UNIVERSITÉ PARIS DAUPHINE

MASTER 1 MIAGE

Projet de Systèmes & Algorithmes Répartis

Réalisé par :
Lyes MEGHARA
Vitalina MALUSH
Ouerdia IDINARENE
Nadia MSELLEK

Encadré par :
Mme Joyce EL HADDAD

10 Janvier 2018

Table des matières

I Introduction

En nous aidant des procédés et méthodes apprises en cours de Systèmes et Algorithmes répartis, nous avons mis en place un système distribué permettant de créer une simulation boursière.

Nous avons mis en place un certain nombre d'entreprises souhaitant mettre en vente des actions (dites actions flottantes) sur la bourse, ainsi que des clients qui eux aussi souhaitent vendre leurs propres actions.
D'autres clients souhaitent eux acquérir des actions d'entreprises et sollicitent la bourse qui leur attribue un courtier pour effectuer toutes ces transactions, modélisées par des ordres.

II Outils de développement

Dans le cadre de ce projet, nous avons utilisé GitHub pour une meilleure collaboration, eclipse sous la JDK 1.8, Doxygène nous a permis de générer une documentation visible depuis le fichier index.html.
Aussi, l'outil ObjectAid intégré à Eclipse nous a permis de générer automatiquement le diagramme de classes via nos sources, et pour finir ce rapport a été rédigé en utilisant LaTeX.

III Description du projet

III.I Hypothèses

La première hypothèse que nous avons émise est que la bourse est d'abord lancée, s'en suit derrière le(s) courtiers(s) pour enfin lancer les clients.

Aussi, par souci de cohérence avec le fonctionnement des systèmes boursiers avec des clients dans la vie active, le Client envoie 3 ordres successivement en les transmettant à son courtier, qui lui les transmet à la Bourse pour effectuer le traitement, le client attend de recevoir ses réponses avant de décider de ses futures actions (achat/vente).

III.II Choix techniques

Après une réflexion entre *RMI* et *Socket TCP*, nous avons finalement décidé d'utiliser les Sockets pour les communications entre nos différentes classes ; d'une part, c'est un modèle que nous connaissions déjà, et d'autre part, les sockets permettaient d'envoyer et de récupérer des objets (*serializables*) via les objets *ObjectOutputStream* ainsi que *ObjectInputStream* et respectivement les méthodes : *readObject()* et *writeObject()*.

Aussi, dans un environnement *multi-thread*, l'utilisation des Vector<> a été préviligiée face aux ArrayList<> du fait de leur propriété *safe-thread*.

Autre chose, pour connaitre l'état du marché, le client possède une *map<NomEntreprise,Prix>* qui correspond donc à *<String,Integer>* le fait de passer le nom de l'entreprise implique la recherche de celle-ci en tant qu'objet pour accéder à ses informations, ce qui impose un détour, néanmoins ce choix nous a paru plus pertinent, car un client n'est pas censé connaitre toutes les informations d'une entreprise juste parce qu'il est un potentiel actionnaire de celle-ci, et que le but de cette map est juste de récupérer le prix de l'action pour chaque (*nom*) entreprise.

Concernant le client cette fois-ci, comme dis explicitement dans l'énoncé, celui-ci lorsqu'il émet un ordre d'achat (resp, de vente) son portefeuille n'est pas réajusté directement, mais uniquement lorsque son courtier reçoit la confirmation de la bourse, pour rester en cohérence avec la vie réelle, celui si prend compte de l'ordre qu'il vient de passer, et enregistre cette dépense éventuelle pour en prendre acte lors de ses futurs achats, l'introduction de l'attribut *depensesEventuelles* (resp *quantiteEventuelleVendu*) nous a permis d'implémenter cette situation, ces derniers sont aussi utilisés dans les méthodes *achatLegal()* et *VenteLegal()* .

