IFT1020 – TRAVAIL PRATIQUE #3 – 17 juin 2005

Un chat entre nous!

Mohamed Lokbani

Équipes: Le travail est à faire en monôme ou en binôme, mais pas plus de deux. Vous ne remettez qu'un seul travail par équipe. Si vous décidez de travailler en binôme, vous devez quand même changer de partenaire par rapport au tp#2.
Remise : Deux remises à effectuer : électronique et papier le **Jeudi 07 juillet, 20h30 au plus tard, sans possibilité de retard**.
Conseil: N'attendez pas le dernier jour avant la remise pour commencer, vous n'aurez pas le temps.

But : Ce TP a pour but de vous faire pratiquer le design d'une application graphique.

Enoncé : On se propose d'écrire un programme qui permet à différents clients de bavarder entre eux. Ce bavardage se fait via l'envoi de messages courts. Ces messages transitent par un serveur. Le serveur aura la double fonction de recevoir les messages envoyés par les clients expéditeurs pour les acheminer par la suite vers les clients destinataires. On peut comparer la fonction de ce serveur au travail effectué par une poste. Une personne A envoie une lettre à une personne B. Cette lettre est d'abord récupérée par la poste, pour la transmettre par la suite à la personne destinataire B.

Interface Serveur La figure **-1-** est une représentation parmi d'autres de l'interface graphique associée au serveur. Cette dernière doit contenir au moins les fonctionnalités suivantes :

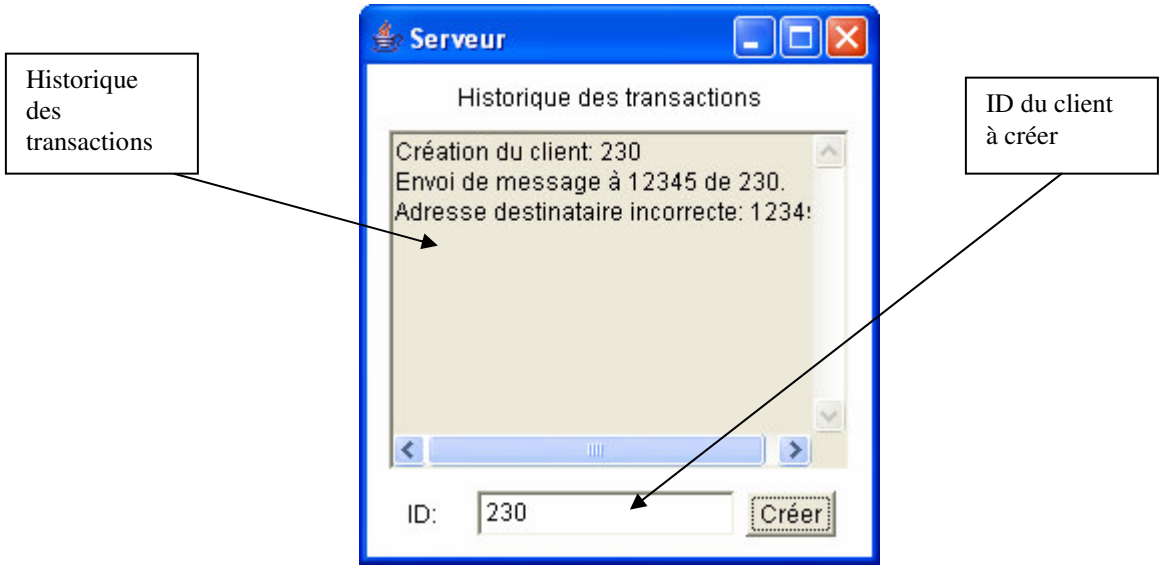

Figure -1- GUI Serveur

ID : Une identité **unique** est associée à chaque client. Cette identité peut-être sous la forme d'un nom, ou un numéro etc.

Registre : Un historique des activités du serveur. Le serveur préserve une série d'informations dans un registre, parmi lesquelles :

-1- Lors de la création d'un client : Vous avez choisi une identité pour le client, exemple **230**, vous avez appuyé par la suite sur le bouton « Créer ». Un nouveau client verra le jour et le serveur va archiver dans son registre « **Création du client : 230** ».
-2- Un client décide d'envoyer un message à un client B. Le serveur va archiver « **Envoi de message à B de A** ».
-3- Une série de message d'erreur, tout dépend la situation dans laquelle on se trouve.

-a- Vous avez appuyé sur le bouton « Créer » en ayant omis de compléter le champ « ID ». Le serveur va archiver « **ID Invalide** ».
-b- Un client A envoie un message à un client B, sachant que l'identité de A existe encore, alors que celle de B ne l'est pas (« ID » non créée ou bien « ID » a été détruite entre temps) « **Adresse destinataire incorrecte** ».
-c- Un client A envoie un message à un client B, sachant que l'identité de A, a été détruite ou altérée entre temps « **Adresse expéditeur incorrecte** ».

