École Polytechnique de Montréal
Département de génie informatique

INF4402 – Systèmes répartis sur l'Internet

TP3 - Développement d'une application Distribuée de partage P2P de fichier avec RMI

Travail présenté à :

M. Abouabdallah Mohamed Kamal

Travail fait par :

DEBONNEL, Yann **1250256**

BRUN, Joel Sedo Afolabi **1238635**

Date :

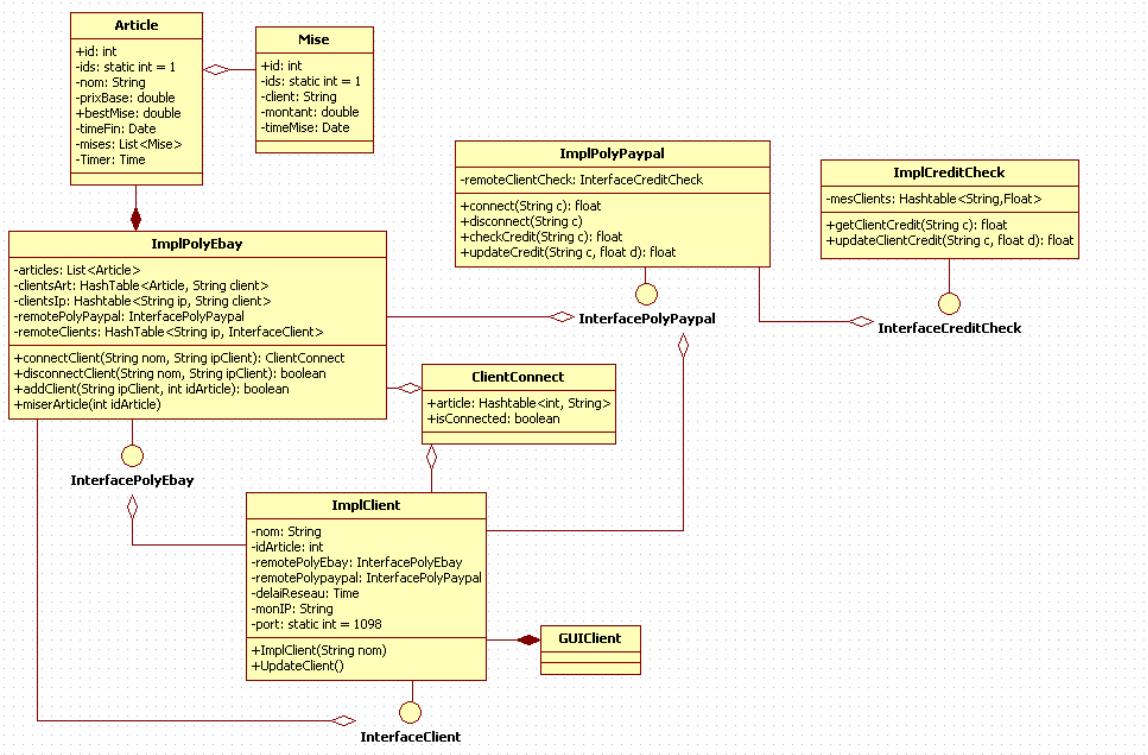
16/04/07

INTRODUCTION

Avec les avancées technologiques, il est très commun aujourd'hui de faire des achats par Internet. En effet, ce mode d'achat présente plusieurs avantages dont celui (non négligeable) d'éviter aux adeptes du "magasinage" des longues files d'attente dans les magasins.

Il est question dans le présent laboratoire d'implémenter un système de vente aux enchères électroniques. Ce système permet à divers clients de se procurer des articles à prix avantageux. Ce système doit fonctionner selon le principe de la vente aux enchères : un article est proposé avec un certain prix de base; puis, les acheteurs intéressés doivent proposer des prix pour l'acquérir; l'article reviendra au client ayant misé la somme la plus forte avant la fermeture de l'enchère. Vu que les différentes transactions se font entre les clients, il est impératif que ce système soit efficace et sécuritaire. Pour ce faire, nous utiliserons divers concepts relatifs aux systèmes distribués. Il s'agit de la synchronisation et des états globaux, la coordination des services de temps, le contrôle de la concurrence et les transactions réparties.

