

# TP5

## SOCKETS EN JAVA

### PROGRAMMATION D'APPLICATIONS RESEAU

À rendre le mercredi 25 janvier 2006

#### 1. INTRODUCTION

Le but de ce travail pratique est de comprendre et mettre en pratique la programmation d'applications réseau en Java utilisant des sockets. Pour cela, on implémentera un jeu simple en réseau qui est décrit dans le paragraphe suivant.

Un aspect important d'une vraie application réseau est la sécurité : authentification, confidentialité et intégrité de données. Le langage Java propose des packages pour établir des connexions sécurisées d'une façon simple et transparente, c'est l'utilisation des Secure Socket Layer (SSL). Une question bonus est proposée pour mettre en pratique la programmation d'applications réseau sécurisées.

#### 2. DESCRIPTION DU JEU

On peut définir le jeu avec la phrase suivante : « Le serveur doit choisir un nombre aléatoire et le client doit essayer de le deviner en trois essais ». Voici deux exemples de comment devrait se dérouler le jeu :

Exemple (a)	Exemple (b)
Serveur : devinez à quel numéro entre 1 et 10 je pense Client : 7 Serveur : plus petit Client : 3 Serveur : plus grand Client : 4 Serveur : t'as gagné... bravo !	Serveur : devinez à quel numéro entre 1 et 10 je pense Client : 2 Serveur : plus grand Client : 10 Serveur : plus petit Client : 7 Serveur : t'as perdu... :

Il faut remarque que le serveur, après avoir joué avec un client, il doit être toujours prêt à jouer avec un autre client.

#### 3. IMPLEMENTATION

Implémentez deux classes, Client et Serveur, capables de jouer le jeu décrit précédemment. Ces classes devront établir une connexion réseau utilisant des sockets selon le modèle client/serveur.

*Serveur :*

Le serveur, dans un premier temps, attendra à qu'un client se connecte pour jouer dans le port **9999**. En suite, il génère un numéro aléatoire entre 1 et 10 et demandera au client de le deviner en trois essais. Bien entendu, les serveurs n'enverra que des messages au client du genre : « Devine à quel numéro entre 1 et 10 je pense », « Plus petit », etc. Cependant, pour

simplifier la tâche, le serveur enverra plutôt des codes (numéros négatifs) représentant ces messages, voire la table qui suit :

Valeur	Phrase (String)
-1	Devine à quel numéro entre 1 et 10 je pense
-2	Plus petit
-3	Plus grand
-4	T'as gagné. Bravo !
-5	T'as perdu :(

Par exemple, supposons que le serveur a pensé au numéro 3 (choisi aléatoirement). Si le client envoie un 5 (une tentative), le serveur enverra un -2 (« plus petit ») pour donner au client un renseignement ( $3 < 5$ ). Si le client envoie un 1, le serveur enverra un -3 (« plus grand ») pour donner au client un renseignement ( $3 > 1$ ). Si le client envoie un 3, le serveur enverra un -4 (« t'as gagné. Bravo ! »). En fin, si le client ne réussie pas à donner la réponse correcte dans ses 3 tentatives, le serveur enverra un -5 (« t'as perdu : ( »).

Il est important de remarquer que c'est le serveur qui doit compter le nombre de tentatives pour le client qui vient de se connecter, et non ainsi le client.

*Client :*

Le client essaiera dans un premier temps d'établir une connexion avec le serveur pour jouer en utilisant l'**adresse IP du serveur** (cette adresse est celle de la station où le serveur est exécuté) et le port **9999**. Une fois cette connexion établie, le client recevra le code envoyé par le serveur et affichera le message concerné (le premier message reçu devrait être « Devine à... »). Le client demandera à l'utilisateur d'introduire un numéro pour en suite l'envoyer au serveur, qui à son tour enverra le code le plus approprié (p.ex. si le numéro envoyé par le client est plus grand que le numéro auquel le serveur pense, le code envoyé sera celui du message « plus petit »). Le jeu termine pour le client lorsqu'il reçoit le code -4 (il a deviné) ou le code -5 (il n'a pas deviné dans les trois essais).

Remarquons que si l'on utilise la table précédente, l'interaction client/serveur revient à ce genre de scénario (comparez-le avec les exemples du paragraphe 2) :

Exemple (a)	Exemple (b)
Serveur : -1 Client : 7 Serveur : -2 Client : 3 Serveur : -3 Client : 4 Serveur : -4	Serveur : -1 Client : 2 Serveur : -3 Client : 10 Serveur : -2 Client : 7 Serveur : -5

Il est important de remarquer que le client essaiera de deviner le numéro autant des fois que le serveur lui propose.

#### 4. CONNEXION SECURISE (BONUS POUR 2 POINTS)

Supposons maintenant que le client a envie de jouer qu'avec le serveur qu'il connaît bien, et pas avec n'importe quel serveur. En plus de cela, le client veut que le jeu se déroule de façon confidentielle. Donc, on a typiquement deux services de sécurité à utiliser : l'authentification et la confidentialité.

Implémentez une deuxième version des classes précédentes et nommez-les maintenant ClientSecurise et ServeurSecurise. Pour cela, utilisez les sockets sécurisés (Secure Sockets Layer – SSL) disponibles en Java, et l'outil keytool ou keystore pour la génération,

importation et exportation des clés privés/publiques. Regardez les références citées dans le séminaire pour vous aider.

## **5. RAPPORT ET CODE**

Un rapport sur papier devra être rendu contenant une description détaillée de la façon dont vous avez pris en main le problème et l'implémentation (diagrammes, schémas, pseudo code, algorithmes, etc.). Il contiendra aussi des exemples d'exécution (un positif et un négatif) et le code source imprimée.

Le code source devra être commenté de façon judicieuse et il devra être envoyé (tous les fichiers \*.java) à l'adresse email *alfredo.villalba@cui.unige.ch*. Assurez-vous que le code que vous envoyez puisse être compilé correctement.

## **6. TIPS**

Vous pouvez utiliser ce bout de code pour générer des entiers aléatoires entre 1 et 10 :

```
Random intGenerator = new Random();  
...  
...  
int number = 1 + intGenerator.nextInt(10);  
...  
...
```