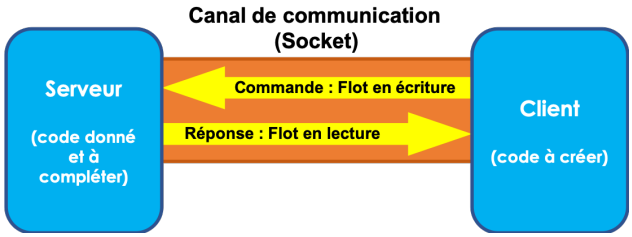
Mini Client/Serveur FTP

Dans le cadre de ce Mini Projet, à faire durant les séances des TPs, on envisage de développer un mini Client/Serveur FTP. Le projet comporte deux applications complètement indépendantes **(2 projets Java : un projet pour le client et un projet pour le serveur)**. L'application serveur contient le code des différentes commandes FTP et l'application client doit pouvoir envoyer des commandes au serveur. Le schéma de communication Client/Serveur des deux applications est donné comme suit :



Le détail concernant chacune de ces applications est donné comme suit :

Le serveur doit permettre à un client de se connecter. Une fois le client connecté, le serveur doit répondre par le texte suivant (3 lignes) :
1 Bienvenue !
1 Serveur FTP Personnel.
0 Authentification :

Le client peut envoyer des commandes au serveur. Pour chaque commande envoyée, le serveur doit répondre par un message sous la forme : **x texte**
où **x** désigne l'état de la réponse. Il peut avoir 3 valeurs possibles :
0 → Signifie que la réponse est positive et finie, il n'y a rien à lire après
1 → Signifie que la réponse est intermédiaire et pas finie. Il faut continuer à lire car il reste encore des lignes à lire.
2 → Signifie que la réponse est négative et finie, il n'y a rien à lire après

Une fois connecté au serveur, le client doit envoyer les deux commandes suivantes pour se connecter sur son compte. Aucune commande ne doit être acceptée si l'utilisateur n'est pas connecté. A cette étape, il n'existe aucun compte physique réel, qui représente un dossier sur le disque du serveur.
user personne
pass abcd

Pour quitter le serveur, il faut lancer la commande **bye**
Les commandes à implémenter sont :

1. CD : pour changer de répertoire courant du côté du serveur
2. GET : pour télécharger un fichier du serveur vers le client
3. LS : afficher la liste des dossiers et des fichiers du répertoire courant du serveur
4. PASS : pour envoyer le mot de passe
5. PWD : pour afficher le chemin absolu du dossier courant

6. STOR : pour envoyer un fichier vers le dossier courant serveur
7. USER : pour envoyer le nom du login
8. MKDIR : pour créer un nouveau dossier dans le pwd
9. RMDIR : pour supprimer un dossier s'il est vide

Le code donné contient une première version du serveur et du client. La commande PWD est déjà implémentée. Uniquement un seul client est accepté à la fois. Un seul client est considéré. Son user est **breton** et son mot de passe est **brestois**

Ce qui est demandé à faire :

Étape 1 : Application de base

1. Utilisez Eclipse et créer un nouveau projet Java au nom de **Serveur**
2. Téléchargez les sources Java du serveur FTP via l'url suivante :

   http://labsticc.univ-brest.fr/~bounceur/cours/java_reseau/tps/serveur_ftp.zip

3. Lisez-le et analysez-le. Quelles sont les commandes ftp implémentées ?
4. Créez un autre projet Java au nom de **Client**. Celui-ci doit se connecter au serveur et lui envoyer des commandes ftp (à ce stade on peut utiliser les commandes **user**, **pass** et **pwd**)
5. Complétez toutes les commandes FTP non implémentées du serveur
6. Complétez le client pour pouvoir envoyer toutes les commandes implémentées

Étape 2 : Gestion des exceptions

On suppose qu'un seul client à la fois peut se connecter sur le serveur. Dans cette étape, on envisage gérer les exceptions java liées au départ subits (sans commande) du client et du serveur (cas d'arrêt de l'exécution des applications). Il faut ajouter la possibilité que le client se déconnecte proprement lorsque le serveur s'arrête subitement. Il doit afficher un message comme quoi le serveur est déconnecté. Et de même pour le serveur, il faut qu'il puisse continuer à attendre d'autres clients lorsqu'un client part.

Étape 3 : Perfection

Dans cette étape, le serveur doit pouvoir recevoir plusieurs clients. Il faut ajouter la possibilité de connexion de plusieurs clients à la fois. Pour ce faire, il faut suivre la démarche suivante :

1. Derrière chaque client se cache un dossier au niveau du serveur portant le nom de son user. A l'intérieur de ce dossier on met un fichier pw.txt contenant le mot de passe. La connexion d'un client passe par la commande **user nom_user** qui permet au serveur de vérifier l'existence d'un dossier au nom de **nom_user** et la commande **passe xyz** qui permet au serveur de vérifier que le texte du fichier pw.txt est le même que **xyz**. A noter que chaque commande envoyée par le client au serveur engendre une action au niveau du serveur et une réponse au client.
2. Il faut commencer par une version gérant un seul client à la fois
3. Ensuite par une version qui gère plusieurs clients à la fois.

Cas des commandes STOR et GET :

Pour programmer ces deux commandes, il faut s'inspirer du protocole FTP réel. C'est-à-dire, il faut prévoir l'utilisation de deux canaux comme suit :

1. **STOR nom_fichier**

   La commande STOR permet au serveur de créer un fichier vide au niveau du working directory (dossier courant) au nom de nom_fichier donné en argument et il doit répondre par un message informant qu'il est prêt à recevoir les données du fichier sur un autre canal (Socket) ayant le port 4000 par exemple. Ensuite, il faut ouvrir un deuxième canal (Socket) sur un le port 4000 du côté du client. A partir de ce deuxième canal il faut créer un flux en écriture. C'est ce dernier qui sera utilisé pour envoyer les données du fichier. Une fois le fichier créé et copié, fermer le flux ainsi que le canal correspondant. La communication continue toujours sur le premier canal.

2. **GET nom_fichier**

   La commande GET fonctionne exactement de la même manière que la commande STOR, faut que c'est le serveur qui envoie le fichier au client.

Étape 4 : L'IHM

Ajouter une interface graphique. Utilisez JavaFX ou Swing selon votre préférence.