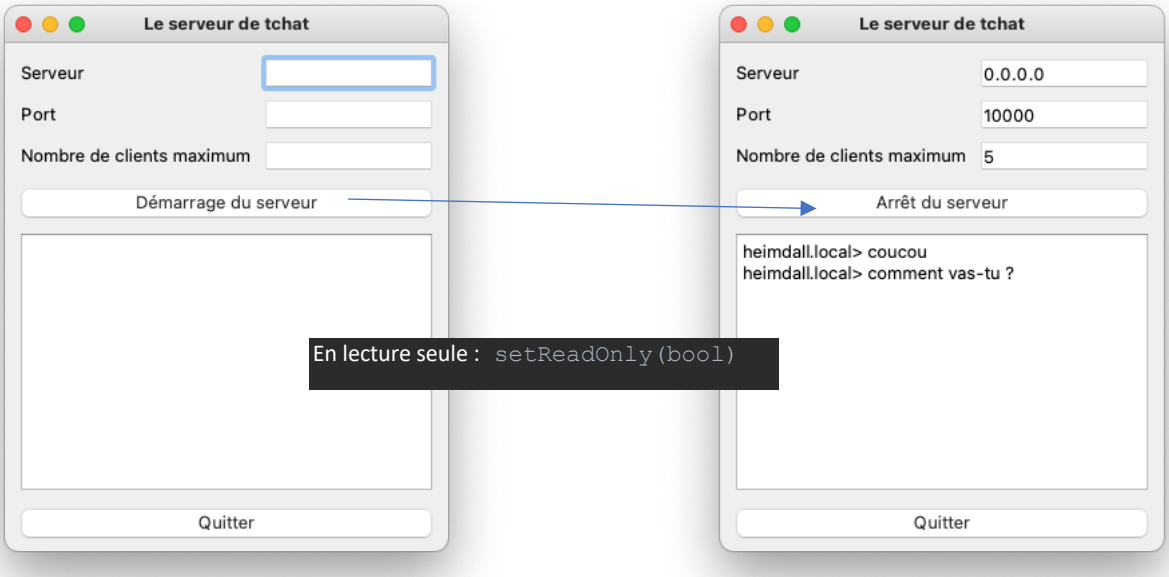
Un serveur pour des clients

Vous avez droit à ***vos*** codes, au cours et aux exercices vus dans le cadre de R3.09. Tout autre document et toute recherche sur internet est interdit.
Votre téléphone portable devra être rangé dans votre sac et celui-ci mis sur le devant de la salle.
Le rendu se fera sur Moodle dans ce devoir correspondant. Vous devrez également mettre dans votre code un lien sur le github où vous avez mis le code l'examen.

Sujet

L'idée est de réaliser une interface pour un serveur sur le modèle ci-dessous :



Partie 1 : réaliser l'interface graphique ci-dessus

- Vous remarquerez que la fenêtre d'affichage des clients est en lecture seule (*readonly*).
- Le texte du bouton connexion change de texte si le serveur est démarré.

Partie 2 : les sockets – bouton Démarrage du serveur, fonction __demarrage

- Démarrage du serveur sur les informations indiquées dans les fenêtres, machine qui supporte le serveur (par défaut 0.0.0.0), le port (10000 par défaut) et le nombre maximum de clients (5 par défaut). Attention, vous ne devrez pas mettre de *accept* dans cette fonction (voir partie 3).
- Mettez les valeurs par défaut dans les textes pour éviter une saisie systématique.
- Changement de nom du bouton de « Démarrage du serveur » → « Arrêt du serveur ».
- A noter que vous pouvez déconnecter le serveur avec le même bouton et de fait désactiver le bouton d'envoi et la fenêtre texte associé.

Partie 3 : connexion synchrone d'un client, fonction __accept

- Ajouter une fonction de connexion d'un client qui permettra d'accepter un client qui se connecte sur le serveur.
- Pensez à mettre en attribut (privé), l'ensemble des connexions de socket (client, serveur et socket).
- Appelez cette fonction dans le démarrage de la socket (fonction __demarrage).
- Constatez que votre interface est bloquée jusqu'à ce qu'un client se connecte.
- Prenez le client proposé dans Moodle pour tester une première connexion.

Partie 4 : Une première thread pour l'acceptation

- Vous allez réaliser une thread permettant de désynchroniser la fonction __accept de l'interface
- A place de l'appel de la fonction __accept, démarrer une thread.
- Vous devriez constater que l'interface n'est plus bloquée.
- Vous pouvez même connecter un premier client (sachant que pour le moment, vous ne gérez pas la réception de messages).

Partie 5 : Réception

- Dans un premier temps, vous allez ajouter une fonction de réception avec une boucle sur la réception de message jusqu'à recevoir le message « deco-server ».
- Appelez cette fonction dans la fonction __accept.
- Vous avez maintenant un système (mono-)client-serveur asynchrone avec une interface graphique pour le serveur.

Partie 6 : La déconnexion (Arrêt du serveur ou Quitter)

- Pour la partie déconnexion, il faut fermer le client, il faut fermer le serveur au niveau des sockets.
- Arrêter la *thread* concernant la réception.
- Pour le bouton déconnexion revenir à l'intitulé « Démarrer le serveur »
- Vous constaterez que la fermeture n'est pas parfaite si c'est le serveur qui s'arrête mais c'est dû au programme du client que je vous ai fourni.

Partie 7 : ah les exceptions !

Il faudrait montrer que vous savez manipuler les exceptions, deux choix possibles :

- Facile : vous allez devoir faire une conversion pour le port. Gérer le fait que vous pourriez avoir un texte au niveau du port et au niveau du nombre de clients.
- Normal : gérer le fait que les exceptions sur la partie client/serveur par exemple lors de la connexion si une erreur se produit sur l'acceptation de la socket, la réception de message ou la déconnexion.

Partie 8 : comment accepter plusieurs clients ? – s'il vous reste du temps (bonus)

Je ne vous demande pas ici de procéder à des modifications de code. Il s'agit de réfléchir et de me donner des éléments réponses dans le code sous forme de commentaire sur les deux questions qui suivent.
Question 1 : comme je le disais dans la partie 6, la déconnexion côté client n'est pas parfaite. A votre avis que faut-il faire côté client pour un arrêt correct ?
Question 2 : pour le moment, nous n'avons qu'un seul client qui peut se connecter à la fois. Que faudrait-il faire dans le programme pour pouvoir accueillir plusieurs clients ?