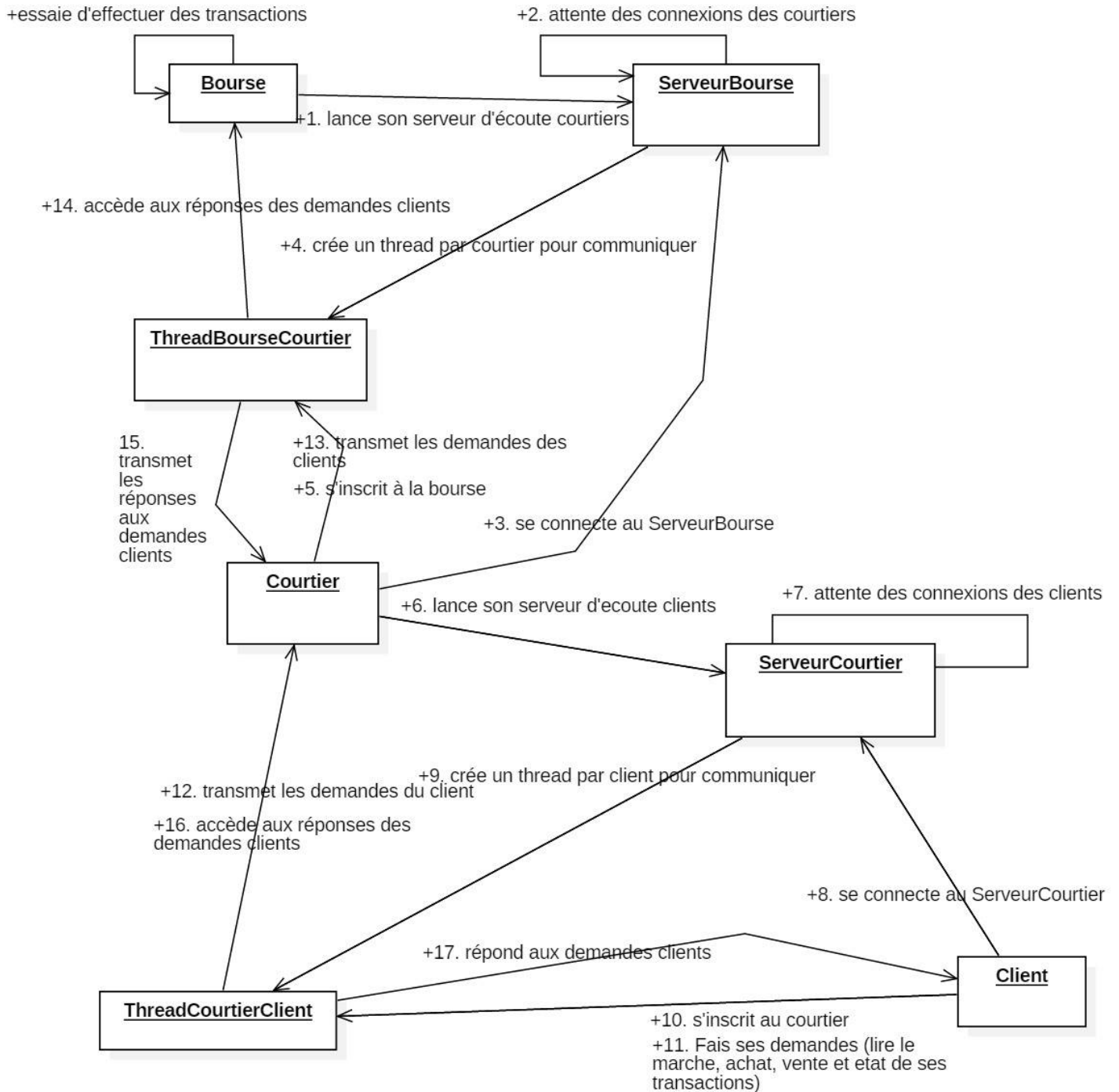


**Rapport Projet SAR – Application Boursière**  
**Mikaël Azoulay, Eliahou Bensimon, Betty Bismuth et Mickaël Blondel**

**I – Graphe de communications du système**



## II – Pseudo-code de l'application répartie

### 1) Courtier

Variables :

