TP 1 Sockets UDP et TCP

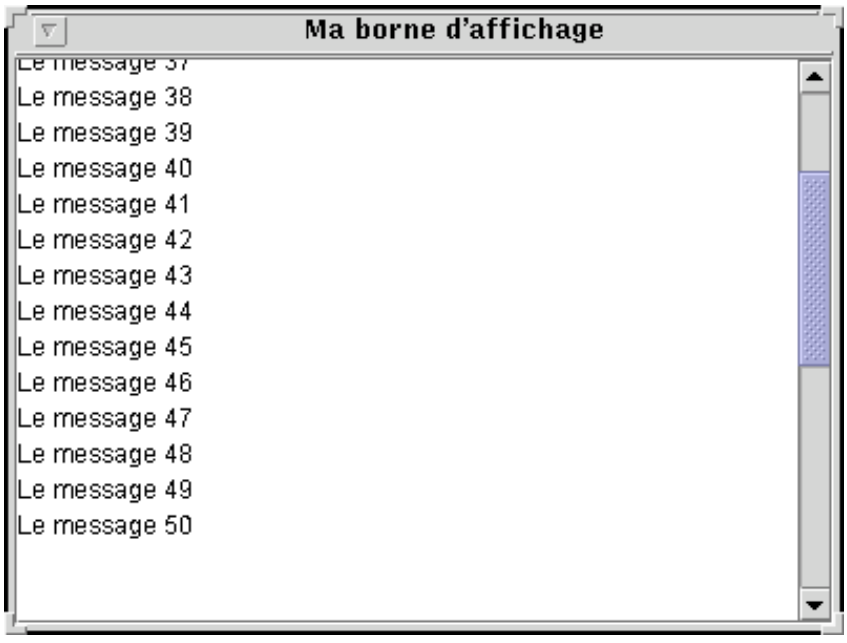
L'objectif du TP est de mettre en œuvre les sockets UDP et TCP/IP du langage Java sur une simple application répartie : un panneau d'affichage électronique accessible et contrôlable depuis internet.

1. Le panneau d'affichage

On dispose d'une classe Java implantant un panneau d'affichage électronique. Cette classe peut être téléchargée depuis moodle.

Voici un exemple d'utilisation de cette classe et le résultat visuel obtenu :

```
public class UtilisationPanneau {
  public static void main(String[] args) {
    // Creation de l'IHM du panneau d'affichage.
    IHM ihm = new IHM("Ma borne d'affichage");
    // Mettre un listener sur les evenements de la fenetre.
    IHM.mettreListenerSortieProgramme(ihm);
    // Afficher la fenetre.
    ihm.setVisible(true);
    // Ajouter des lignes de texte dans le panneau d'affichage.
    for (int i=0; i<30; i++)
    ihm.ajouterLigne("Le message " + i);
    // Effacer le contenu du panneau d'affichage.
    ihm.raz();
    // Ajouter de nouvelles lignes de texte.
    for (int i=30; i<50; i++)
    ihm.ajouterLigne("Le message " + i);
  }
}
```



On désire rendre accessible à distance, c'est-à-dire piloter et contrôler, ce panneau d'affichage à travers le réseau internet. Pour cela, vous allez développer différents serveurs et clients en utilisant les sockets UDP, et TCP/IP du langage Java. Attention, vous ne devez en aucun cas modifier la classe `IHM.java`, vous devez uniquement la réutiliser !

2. Le panneau accessible en UDP

Cette première partie a pour objectif de rendre accessible le panneau d'affichage à travers un serveur et un client dialoguant via des sockets UDP. Inspirez vous de l'exemple donné en cours pour implanter le serveur et client suivants.