Schéma logique



Article

Classe permettant de représenter un article en vente sur le système PolyEbay. Elle contient le nom de l'article, son ID, son prix de base, sa date de mise en vente et la liste des mises qui lui ont été attribuées.

Mise

Classe permettant de représenter une mise effectuée sur un *Article* sur le système PolyEbay. Elle contient le nom du client l'ayant effectuée, son ID, son montant ainsi que sa date d'attribution.

ImplPolyEbay

Classe représentant le serveur central du système PolyEbay. Elle conserve l'information sur les clients et les articles mis en enchères. Elle transmet l'information aux clients et vérifie leurs crédits à l'aide de PolyPayPal.

ImplPolyPaypal

Classe représentant le serveur de PolyPayPal. Celui-ci peut vérifier grâce à CreditCheck l'état de crédit des clients et peut ainsi, valider ou non les transactions de PolyEbay.

ImplCreditCheck

Classe représentant le serveur CreditCheck qui conserve l'information de crédit des clients.

ImplClient

Classe représentant un client. Elle contient l'information sur le nom du client, son adresse IP.

ClientConnect

Classe utilisée pour la communication entre les différents serveurs. Permet en effet,de transporter l'information sous la forme d' un objet sérialisable sur le réseau.

GUIClient

Interface utilisateur du client. Permet l'interaction entre l'utilisateur et le système.

InterfacePolyPaypal , InterfaceCreditCheck et InterfaceClient

Ces classes sont les interfaces utilisées afin de procéder à des appels remote.

Problèmes de concurrence et de temps universelle

Le premier problème que nous avons rencontré était lorsque plusieurs instance du client était lancé sur le même ordinateur. Il y avait en effet un conflit au niveau des ports. Notre première solution était basé sur un compteur statique qui incrémentait la valeur à assigner au port à chaque assignation. Nous avons bien vite vu que cette solution était inadéquate puisque deux clients différents exécutaient deux processus différents et ne pouvaient donc pas partager la valeur statique. Nous avons envisagé une solution basée sur de l'échange d'information à travers des pipes mais nous avons jugé cette solution trop compliquée et longue à implémenter. Notre solution finale est de laisser l'utilisateur libre de définir lui-même son numéros de port lors de l'initialisation du système.

Environnement de test

Les premiers tests que nous avons fait était local à une machine. Nous exécutions un processus pour le gestionnaire de fichiers partagés et trois autres pour les utilisateurs. Nous nous assurions ensuite que les utilisateurs pouvaient se connecter au gestionnaire de fichiers partagés. Nous testions ensuite la fonctionnalité de transfert de fichiers.

Une fois ces tests complétés nous avons voulu tester notre solution sur un réseau local. Nous exécutions donc notre gestionnaire de fichiers partagés et nos utilisateurs sur des ordinateurs différents (plateforme LINUX). Nous récupérions les adresses IP des ordinateurs à l'aide d'une application externe et procédions aux tests de connexions et de transferts de fichiers.

Une fois les tests de connections passés, nous avons ensuite vérifier que le système fonctionne bien avec 3 clients différents qui misent sur le même livre. Nous avons vérifié que le livre revient à celui qui a misé le plus grand prix avant la fermeture de l'enchère.

Ensuite nous avons vérifié qu'un client malicieux ne puisse pas acheter en même temps 2 livres qui reviennent plus cher que l'argent dont il dispose dans son compte.

Finalement, nous avons vérifié que le client ne donne pas une mise plus de ce qu’il a en compte.

Conclusion

Au terme de ce laboratoire, nous avons assimilé la notion d'application distribuée par lebiais d'un programme de vente aux enchères sur internet de style eBay et avons rencontré les différentes difficultés qu'engendre ce type de système. En effet, nous avons eu les problèmes d'accès concurrent entre plusieurs processus et threads aux ressources qui entrainent en général la détérioration des données. Pour résoudre ces problèmes, nous avons utilisé le principe de synchronisation offert par Java à savoir les méthodes *synchronised*.