III.III Pseudo code

Algorithm 1 Client : Produit un ordre

```
procedure PRODUIRE(Ordre)
    EnvoyerA(Courtier, Ordre)
end procedure
```

Algorithm 2 Client : Transmet un ordre au courtier

```
procedure ECHANGEORDRECLIENTCOURTIER())
    for each Ordre o in ListOrdres do
        Produire(o)
    end for
    Attendre(Délai)
    for each Ordre o in ListOrdres do
        RecevoirReponse()
    end for
    Deconnexion()
end procedure
```

Algorithm 3 Courtier : Transmet un ordre au courtier a la bourse

```
procedure TRANSMETTRE(Ordre)
    Envoyer_A Bourse(Ordre)
end procedure
```

Algorithm 4 ThreadBourse : Transmet un ordre

```
procedure SURRECEPTIONDE(Ordre)
    AjouterALaListedeBourse(o)
    AjouterALaListeEntreprise(o)
    if (OrdreAchat) IncDemandesAchats
    else IncDemandesVentes
end procedure
```

Algorithm 5 Bourse : Consommer un ordre

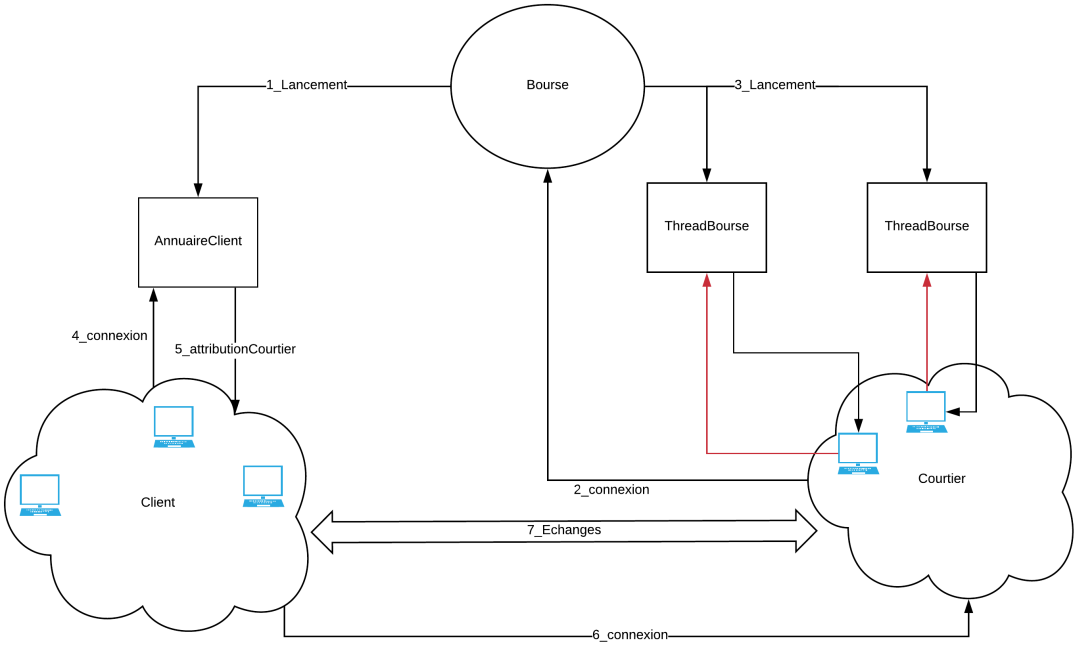
```
procedure CONSOMMER (nomCourtier)
    for each Ordre o in ListOrdres do
        if (Ordre ProvientDe nomCourtier)
        SupprimerDeLaListe(Ordre)
        renvoyer(Ordre)
        endif
    end for
end procedure
```

Algorithm 6 Bourse : Accord

```
procedure ACCORD (nomCourtier)
    Consommer (nomCourtier)
    RepondreAuxDeuxCourtiers()
end procedure
```