- nomCourtier : entier unique permettant d'identifier le courtier
- tauxCommission : donne le taux de commission demandé par le courtier
- especes : argent possédé par le courtier
- listeClient : hashmap contenant les clients du courtier et qui associe à chaque nomClient son état (ouvert ou fermé)
- societes : liste qui stocke les sociétés proposant actuellement des actions sur le marché
- commandesEffectuees : liste des transactions terminées
- etatCourtier : ouvert ou fermé

Primitives :

Sur réception de(emetteur, message, type)

**debut**

**si** (type == inscription) alors

    listeClient += emetteur ;

    Envoyer\_à (emetteur, nomCourtier, « ack ») ;

    Envoyer\_à (bourse, nomCourtier, « nvClient ») ;

**fin si**

**si** emetteur ∈ listeClient alors

**si** (type == marche) alors

        Envoyer\_à (emetteur, societes, « marche ») ;

**fin si**

**si** (type == achat || type == vente) alors

**si** Produire() == true alors

            Envoyer\_à (emetteur, « comOK ») ;

**sinon**

            Envoyer\_à (emetteur, « comNO ») ;

**fin si**

**si** (type == resultat) alors

        Envoyer\_à (emetteur, commandesEffectuees, « notif ») ;

**fin si**

**si** (type == fermer) alors

        listeClient.emetteur.etat = ferme

**si** pour chaque c ∈ listeClient, c.etatClient == ferme alors

            etatCourtier = ferme ;

            Envoyer\_à (bourse, nomCourtier, « rmClient ») ;

**fin si**

**fin si**

**fin si**

**si** (emetteur == bourse) alors

**si** (type == ack) alors

        societes = message ;

**fin si**

**si** (type == comOK) alors

        Consommer() ;

**fin si**

**si** (type == notif) alors

```
commandesEffectuees += message ;  
especes += message.prix * message.nbActions*tauxCommission ;
```

```
fin si
```

```
fin si
```

```
fin
```

## 2) Client

Variables :

- nomClient : entier unique permettant d'identifier le client
- nomCourtier : entier représentant le courtier auprès duquel le client est inscrit
- portefeuille : hashmap donnant le nombre d'actions détenues par le client en fonction de la société
- especes : argent possédé par le client
- societes : garde en mémoire les sociétés actuellement sur le marché
- commandesEffectuees : liste des transactions terminées
- etatClient : ouvert ou fermé

Primitives :

Sur reception de (emetteur, message, type)

```
debut
```

```
si (type == ack) alors
```

```
    nomCourtier = message ;
```

```
    etatClient = ouvert ;
```

```
fin si
```

```
si (emetteur == nomCourtier) alors
```

```
    si (type == comOK) alors
```

```
        afficher(« La commande est passée ») ;
```

```
    fin si
```

```
    si (type == comNO) alors
```

```
        afficher(« La commande n'a pu aboutir, réessayez ultérieurement ») ;
```

```
    fin si
```

```
    si (type == marche) alors
```

```
        societes = message ;
```

```
    fin si
```

```
    si (type == notif) alors
```

```
        commandesEffectuees = message ;
```

```
    fin si
```

```
fin si
```

```
fin
```

## 3) Bourse

Variables :

- connexion : hashmap contenant les courtiers et qui associe à chaque nomCourtier son nombre de clients connectés
- etatBourse : ouverte ou fermée
- societes : liste qui stocke les sociétés proposant actuellement des actions sur le marché
- commandesEffectuees : liste des transactions terminées

Primitives :

Sur réception de(emetteur, message, type)

```
debut
```

```
si (type == inscription) alors
```

```

        connexion += (emetteur,0) ;
        Envoyer_à (emetteur, « ack ») ;
        Envoyer_à (emetteur, societes) ;
    fin si
    si (type == nvClient) alors
        connexion+=(emetteur,+1);
    fin si
    si (type == rmClient) alors
        connexion+=(emetteur,-1);
        si (pour tout emetteur j on a connexion(j,0)) alors
            etatBourse=ferme ;
        fin si
    fin si
    si (type == commander) alors
        si (produire(message)==true)alors
            Envoyer_à (emetteur, « comOK ») ;
        sinon
            Envoyer_à (emetteur, « comNO ») ;
        fin si
    fin si
    si( il existe commande c ∈ commandesEffectuees tq nomCourtierCommande ==
emetteur)

        Envoyer_à (emetteur, « notif ») ;
        commandesEffectuees-=c ;
    fin si
fin

