TP RMI

L'objectif de ce TP est l'initiation à **JAVA RMI** (Remote Method Invocation) qui permet de créer des applications réseau aux caractéristiques suivantes :

1. Les applications client/serveur sont des applications Java aux deux extrémités de la communication.

2. Le client peut utiliser des objets situés sur le serveur comme s'ils étaient locaux.

3. La couche réseau devient transparente : les applications n'ont pas à se soucier de la façon dont sont transportées les informations d'un point à un autre.

Exercice 1 : Initiation à Java RMI

Pour y voir plus clair, nous allons suivre pas à pas l'écriture d'une application client/serveur utilisant le package RMI de Java. Nous prenons un exemple simple ; le client invoque une méthode d'un objet distant qui lui renvoie alors une chaîne de caractères à afficher à l'écran. Le client peut invoquer une autre méthode qui fait la somme de deux entiers. Le scénario est le suivant :

- **Coté Serveur :**

1. Définir une interface héritant de l'interface *java.rmi.Remote* et déclarant une méthode qui doit faire l'écho d'un message donné en paramètre.

2. Définir une deuxième interface héritant de l'interface *java.rmi.Remote* et déclarant une méthode qui doit faire la somme de deux entiers donnés en paramètre.

3. Définir une classe Serveur implémentant les interfaces déjà créées et héritant de la classe *UnicastRemoteObject* qui contient les différentes méthodes autorisant l'invocation de méthodes à distance.

4. Créer une instance de cette classe dans la méthode *main* et l'enregistrer dans l'annuaire des objets accessibles de l'extérieur.

- **Coté Client :**

On vous demande d'écrire une classe Client à qui on passe en paramètre l'URL du serveur précédent et qui peut appeler les deux méthodes du serveur:

1. Permet d'afficher le message « Hello world from RMI server »

2. Permet de faire l'opération suivante : 129+45

Exercice 2 :

Nous disposons des services implantés lors des exercices des TPs précédents : un qui permet d'identifier si un mot est un palindrome ou non, un autre qui permet de calculer l'occurrence d'un caractère dans une chaîne de caractères et le dernier qui permet de découper une ligne de chaînes de caractère en des mots séparés.

1. On souhaite rendre chacune de ces méthodes accessibles à distance. Donnez alors la structure des interfaces qui seront partagées par le serveur et le client sachant que chaque méthode appartient à un objet distinct.

2. Compléter le code de ces méthodes afin qu'elles puissent gérer les erreurs dues à leur appel à distance.

3. Sachant que toute méthode Java appelée par un programme Java distant doit appartenir à un objet accessible à distance. Donnez la structure des classes Java qui vont représenter respectivement les objets créés.

Exercice 3 :

On vous demande maintenant de créer deux programmes client et serveur simples permettant d'établir les services suivants :

1. Le client doit appeler une méthode du serveur permettant d'afficher la date actuelle.

2. Le client doit appeler une méthode du serveur permettant de signaler qu'un nombre est premier ou non.

3. Le client doit appeler une méthode du serveur permettant d'afficher tout les nombres premiers inférieurs à un entier donné en paramètre.

4. Le client doit appeler une méthode du serveur permettant d'indiquer si les deux entiers, donnés en paramètre, sont premiers entre eux ou non.

Exercice 4 : Gestion d'un compte bancaire

Nous disposons d'un service qui offre les opérations suivantes de gestion d'un compte bancaire *debiter(), crediter()* et *lireSolde().*

1. On souhaite rendre chacune de ces méthodes accessibles à distance de manière à ce qu'elles définissent l'interface entre le client et le serveur. Ecrire cette interface.

2. Déduire la classe qui matérialise le service qui offre les opérations *debiter(), crediter()* et *lireSolde().*

3. Ecrire le programme du serveur qui doit permettre l'enregistrement du service auprès de *RMI Registry*.

4. Ecrire le programme du client qui doit être lancé à partir d'un autre répertoire ou d'une autre machine.