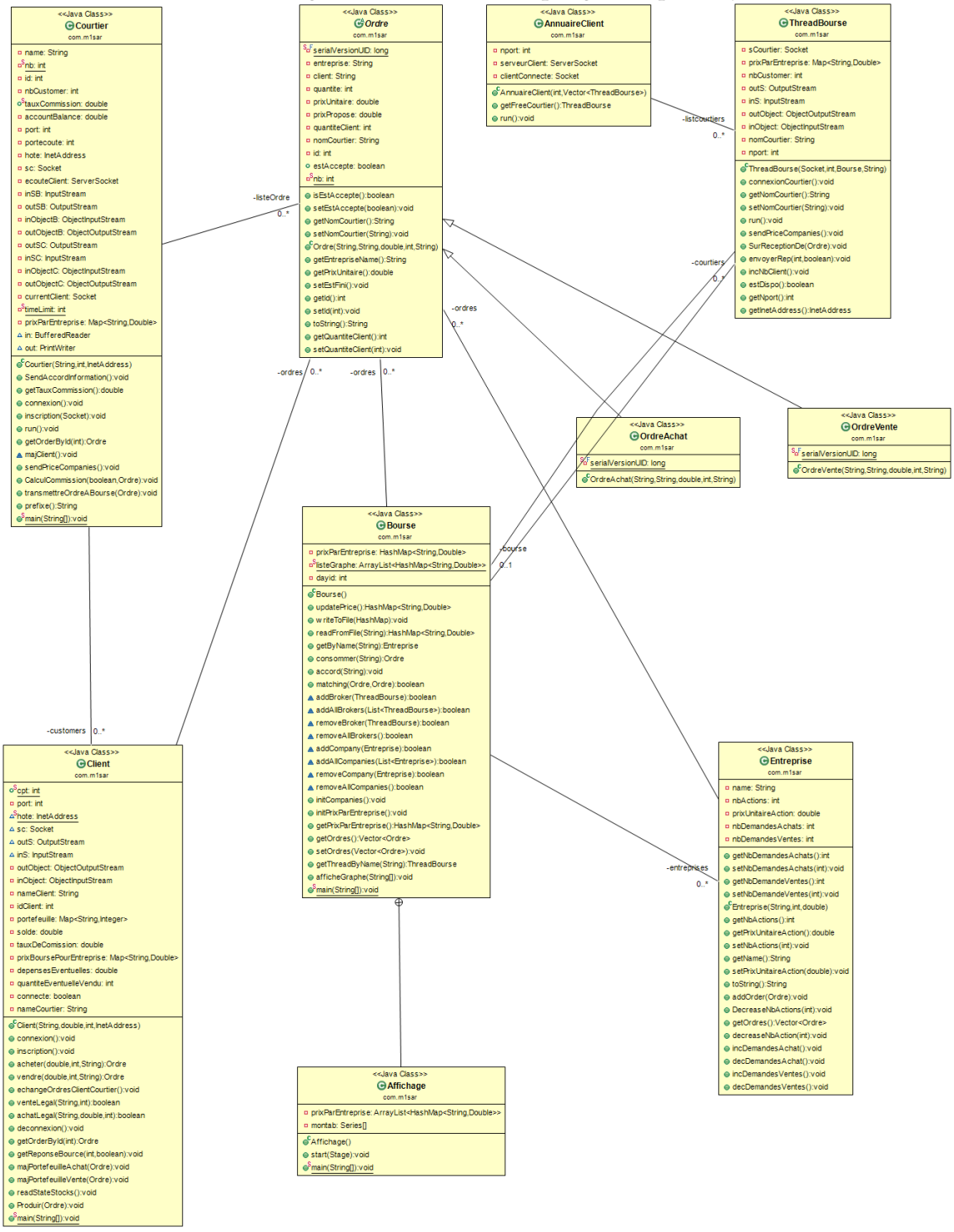
IV Graphe de communication

FIGURE 1 – Graphe de communications



V Diagramme de classe

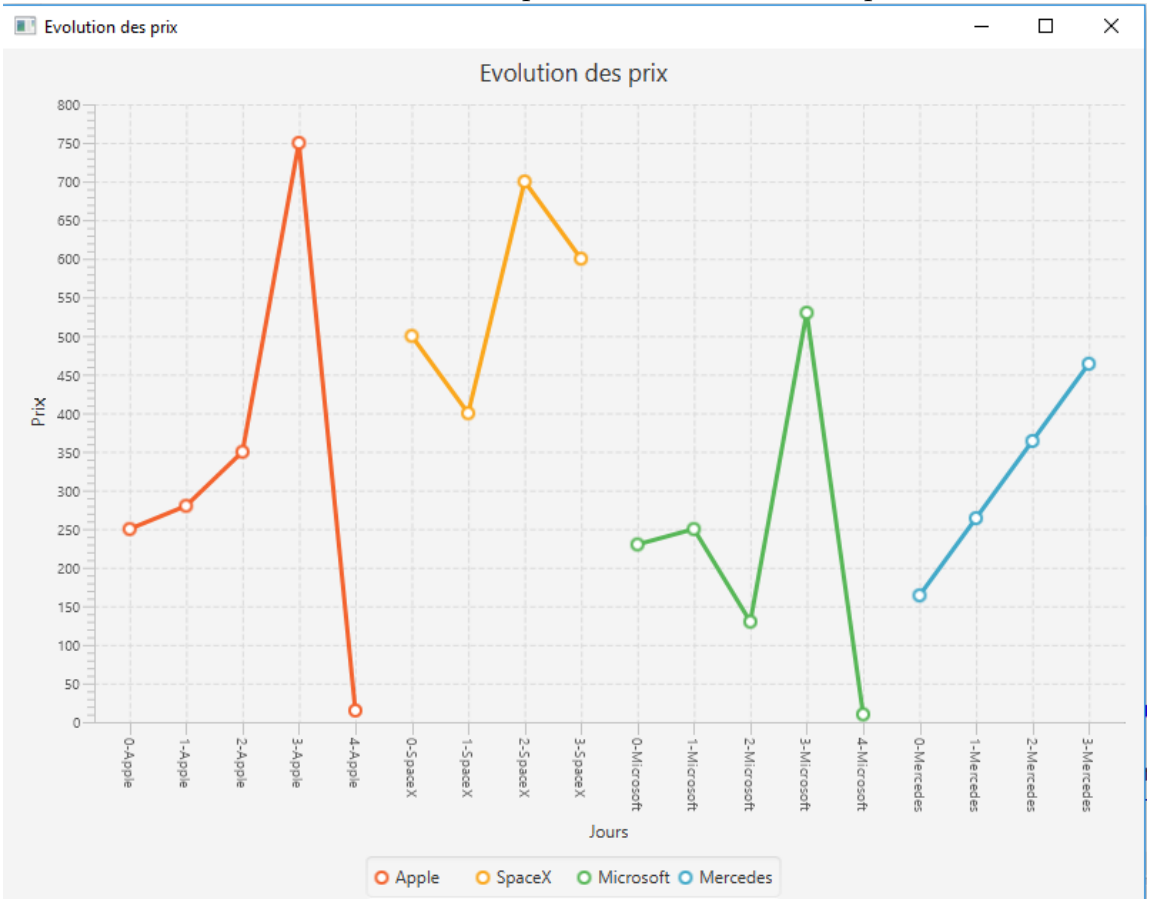
FIGURE 2 – Diagramme de classe tel que généré par ObjectAid



VI Nos initiatives

Dans le but d'aller plus loin, nous avons intégré un affichage représentant le cours d'évolution des prix par entreprise, et selon les jours, cet affichage est réalisé avec *JavaFX*, et fait partie, en tant que classe interne de la classe Bourse.

FIGURE 3 – Courbes représentant l'évolution des prix



Aussi, nous voulions tirer parti des avantages des systèmes répartis en mettant en place une politique de panne, le principe auquel nous avions pensé est le suivant : la bourse (serveur principal) détecte (*catch*) une exception, et se met donc à lancer un serveur secondaire, dans ce catch la bourse génère un clone d'elle même (interface *cloneable*), et le relance (*invoke*) via son main.

Cependant, lors de la coupure de cet objet actuel, les connexions avec les autres acteurs (Clients, Courtiers ...) Est aussi interrompu, l'objet cloné et relancé ne les récupère pas, car ces derniers génèrent à leurs tours des exceptions du à la connexion interrompue.

Une *API* trouvée sur GitHub nous semblait pouvoir pallier efficacement à ce problème, mais nous en sommes restés là par manque de temps.
Pour garder un historique des prix et de leurs évolutions, ces derniers sont *serializés* dans des fichiers à la fin de chaque journée, via la méthode *writeToFile()* .