```

#### 4) Producteur/Consommateur

Variables :

- Tampon : objet ayant pour attribut une BlockingQueue queue de taille 10 et des méthodes retirer() et deposer(commande)
- nbElements : nombre d'éléments dans la queue du tampon

Primitives :

Produire(commande)

```

debut
    si (nbElements < 10) alors
        deposer(commande) ;
    fin si
fin

```

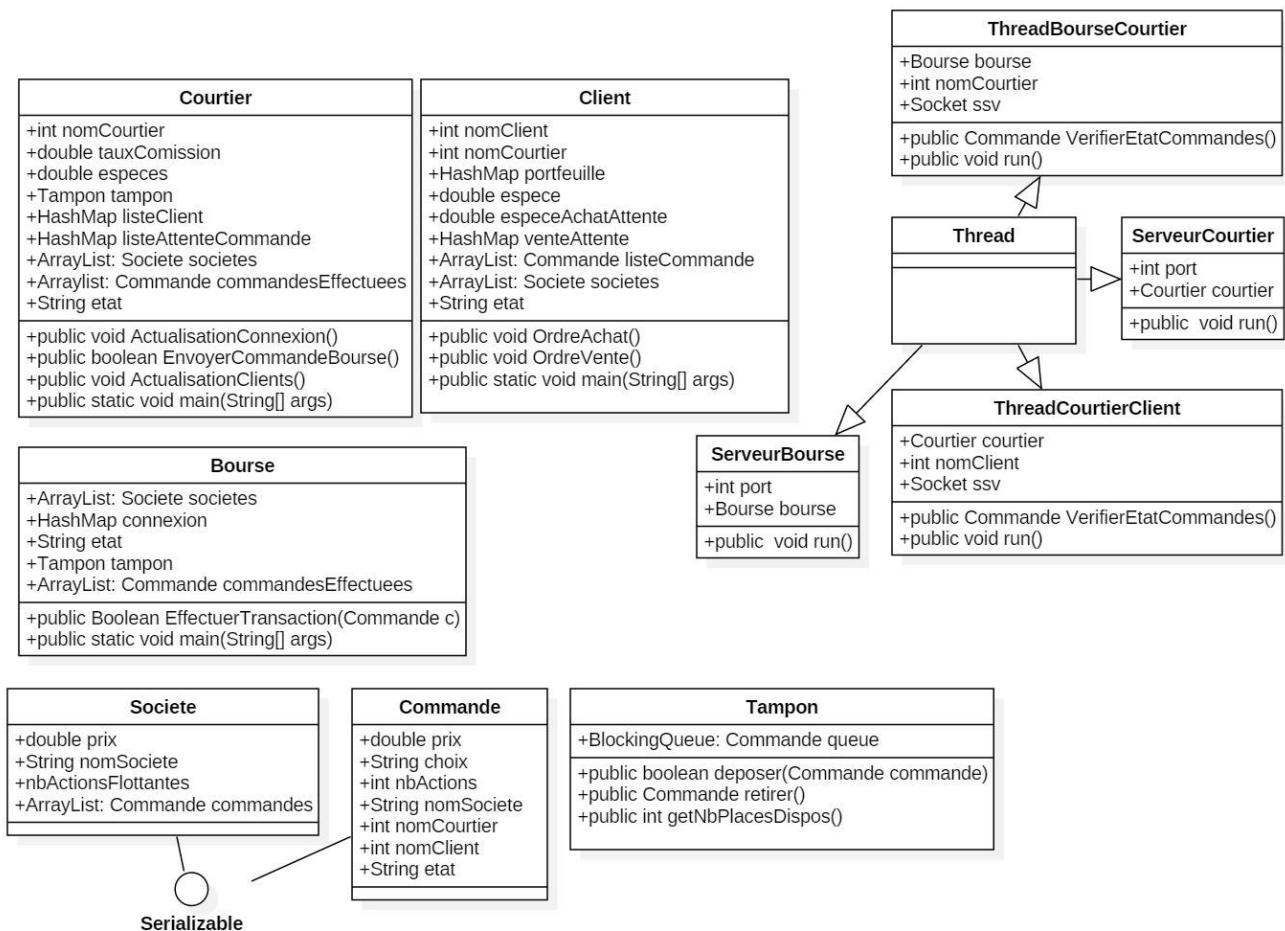
Consommer()

```

debut
    si queue n'est pas vide alors
        retirer() ;
    fin si
fin