Les actions associées au Serveur

Créer : Un bouton qui valide la création du client dont l'identité est ID.

Interface Client La figure **-2-** est une représentation parmi d'autres de l'interface graphique associée au client. Cette dernière doit contenir au moins les fonctionnalités suivantes :

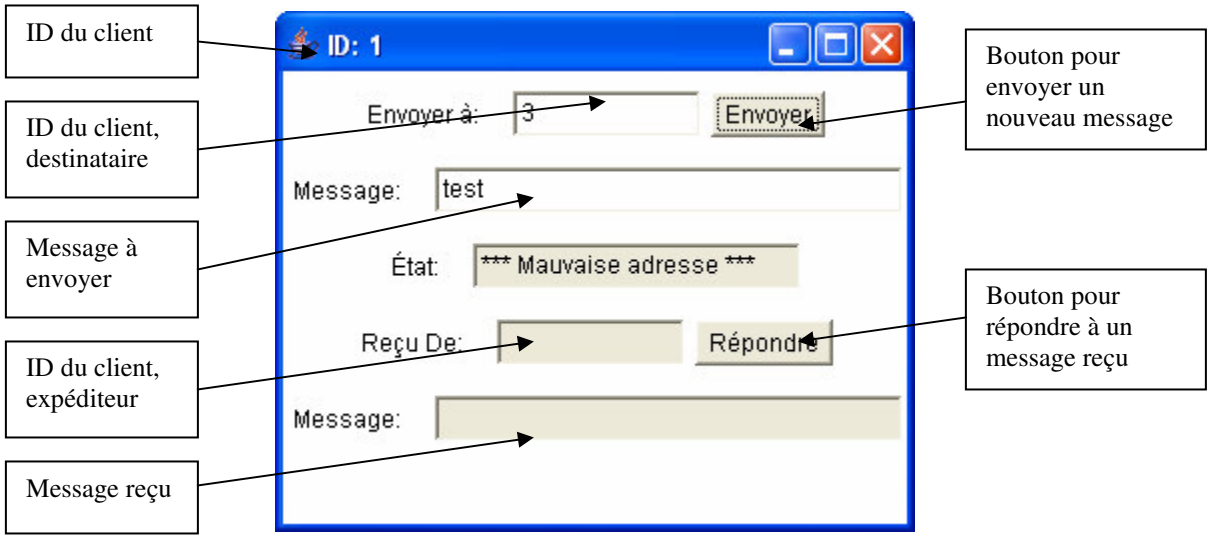

Figure -2- GUI Client

On ne permet aux clients d'interagir directement avec les champs « État », « Reçu De : » et le « Message » reçu. Le client propriétaire de l'interface ne pourra donc pas modifier ces champs.

« **Envoyer à** : » : Ce champ contient l'adresse du destinataire.

« **Message** : » : Message à envoyer au destinataire.

« **État** »: permet d'informer l'expéditeur ou le destinataire sur l'état de l'envoi ou de la réception. Il y a 3 états possibles:

-a- Un client A envoie un message à un client B, les deux identités existent ; le message « **Message envoyé** » sera affiché dans le champ « État » du client A, alors que le message « **Message reçu** » sera affiché dans le champ « État » du client B.

-b- Un client A envoie un message à un client B, sachant que l'identité de A existe encore, alors que celle de B ne l'est pas (Le client B n'a pas été créé encore, ou bien il a cessé d'exister) ; le message « ***** Mauvaise adresse ***** » sera affiché dans le champ « État » de A.
-c- Le champ « Envoyer à : » du client A est vide et on clique quand même sur le bouton « Envoyer » du client A. Le message « **ID Destinataire est invalide** » sera affiché dans le champ « État » de A.
-d- Le champ « Reçu de : » du client A est vide et on clique quand même sur le bouton « Répondre » du client A. Le message « **ID Expéditeur est invalide** » sera affiché dans le champ « État » de A.

« **Reçu De** : » : Ce champ contient l'adresse de l'expéditeur.

« **Message** : » : Message reçu, envoyé par l'expéditeur. L'expéditeur peut-être le même que le destinataire du message. Un client peut s'envoyer à lui-même un message.

Les actions associées au Client

Envoyer : Un bouton qui valide l'envoi du message.

Répondre : Un bouton qui permet de répondre à l'expéditeur du dernier message reçu. Un clique sur ce bouton, fait que l'adresse de l'expéditeur sera affichée dans le champ « **Envoyé à :** ». Le client n'aura donc qu'à compléter le champ « **Message** » à envoyer.