VII Scénario d'éxecution

VII.I Manuel d'utilisation

Il convient d'abord de compiler les classes, pour ce faire, sur un environnement Linux, lancer simplement compile.sh (chmod + x peut-être nécessaire)
Sur un environnement Windows, lancer compile.exe.
Une fois la compilation faite, il suffit de lancer les programmes dans l'ordre suivant :

Bourse [NPort]
Courtier [NPort] [IP]
Client [NPort] [IP]

IP peut correspondre à localhost dans le cas d'une exécution en local.

VII.II Phase de connexion

Un client se connecte d'abord à l'annuaire que publie la bourse où un courtier lui est affecté ensuite ,le client se connecte à ce courtier.

1-Lancer la bourse avec un numéro de port p1=4000.
2-Lancer un courtier "James" en spécifiant le numéro de port p1=4000 et @IP de la bourse.
3-Lancer un autre courtier "Mario" en spécifiant le numéro de port p1=4000 et @IP de la bourse.
4-Lancer un client "Jack" en spécifiant le numéro de port p2=4001 et @IP de la bourse.
5-Lancer un client "Rose" en spécifiant le numéro de port p2=4001 et @IP de la bourse.
6-Lancer un client "Eva" en spécifiant le numéro de port p2=4001 et @IP de la bourse.
7-Lancer un client "Toto" en spécifiant le numéro de port p2=4001 et @IP de la bourse.

VII.III Phase d'échanges

Hypothèses : Le client peut envoyer autant d'ordres qu'il veut, deux par deux, c'est à dire, il envoie deux ordres, ensuite attends la réponse de la bourse pour envoyer la suite. Le courtier accepte deux clients à la fois, mais traite ces clients séquentiellement.

Éva crée 4 ordres :
un ordre d'achat : a,prix=15,quantité=2,entreprise=Apple
Jack crée 5 ordres :
un ordre d'achat : a,prix=15,quantité=3,entreprise=Apple
Jack crée un ordre d'achat : a,prix=10,quantité=3,entreprise=Microsoft
Éva crée un ordre d'achat : a,prix=6,quantité=2,entreprise=Ubisoft
Éva crée un ordre d'achat : a,prix=10,quantité=1,entreprise=Microsoft
Jack crée un ordre de vente : v,prix=15,quantité=50,entreprise=Apple
Jack crée un ordre de vente :v,prix=9,quantité=2,entreprise=Microsoft
Éva crée un ordre d'achat :a,prix=9,quantité=2,entreprise=Microsoft
Éva reçoit les réponses : ces ordres d'achat sont validés
Déconnexion d'Eva
Jack crée un ordre de vente :v,prix=10,quantité=2,entreprise=Apple
Déconnexion de Jack
un ordre d'achat : a,prix=15,quantité=3,entreprise=Apple
Rose ne crée aucun ordre (ordre=0)
déconnexion de James et Mario
La bourse incrémente la journée et met à jour les prix des entreprises en prenant en compte le nombre des ordres émis pour chaque entreprise.

