogger extends java.rmi.Remote
{
void makeNote(String note) throws java.rmi.RemoteException;
String getList() throws java.rmi.RemoteException;
}
```

2. Écrire une classe implémentant de serveur. Pour écrire un objet distant, vous écrivez une classe qui implémente une ou plusieurs interfaces distantes.
3. Écrivez une applet cliente qui utilise le service distant. L'applet appelle à distance les méthodes **getList ()** et **makeNote ()** de serveur.

Exercice 5 : Système de chat

Écrivez un système de discussion basé sur Java RMI. Les utilisateurs discutant avec leur client de discussion, qui reçoit des messages de clients de discussion distants et les affiche dans une fenêtre de texte à côté de leur nom.

Exercice 6 : Tableau blanc distribué

Écrivez un tableau blanc distribué basé sur Java RMI où les utilisateurs peuvent dessiner dans le même tableau en même temps via une connexion réseau.