```

### III – Diagramme des classes



## IV – Explication et justification des choix techniques

## 1) Communication en TCP multithread

Au début de notre réflexion, nous pensions coder le projet en RMI. Cependant, nous avons trouvé que la communication en TCP permettait de mieux recevoir mais également envoyer les messages entre les différentes entités.

L'application est construite de la manière suivante :

- La Bourse est le serveur du Courtier, dès qu'un courtier souhaite se connecter à la bourse, un ThreadBourseCourtier est lancé, et les deux entités pourront alors communiquer via ce thread.
- Le Courtier est le serveur du Client, dès qu'un client souhaite se connecter à un courtier, un ThreadCourtierClient est lancé, et les deux entités pourront communiquer via ce thread.

Il n'y a donc aucune communication directe entre la Bourse et le Client, tout doit passer par l'intermédiaire du courtier

## 2) Implémentation du protocole Producteur/Consommateur

Lors de l'envoi des commandes du client à son courtier, mais aussi du courtier à la bourse, on considère que le courtier et la bourse ne doivent pas être « débordés » par un flux de commandes trop grand. Pour parer à ce problème, nous avons implémenté un protocole Producteur/Consommateur à l'aide de BlockingQueue.

Nous avons créé une classe Tampon qui contient une BlockingQueue queue de taille 10 ainsi que deux méthodes, déposer(Commande c) qui permet de déposer une commande dans la queue si il y a encore de la place, et retirer() qui permet de retirer une commande de la queue s'il y en a une.

Ces deux méthodes disposent d'un délai d'attente de quelques millisecondes, permettant possiblement à la queue de libérer une place durant cet intervalle de temps. A la fin de ce délai, on sort de la méthode sans bloquer sur cette instruction.

Ainsi, dans le cas d'une commande passée par le Client, celui-ci envoie les données de la commande via le ThreadCourtierClient. Le thread tente alors de la déposer dans le tampon possédé par son courtier. S'il y parvient, un message est envoyé au client pour l'avertir que sa commande est passée. Sinon, le client recevra un message lui conseillant de réitérer sa commande ultérieurement.

Du côté du Courtier, une méthode EnvoyerCommandeBourse est lancée, elle permet de copier la tête de la queue présente dans courtier, de l'envoyer à la bourse, et seulement si elle est bien reçue dans le tampon de la bourse, elle pourra être retirée de la queue.

De la même manière, on implémente le protocole producteur/consommateur entre le courtier et la bourse, celle-ci disposant également d'un tampon avec une BlockingQueue.

On a donc bien une limite au niveau du nombre de commandes que peuvent recevoir le courtier et la bourse, et ce sans que cela bloque les différents acteurs.

### 3) ObjectOutputStream et ObjectInputStream

Afin de pouvoir envoyer des commandes sous forme d'objet Commande ou des listes mais aussi des String, nous avons utilisé des ObjectOutputStream et ObjectInputStream au lieu des OutputStream et InputStream habituels. Ainsi les méthodes utilisées pour l'envoi et la réception sont writeObject(Object o) et readObject().

### 4) Réalisation d'une transaction

Lorsque la bourse tente d'effectuer une commande, elle va donner priorité aux actions flottantes. Puis si sa recherche est infructueuse, elle regarde les commandes en attente de ses clients.

Il faut distinguer deux cas : l'achat et la vente.

Pour l'achat, si il y a assez d'actions et que le prix désiré dans la commande est supérieur ou égal au prix du marché, on effectue l'achat, soit au prix désiré (si égal), soit en faisant une plus-value (si supérieur). Dans le cas de la plus-value le prix indiqué dans la commande est alors mis à jour pour que les modifications du portefeuille soient exactes.

