

## TP 01 : Rapport The Java Chat

Pour ce programme j'ai utilisé Les classes suivantes:

### **Serveur**

**ServeurConnexion** (hérite de Thread)

### **Client**

**ClientEcouleur** (hérite de JFrame et implémente Runnable, NetListener, ActionListener, KeyListener)

La classe **Serveur** lance une boucle infinie dans son main dans laquelle elle appelle la Classe **ServeurConnexion** à chaque fois qu'elle reçoit une connexion sur son port et reste ainsi prête pour une nouvelle connexion. On travaille avec des **ObjectInputStream** pour standardiser les transmissions.

La classe **ServeurConnexion** est initialisée par **Serveur** par son constructeur avec pour paramètres le Socket d'un client et le vecteur connexions contenant la liste des sockets des clients connectés. Une fois lancé ce thread reçoit des messages qu'il traite pour extraire différentes informations et selon celles-ci il les envoie aux autres **ServeurConnexion** connectés (en théorie seulement car en pratique je n'ai réussi qu'à envoyer directement et non à extraire les informations, ceci par manque de temps).

La classe **Client** comme la classe serveur ne sert qu'à initialiser une autre classe qui cette fois si n'est pas un thread mais qui implémente l'interface **Runnable** ce qui revient en fait au même. On peut lancer la classe **ClientEcouleur** en passant l'adresse (127.0.0.1) en paramètres ou sans.

La classe **ClientEcouleur** est donc appelée par **Client** et si l'adresse est déjà rentrée cela démarrera directement le constructeur appelant la méthode **run**, entraînant le lancement du thread. Si par contre on lance **ClientEcouleur** via l'autre constructeur la méthode **login** sera appelée et demandera à l'utilisateur l'adresse du serveur (127.0.0.1 ou autre si sur machine distante) et son nom d'utilisateur (le bouton ok n'a pas l'air de fonctionner mais lorsqu'on appuie sur enter ça marche !!!). Une fois les informations correctement rentrées la méthode appelle le premier constructeur, la seule différence est que l'on a un nom d'utilisateur personnalisé.

On utilise des **ObjectInputStream** et **ObjectOutputStream** pour pouvoir communiquer entre **JFrame** mais on les traite comme des **String** dès lors qu'ils arrivent.

Cette classe implémente **ActionListener** et **KeyListener** pour pouvoir gérer les actions sur les **JFrame** et sur le clavier.

Elle implémente également **NetListener** afin de savoir quels autres clients se connectent sur le réseau et pour fermer le programme au cas où le serveur s'éteint.

Les méthodes de chaque classe sont décrites brièvement dans les commentaires du code.

Je n'ai pas pu finir de développer toutes les options par faute de temps, j'ai laissé le début de quelques idées que je n'ai pas réussi à faire fonctionner en commentaire.