TD/TME 8 : Sérialisation et Appels Distants

L'objectif de cette séance de TD/TME est d'étudier les mécanismes inhérents à la sérialisation d'objets dans l'exercice 1 puis de s'intéresser à RMI dans l'exercice 2.La sérialisation des objets est un procédé qui permet de rendre un objet persistant, c'est-à-dire que l'objet est représenté sous une forme grâce à laquelle il pourra être reconstitué à l'identique après sauvegarde et lecture depuis un disque dur ou après transfert à travers un réseau. Dans ce dernier cas, l'objet peut être transmis par copie à une machine (virtuelle dans le cas de Java) "distante" de celle dans laquelle il a été initialement instancié. On parlera alors d'objets distants, un mécanisme repris dans le standard RMI (Remote Method Invocation) qui permet à plusieurs programmes écrits en Java de disposer des méthodes d'objets distants de manière quasi-transparente.

Exercice 1 – Sérialisation en Java

Question 1

Créer une classe `Individu` qui décrit un individu par 3 champs `(Nom, Prénom, Age)` et qui surcharge la méthode `toString()` pour renvoyer un texte type : *"Prénom Nom, Age an(s)"*.

Que faut-il faire pour que cette classe puisse être sérialisée ?

Question 2

Ecrire une classe `CreationIndividu` qui permet l'instanciation d'un objet individu via la saisie au clavier de ses différents champs et qui sauvegarde les informations relatives à cet individu dans le fichier « individu.ser ». (On utilisera pour cela la méthode `writeObject()` de la classe `ObjectOutputStream`).

Question 3

Ecrire une classe LectureIndividu qui instancie un objet Individu à partir des valeurs lues dans le fichier « individu.ser » et qui utilise la méthode toString de ce dernier pour indiquer son contenu. (On utilisera pour cela la méthode `readObject()` de la classe `ObjectInputStream`).

Question 4

Ecrire une application de type client-serveur dont les fonctionnalités sont les suivantes : le serveur est en attente permanente de la transmission d'un objet Individu par le client sur le port 12345 et affiche les données relatives à cet individu lorsqu'un appel d'un client est accepté ; le client a pour simple rôle d'instancier un objet Individu et le transmettre au serveur.

Exercice 2 – RMI un premier exercice

Cet exercice a pour objectif de mettre en oeuvre les principes fondamentaux de RMI (Remote Method Invocation). On s'intéresse ici au développement d'un additionneur distant qui a les fonctionnalités suivantes :

- `eval(a,b)` prend 2 entiers (int ou Integer) en entrée et en retourne la somme ;
- `cumul(val)` permet le cumul des valeurs envoyées une par une ;
- `getCumul()` permet d'obtenir le cumul de toutes les valeurs fournies ;
- enfin `resetCumul()` permet de réinitialiser le cumul à 0.

Question 1 – Interface de l'objet
Ecrire l'interface de l'objet distant `IRemoteCalculette.`

Question 2 – Implantation de l'objet pour le serveur
Ecrire la classe RemoteCalculette implémentant cette interface par un objet contactable à distance.

Question 3 – Création des classes, de la couche (stub) et du squelette (skel)
Compiler RemoteCalculette (et, donc par dépendance, IRemoteCalculette). Exécuter ensuite la commande rmic RemoteCalculette, qui va créer `RemoteCalculette_Stub.class` ainsi que le fichier `RemoteCalculette_Skel.class`.

(TME uniquement) Utiliser `rmic -keepgenerated` pour obtenir les fichiers sources Java intermédiaires.

Question 4 – Serveur de création de l'objet
Ecrire le serveur `ServCalculette` qui a pour rôle de : · créer une instance de `RemoteCalculette`, l'enregistrer sous le nom calculette auprès de la base des objets nommés contactables à distance du registre des objets. Dans un premier temps, on utilisera la base de localhost dont le serveur est supposé à l'écoute sur localhost sur le port `1234`.

Question 5 – Client utilisant l'objet distant
Ecrire une application utilisatrice de l'objet qui : • prend contact avec le serveur de la base d'objets pour obtenir une référence distante ; • réalise quelques calculs, cumule les nombres de 1 à 100 et récupère le résultat avant de faire un reset et en affiche les résultats sur la console client.

Question 6 – Conception d'un fichier de règles de sécurité
Ecrire un fichier nommé `Calculette.policy` assouplissant les règles par défaut en :
- autorisant et acceptant la connection au port 1234 quelle que soit l'origine de la requête,
- autorisant la connection au port 80.

Question 7 – Exécution (TME uniquement)
Toutes les classes sont supposées compilées.
- Créer par rmic les classes couches et squelettes de l'objet distant RemoteCalculette
- Lancer le serveur de la base d'objets distants par rmiregistry 1234 & ;
- Lancer le serveur d'objet en pensant au gestionnaire de sécurité ;
- Lancer leclient en pensant au gestionnaire de sécurité.

Exercice 3 – RPC un petit exemple
Dans cet exercice nous allons mettre en place un service de RPC, qui à la reception d'une chaine de caractères renvoie sa longueur.

Question 1
Ecrire un fichier IDL pour les RPC, décrivant les types et les fonctions pour notre module RPC :
- déclaration d'une fonction prenant une `string` et rendant un `int`.

Question 2
Compléter la fonction `calcul_addition_1_svc` du fichier `long-server.c` et compiler le serveur.

Question 3
Compléter/Modifier le fichier `_client.c` pour que la fonction fasse l'appel RPC et affiche le resultat obtenu.

Question 4
Lancer le serveur sur votre machine, puis le client et verifier que tout se passe bien.