Pour la vente, si il y a assez d'actions et que le prix désiré dans la commande est inférieur ou égal au prix du marché, on effectue la vente, soit au prix désiré (si égal), soit en faisant une plus-value (si inférieur). Dans le cas de la plus-value le prix indiqué dans la commande est alors mis à jour pour que les modifications du portefeuille soient exactes.

### 5) Notification d'une transaction

Lorsque la bourse a effectué une transaction, elle la garde en mémoire dans une liste « commandesEffectuees ».

Dans ThreadBourseCourtier, on appelle à plusieurs reprises sa méthode verifierEtatCommandes() qui lit la liste de la Bourse et récupère une commande concernant le courtier. On envoie ensuite un message « notif » puis la commande en question au courtier qui l'ajoute à sa propre liste « commandesEffectuees ».

L'utilisateur dans le terminal associé au client va pouvoir taper la commande « resultat ».

Cela notifie le ThreadCourtierClient qui lance sa méthode verifierEtatCommandes(), renvoyant cette fois la liste des commandes concernant le client. On envoie ensuite cette liste au client qui va pouvoir mettre à jour son portefeuille.

## V – Documentation utilisateur

L'utilisateur compile les fichiers en écrivant en ligne de commande, **javac \*.java**.

L'utilisateur ouvre un terminal pour la Bourse, un terminal pour chaque Courtier et un terminal pour chaque Client et lance les commandes respectivement :

```
java Bourse
java Courtier <adresseIPBourse> <portBourse>
java Client <adresseIPCourtier> <portCourtier>
```

La bourse est alors fonctionnelle.

Dans le terminal Courtier, il faut taper **inscription** puis un entier comme nom de courtier. Le courtier est alors fonctionnel.

Dans le terminal Client, il faut taper **inscription** puis un entier comme nom de client. Le client est alors fonctionnel.

L'utilisateur va alors pouvoir effectuer certaines commandes, en tapant :

**marche** : obtient l'état actuel du marché avec les actions proposées par chaque société et leur prix

**achat** : si le client veut acheter des actions, il devra alors spécifier auprès de quelle société, à quel prix et combien d'actions

**vente** : si le client veut vendre des actions, il devra alors spécifier auprès de quelle société, à quel prix et combien d'actions

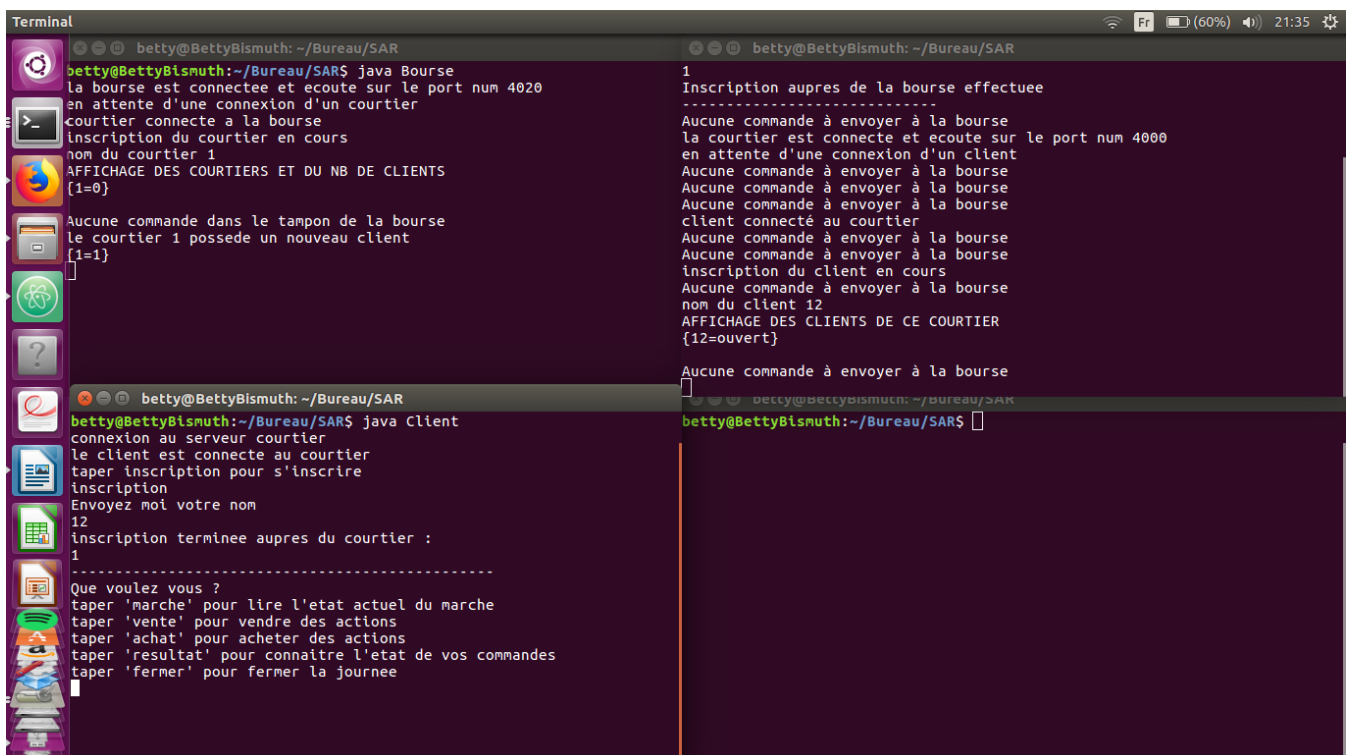
**resultat** : pour savoir si ses commandes ont abouti ou non et ainsi mettre à jour son portefeuille