2.1 Le serveur UDP

Le serveur à réaliser doit :

- Instancier la classe IHM du panneau d'affichage électronique.
- Afficher ce panneau d'affichage.
- Créer une socket UDP sur le port 8000 (par exemple).
- Dans une boucle infinie :
    - Créer un datagramme UDP.
    - Attendre la réception de ce datagramme via la socket créée.
    - Transformer le datagramme reçu en une chaîne de caractères qui contiendra l'ordre demandé par le client distant.
    - Traiter l'ordre reçu en fonction de son contenu :
    - Si l'ordre commence par la chaîne « afficher » alors ajouter au panneau d'affichage la fin de la chaîne contenant l'ordre.
    - Si l'ordre commence par la chaîne « effacer » alors effacer le panneau d'affichage.
    - Si l'ordre contient autre chose alors c'est une erreur.
    - Construire une chaîne contenant la réponse du serveur : « OK : Ordre exécuté » ou « ERREUR : Ordre inconnu » en fonction du traitement précédent de l'ordre.
    - Créer un datagramme contenant la réponse au client.
    - Envoyer le datagramme à la socket UDP du client.

Implanter ce serveur dans la classe `ServeurUDP.java`.

2.2 Le client UDP

Le client à réaliser doit :

- Créer une socket UDP anonyme.
- Créer un datagramme UDP contenant l'adresse et le port de la socket UDP du serveur et l'ordre à envoyer au panneau :
- Soit la chaîne « afficher le message à afficher »
- Soit la chaîne « effacer »
- Soit une chaîne contenant un mauvais ordre.
- Envoyer le datagramme créé via la socket UDP.

- Créer un datagramme pour recevoir la réponse du serveur.
- Attendre sur la socket UDP la réception du datagramme envoyé par le serveur.
- Afficher la chaîne contenue dans le datagramme envoyé par le serveur.
- Fermer la socket UDP.

Implanter ce client dans la classe `ClientUDP.java`.

2.3 Exécution

Lancer le serveur UDP sur votre machine et y accéder depuis le client UDP lancé aussi sur votre machine. Lancer le serveur UDP sur différentes machines et accéder à ces serveurs depuis le client UDP lancé sur votre machine.

3. Le panneau accessible en TCP/IP

Cette deuxième partie a pour objectif de rendre accessible le panneau d'affichage à travers un serveur et un client dialoguant via des sockets TCP/IP.

3.1 Exemple d'utilisation des sockets TCP/IP

Inspirez vous de l'exemple donné en cours pour implanter le serveur et client suivants.

3.2 Le serveur TCP/IP

Le serveur à réaliser doit :

- Instancier l'interface graphique du panneau d'affichage.
- Afficher cette interface graphique.
- Créer une socket TCP serveur sur un certain port (e.g. 8000).
- Boucler éternellement sur :
  - Attendre une connexion cliente.
  - Créer un flux pour lire les données émises par le client.
  - Lire une chaîne contenant l'ordre émis par le client.
  - Traiter la chaîne ordre comme dans le serveur précédent.
  - Créer un flux de sortie pour envoyer la réponse du serveur.
  - Envoyer la réponse du serveur.
  - Fermer la socket vers le client.

Implanter ce serveur dans la classe `ServeurTCP.java`.

3.3 Le client TCP/IP

Le client à réaliser doit :

- Créer une socket TCP pour se connecter au serveur.
- Créer un flux de sortie pour envoyer l'ordre au serveur.
- Envoyer l'ordre au serveur.
- Créer un flux pour lire la réponse émise par le serveur.

- Lire la réponse du serveur.
- Afficher cette réponse.
- Fermer la socket TCP.

Implanter ce client dans la classe `ClientTCP.java`.

3.4 Exécution

Lancer le serveur TCP sur votre machine et y accéder depuis le client TCP lancé aussi sur votre machine. Lancer le serveur TCP sur différentes machines et accéder à ces serveurs depuis le client TCP lancé sur votre machine.

3.5 Accès par `telnet`

Accéder au serveur TCP par le programme Unix `telnet`.

```
unix> telnet NomMachine NumeroPort
```

Vous allez constater que vous ne pourrez saisir qu'un seul ordre pour chaque lancement du programme `telnet`. Modifier votre serveur pour qu'il accepte une succession d'ordres plutôt qu'un seul par connexion cliente. Pour cela, il faut mettre dans le serveur une boucle autour de la réception et du traitement des ordres envoyés par un client. Un seul client TCP peut se connecter à la fois au serveur TCP. Modifier le serveur TCP pour qu'il accepte plusieurs clients simultanément. Pour cela, il est nécessaire de créer un thread de gestion de la socket de communication avec chaque client.