M1 Programmation avancée et répartie en Java : TP 7

Jean Fortin et Frédéric Gava
gava@u-pec.fr ou jean.fortin@gmail.com

1 Exercice 1

1. Créer une classe `Client`, ayant comme champs privé un hôte (de type String) et un port (de type int). Créer le constructeur correspondant.
2. Dans cette classe, ajouter une méthode `connecte`, qui à l'aide d'une `Socket` se connecte au serveur correspondant à l'hôte et au port, puis ferme la socket. Afficher dans la console un message lorsque la connection est effectuée, et lorsqu'elle est fermée.
3. Ajouter une méthode main pour tester votre classe, en vous connectant à l'hote "173.194.40.223" (google.fr) sur le port 80.
4. Dans la méthode `connecte` puis la lecture d'un message de type String depuis la socket, et l'afficher dans la console. Testez (normalement le String doit être vide).

2 Exercice 2

1. Créer une classe `ClientGUI`, représentant une fenêtre comprenant un `JLabel` au centre, et un `JTextField` et deux `JButton` (Envoyer et Connecter) au sud. (On pourra s'inspirer des interfaces réalisées dans les TPs précédants). Ajouter une méthode main permettant de tester votre fenêtre.
2. Ajouter un gestionnaire d'évenement pour le bouton Connecter, qui se connecte à un hôte de la même façon que pour l'exercice 1. (On rajoutera des champs privés pour l'hôte et le port, et on adaptera le constructeur en conséquence). Cette méthode ne doit pas fermer la connexion réseau. On enregistrera les flots d'entrée/sortie liées à la socket dans des variables privées de la classe.
3. Ajouter un gestionnaire d'évenement pour le bouton "Envoyer". Ce bouton doit envoyer le texte entré par l'utilisateur via la `Socket`, lire un message en réponse, et l'afficher dans le `JLabel`.
4. Testez votre classe sur l'hote "173.194.40.223" (google.fr) sur le port 80 en envoyant le message "`GET /\n`". Essayez des "PUT" et des "GET" du protocole HTTP en analysant les réponses du serveur.

3 Exercice 3

Le but de cet exercice est de réaliser un programme de messagerie instantannée. Afin de se concentrer sur l'aspect programmation réseau, une base de travail est fournie sur la page web des TPs, comprenant l'intégralité de l'interface graphique, et deux classes à compléter, Client et Serveur.

1. Compléter la classe Client.
   On créera les flots d'entrée sortie sur le modèle

```
out = new PrintWriter(sck.getOutputStream(), true);
out.println(message);

in = new BufferedReader(new InputStreamReader(sck.getInputStream()));
System.out.println(in.readLine());
```

2. Compléter les classes pour le Serveur

4 Exercice 4

On reprend l'exercice sur la calculatrice.

1. Créer une classe `ServerCalc` qui accepte les connexions de type `Socket` sur un port de votre choix. Pour chaque connexion, un thread est créé pour traiter le contenue de la connexion. Les messages seront du type `nomDeFonction` (comme sin, cos, etc.) puis un entier. La réponse sera bien sûr l'évaluation de la fonction sur l'entier.
2. Créer une classe `ClientCalc` qui se connecte sur un `ServerCalc`. L'interface est comme celle de l'exercice 2 mais on supppose deux `JTextField`, l'un pour la fonction l'autre pour l'entier ; La réponse du `ServerCalc` sera affichée sur le `JLabel`.
3. Testez votre code et si possible avec deux machines de la salle de TP.
4. Essayer maintenant que le client envoie un objet (qui sera serialisé) contenant un entier (dans un champ "valeur") ainsi qu'une méthode "int toEval(int arg)". Le client choisira la fonction à faire évaluer par le serveur (dans une liste Swing par exemple). Construire l'objet dynamiquement. Le serveur, lui, doit de-sérialiser le string reçu puis l'analyser pour trouver la valeur (si elle existe) ainsi que vérifier l'existence et le type de la méthode "toEval" avant de rééllement évaluer la demande du client.

5 Exercice 5

Réaliser un jeu de Tic-Tac-Toe (morpion) en réseau. Le programme proposera à l'utilisateur de démarrer en mode client ou serveur.