**fermer** : pour fermer sa journée

**connexion** : pour ouvrir une nouvelle journée

## VI – Captures d'écran

Connexion de la bourse (haut gauche), d'un courtier (haut droite) et d'un client (bas gauche)



```
Terminal
betty@BettyBismuth: ~/Bureau/SAR
betty@BettyBismuth:~/Bureau/SAR$ java Bourse
La bourse est connectee et ecoute sur le port num 4020
en attente d'une connexion d'un courtier
courtier connecte a la bourse
inscription du courtier en cours
nom du courtier 1
AFFICHAGE DES COURTIER ET DU NB DE CLIENTS
{1=0}
Aucune commande dans le tampon de la bourse
le courtier 1 possede un nouveau client
{1=1}

betty@BettyBismuth: ~/Bureau/SAR
betty@BettyBismuth:~/Bureau/SAR$ java Client
connexion au serveur courtier
le client est connecte au courtier
taper inscription pour s'inscrire
inscription
Envoyez moi votre nom
12
inscription terminee aupres du courtier :
1
-----
Que voulez vous ?
taper 'marche' pour lire l'etat actuel du marche
taper 'vente' pour vendre des actions
taper 'achat' pour acheter des actions
taper 'resultat' pour connaitre l'etat de vos commandes
taper 'fermer' pour fermer la journee

betty@BettyBismuth: ~/Bureau/SAR
1
Inscription aupres de la bourse effectuee
-----
Aucune commande à envoyer à la bourse
la courtier est connecte et ecoute sur le port num 4000
en attente d'une connexion d'un client
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
client connecté au courtier
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
inscription du client en cours
Aucune commande à envoyer à la bourse
nom du client 12
AFFICHAGE DES CLIENTS DE CE COURTIER
{12=ouvert}
Aucune commande à envoyer à la bourse

betty@BettyBismuth: ~/Bureau/SAR$
```