Scénario Clients/Serveur On crée d'abord un serveur, représenté par une interface graphique. À partir de l'interface graphique du serveur, on crée une série de clients avec une identité unique. C'est le serveur qui aura la tâche de créer les clients. Il pourra donc garder un lien avec chacun des clients qu'il a créé. Chaque client aura sa propre interface graphique. Ainsi donc, les clients pourront communiquer entre eux (au maximum une communication à deux), à travers l'envoi via leur interface graphique respective une série de messages. Le message envoyé va transiter d'abord par le serveur, ce dernier après avoir effectué toutes les vérifications nécessaires, il se chargera de transmettre le message vers le destinataire.

Contraintes :
-1- On ne vous demande pas de coder une application client serveur via le réseau! On vous demande tout juste de coder des interactions entre différents clients se trouvant sur la même machine. Pour ce faire, vous utilisez un serveur qui se charge de jouer le rôle de « pivot », comme la poste! Donc, oublier l'idée d'utiliser des connexions TCP/IP ou des méthodes de la classe Socket pour faire ce travail! Si vous le faite, vous serez de facto hors sujet.
-2- La méthode « **main** » doit figurer dans le fichier « **Tp3.java** »
-3- Vous devez utiliser pour ce travail pratique « **javadoc** ».
-4- Si nécessaire (ou si vous jugez que c'est utile), joindre à votre remise un « **manifest** » et un fichier « **jar** ».
-5- Vous pouvez créer des interfaces graphiques à votre goût. L'essentiel est que les méthodes de base décrites dans l'énoncé, ainsi que les différentes interactions et les différents messages à afficher doivent être présents. Tous les chemins mènent à Rome! Mais quand même, il préférable d'utiliser le plus simple et le plus court! N'embellissez votre interface qu'après avoir réalisé le travail demandé.
-6- Il n'y a pas de bouton pour détruire un client. Libre à vous si vous voulez en ajouter un. Si vous le faite, il sera testé et noté☺.

Remise : Il est important de noter que votre TP sera exécuté par les démonstrateurs sur une machine Linux avec le compilateur jdk (1.4.2). Si par choix vous décidez d'utiliser un autre compilateur, vérifiez que le code que vous avez produit (qui normalement fonctionne correctement chez vous) fonctionne bien sur les ordinateurs du DIRO. Avant de faire cela, assurez vous d'abord que vous avez la bonne version de jdk, en utilisant pour cela la commande : "**java –version**" devra donnée le numéro de version "**1.4.2**".

La remise comprend **3** (**TROIS**) choses (avant le jeudi 07 juillet à 20h30) à Étienne :

1. Envoyez vos fichiers par la procédure de remise électronique habituelle (Pour obtenir de l'aide sur cette commande, tapez dans un xterm : man remise). Respecter les noms des fichiers.

remise dift1020 tp3 Tp3.java Les_autres_fichiers

2. Remettez une **copie papier** d'un rapport qui devrait décrire votre programme (pour le contenu d'un rapport voir la FAQ sur la page web du cours).

3. N'oubliez pas de remettre avec votre rapport une **copie papier** de votre programme Java.

Barème

Il est à noter que votre programme doit être bâti sur une approche orientée objet. Ainsi donc, nous apporterons une grande attention à la manière avec laquelle vous avez conçu votre programme.

Ce TP3 est noté sur 12 points.

Compilation et respect des spécifications	**1**
Algorithmique, Codage, commentaires etc.	**4**
Rapport	**4**
Tests (fournis et non fournis)	**3**

En plus du précédent barème, vous risquez de perdre des points dans les cas suivants ….

- La non remise électronique (volontaire ou par erreur) est sanctionnée par la note 0.
- La non remise papier vous pénalise de 1 point.
- Les programmes ne contenant pas d'en-tête, -1 point.
- Un programme qui ne compile pas : 0/12.
- Un programme qui compile mais ne fait les choses prévues dans la spécification : 0/12.
- Le code doit être lisible, bien indenté et commenté.
- Le nom respect du nom de fichier, donc forcement il ne va pas compiler et un des points de la spécification n'a pas été respecté : 0/12.
- Aberration dans le codage : Même si tous les chemins mènent à Rome, faites l'effort nécessaire pour éviter de prendre le plus long! -0.5 et plus en fonction des aberrations détectées.

Des questions à propos de ce TP?

L'adresse email: dift1020@iro.umontreal.ca

Pour faciliter le traitement de votre requête, inclure dans le sujet de votre email, au moins la chaîne: [IFT1020]

Mise à jour

17-06-2005 diffusion