VIII Captures d'écran

```
Bourse est ouverte
Le jour numero : 0
Le serveur courtier est a l'ecoute sur le port 5050
La bourse attend un courtier
Le serveur client est a l'ecoute sur le port 5051

Donnez le nom du courtier :
james
Je me suis connecte a la bourse

Donnez le nom du courtier :
Mario
Je me suis connecte a la bourse

Jack
Connexion au courtier reussi
james : Bienvenu cher client Jack0, vous pouvez envoyez vos ordre
Client Jack0 veut savoit l'etat du marche
Voila l'etat du marche :
{Apple=15.0, Kerima Moda=2.0, Microsoft=10.0, Ubisoft=6.0}
Echange des ORDRES
Donnez le nbOrdres a creer :
Donnez le nom du client :
rose
Connexion au Courtier reussi

Eva
Connexion au courtier reussi
mario : Bienvenu cher client Eva0, vous pouvez envoyez vos ordre
Client Eva0 veut savoit l'etat du marche
Voila l'etat du marche :
{Apple=15.0, Kerima Moda=2.0, Microsoft=10.0, Ubisoft=6.0}
Echange des ORDRES
Donnez le nbOrdres a creer :


Donnez le nbOrdres a creer : 4
Donnez l Ordre a cree 'v'-Vente ou 'a'-Achat : a
Donnez le nom de l entreprise : Apple
Donnez le prix : 15
Donnez le nombre d actions a acheter ou vendre : 2
Client Eva0 envoie un Ordre d Achat au Courtier : Apple, quantite : 2,  prix : 15.0
OrdreAchat bien envoyer


j'ai recu l'ordre 1 du clientEva0
j'ai transmis l'ordre a la bourse

Donnez le nbOrdres a creer : 5
Donnez l Ordre a cree 'v'-Vente ou 'a'-Achat : a
Donnez le nom de l entreprise : Apple
Donnez le prix : 15
Donnez le nombre d actions a acheter ou vendre : 3
Client Jack0 envoie un Ordre d Achat au Courtier : Apple, quantite : 3,  prix : 15.0
OrdreAchat bien envoyer

j'ai recu l'ordre 1 du clientJack0
j'ai transmis l'ordre a la bourse


Client Jack0 envoie un Ordre d Achat au Courtier : Microsoft, quantite : 3,  prix : 10.0
OrdreAchat bien envoyer
J attends la reponse de la Bourse

j'ai recu l'ordre 2 du clientJack0
j'ai transmis l'ordre a la bourse
j'attend que la bourse me réponde

mon solde après l'ordre N° 1 qui est accepte  est: 4.5
j'attend que la bourse me réponde
mon solde après l'ordre N° 2 qui est accepte  est: 7.5
```

```
OrdreAchat traite : Apple, quantite : 3, reponse de la Bourse : true
Portefeille de Client : {Apple=3}
Solde de Client : 160.5
OrdreAchat traite : Microsoft, quantite : 3, reponse de la Bourse : true
Portefeille de Client : {Apple=3, Microsoft=3}
Solde de Client : 127.5

Client Eva0 envoie un Ordre d Achat au Courtier : Ubisoft, quantite : 2,  prix : 6.0
OrdreAchat bien envoyer
J attends la reponse de la Bourse

j'ai recu l'ordre 2 du clientEva0
j'ai transmis l'ordre a la bourse
j'attend que la bourse me réponde

mon solde après l'ordre N° 1 qui est accepte  est: 3.0
j'attend que la bourse me réponde
mon solde après l'ordre N° 2 qui est accepte  est: 4.2
OrdreAchat traite : Apple, quantite : 2, reponse de la Bourse : true
Portefeille de Client : {Apple=2}
Solde de Client : 177.0
OrdreAchat traite : Ubisoft, quantite : 2, reponse de la Bourse : true
Portefeille de Client : {Apple=2, Ubisoft=2}
Solde de Client : 163.8

Donnez le nombre d actions a acheter ou vendre : 50
ce Vente n est pas legal

Client Jack0 envoie un Ordre de Vente au Courtier : Microsoft, quantite : 2,  prix : 9.0
OrdreVente bien envoyer

j'ai recu l'ordre 3 du clientEva0
j'ai transmis l'ordre a la bourse
j'ai recu l'ordre 4 du clientEva0
j'ai transmis l'ordre a la bourse
j'attend que la bourse me réponde

mon solde après l'ordre N° 3 qui est accepte  est: 5.2
j'attend que la bourse me réponde
mon solde après l'ordre N° 4 qui est accepte  est: 7.0

OrdreAchat traite : Microsoft, quantite : 1, reponse de la Bourse : true
Portefeille de Client : {Apple=2, Microsoft=1, Ubisoft=2}
Solde de Client : 152.8
OrdreAchat traite : Microsoft, quantite : 2, reponse de la Bourse : true
Portefeille de Client : {Apple=2, Microsoft=3, Ubisoft=2}
Solde de Client : 133.0
Client Eva0 a fini d envoyer des ORDRES
Client Eva0 se deconnecte

Client jack0 a fini d envoyer des ORDRES
Client jack0 se deconnecte

Donnez le nbOrdres a creer : 0
Client rose0 a fini d envoyer des ORDRES
Client rose0 se deconnecte

james : Je n'ai aucun client, J'attend si un client me contacte
james : Je n'ai plus de clients, je me deconnecte de la bourse
mario : Je n'ai aucun client, J'attend si un client me contacte
mario : Je n'ai plus de clients, je me deconnecte de la bourse
```