3 EIB - TP Java (sockets, threads)

S. Reynal

17 septembre 2013

L'objectif du TP est de développer une application en langage java permettant :

— de simuler la réception par un serveur (la machine sur laquelle vous développez) de données (température, etc.) émises par plusieurs capteurs intelligents connectés au serveur via le réseau (ethernet ou WiFi),

— d'afficher ces données sur une interface graphique (UI) simple.

Guidelines...

Q1. Commencer par implémenter la classe "Sensor" qui hérite de la classe "Thread". Cette classe se contente pour l'instant, dés que le thread est démarré, d'afficher régulièrement, chaque seconde, une donnée de mesure factice (un nombre aléatoire) à l'écran (via la console). Le code correspondant se trouvera dans la méthode "run()", et repose sur une boucle infinie et l'appel de la méthode "sleep()" (attention à bien gérer les exceptions associées). Les threads seront lancés depuis la méthode "main()". L'affichage souhaité pour chaque Sensor est à titre indicatif du type : *"Capteur numéro 4 : température = 28C"*.

Q2. Implémenter maintenant la classe Serveur. Cette classe hérite de ServerSocket et est chargé d'attendre les connections émanant de clients (les capteurs). Elle repose sur l'appel de la méthode "accept()" de la classe ServerSocket, qui renvoie un Socket représentant la connection active. Une fois la connection activée, récupérer le InputStream associé à la connection via la méthode ad hoc dans Socket, lire une donnée (un double) depuis la connection, fermer la connection, et passer au client suivant. Pour l'instant, le serveur ne peut gérer qu'une connection active avec un client à la fois. On devra utiliser la classe DataInputStream (paquet java.io).

Q3. Le code n'est véritablement fonctionnel que s'il existe des clients ! Ajouter à la classe Sensor dans la méthode run(), la connection au serveur (via la classe Socket). Une fois la connection acceptée

par le serveur, récupérer le OutputStream associé (via la méthode getOutputStream de la classe Socket), et écrire un double correspondant à la température.

Q4. Ajouter à la classe Serveur la possibilité de gérer plusieurs connection simultanément, via le multithreading. Pour cela, il faut insérer le code qui attend un client (la méthode accept()), à l'intérieur de la méthode run() d'un thread (ou implémenter la méthode run() de l'interface Runnable, autre approche), puis lancer un nouveau thread à chaque nouveau client. Ensuite, il suffit de laisser les threads tourner en boucle infinie et communiquer avec leur capteur attaché.

Q5. Pour les plus courageux, réaliser une interface graphique simple, via Swing, permettant d'afficher les valeurs de températures envoyées par les différents capteurs (utiliser des JTextField).

Remarque : vous pouvez travailler en équipe, un binôme développant la classe Sensor, un autre la classe Serveur. Ainsi les "faux" capteurs peuvent être disséminés sur plusieurs machines de la salle 070 et émettre leur données pour le reste du groupe !