Affichage du marché chez le client (en bas à droite) et envoi d'une commande d'achat (en bas à gauche)

```
Terminal
betty@BettyBismuth: ~/Bureau/SAR
en attente d'une connexion d'un courtier
courtier connecte a la bourse
inscription du courtier en cours
nom du courtier 1
AFFICHAGE DES COURTIER ET DU NB DE CLIENTS
[1=0]

Aucune commande dans le tampon de la bourse
le courtier 1 possede un nouveau client
[1=1]
Aucune commande dans le tampon de la bourse
le courtier 1 possede un nouveau client
[1=2]
Aucune commande dans le tampon de la bourse
Une transaction est en cours
la transaction n'a pas ete effectuee
Commande [prix=15.0, choix=0, nbActions=2, nomSociete=apple, nomClient=12, nomCourtier=1, etat=encours]
commande mise dans le tampon de la bourse

betty@BettyBismuth: ~/Bureau/SAR
taper 'fermer' pour fermer la journee
achat
-----
Societe ?
apple
Prix ?
15
Combien d'actions ?
2
-----
693.7704287382327

Ressources suffisantes, envoi de l'ordre d'achat au courtier...
Votre commande d'achat est arrivee chez votre courtier
-----
Que voulez vous ?
taper 'marche' pour lire l'etat actuel du marche
taper 'vente' pour vendre des actions
taper 'achat' pour acheter des actions
taper 'resultat' pour connaitre l'etat de vos commandes

betty@BettyBismuth: ~/Bureau/SAR
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
-----
ACHAT DU CLIENT 12
Societe : apple
Prix : 15.0
Nombre d'action : 2
le courtier 1 a reçu la commande d'ACHAT du client 12 et la mise dans son
tampon
Commande [prix=15.0, choix=0, nbActions=2, nomSociete=apple, nomClient=12,
nomCourtier=1, etat=encours]
Une commande est en cours d'envoi a la bourse
comOK
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse

betty@BettyBismuth: ~/Bureau/SAR
Que voulez vous ?
taper 'marche' pour lire l'etat actuel du marche
taper 'vente' pour vendre des actions
taper 'achat' pour acheter des actions
taper 'resultat' pour connaitre l'etat de vos commandes
taper 'fermer' pour fermer la journee
marche
-----
Etat du marche :
SOCIETE : apple PRIX : 20.0 ACTIONS FLOTTANTES :10
SOCIETE : amazon PRIX : 10.0 ACTIONS FLOTTANTES :800
SOCIETE : samsung PRIX : 40.0 ACTIONS FLOTTANTES :100
SOCIETE : peugeot PRIX : 60.0 ACTIONS FLOTTANTES :80
-----
Que voulez vous ?
taper 'marche' pour lire l'etat actuel du marche
taper 'vente' pour vendre des actions
taper 'achat' pour acheter des actions
taper 'resultat' pour connaitre l'etat de vos commandes
taper 'fermer' pour fermer la journee
```

Affichage du résultat (en bas à gauche)

```
Terminal
betty@BettyBismuth: ~/Bureau/SAR
{1=2}
Aucune commande dans le tampon de la bourse
Une transaction est en cours
La transaction n'a pas ete effectuee
Commande [prix=15.0, choix=0, nbActions=2, nomSociete=apple, nomClient=12, nomCourtier=1, etat=en cours]
commande mise dans le tampon de la bourse
Aucune commande dans le tampon de la bourse
Aucune commande dans le tampon de la bourse
Aucune commande dans le tampon de la bourse
Une transaction est en cours
Commande [prix=20.0, choix=0, nbActions=2, nomSociete=apple, nomClient=12, nomCourtier=1, etat=en cours]
commande mise dans le tampon de la bourse
Commande [prix=20.0, choix=0, nbActions=2, nomSociete=apple, nomClient=12, nomCourtier=1, etat=effectuee]
la transaction a ete effectuee et on doit maintenant informer le client
Aucune commande dans le tampon de la bourse

betty@BettyBismuth: ~/Bureau/SAR
taper 'fermer' pour fermer la journee
resultat
-----
Votre commande pour 2 actions de la societe apple au prix de 20.0 a abouti
-----
Votre solde est de : 682.7704287382327
-----
Votre portefeuille :
Vous possédez 0 actions de la societe amazon
Vous possédez 2 actions de la societe apple
Vous possédez 0 actions de la societe samsung
Vous possédez 0 actions de la societe peugeot
-----
Que voulez vous ?
taper 'marche' pour lire l'etat actuel du marche
taper 'vente' pour vendre des actions
taper 'achat' pour acheter des actions
taper 'resultat' pour connaitre l'etat de vos commandes
taper 'fermer' pour fermer la journee

betty@BettyBismuth: ~/Bureau/SAR
commande [prix=20.0, choix=0, nbActions=2, nomSociete=apple, nomClient=12, nomCourtier=1, etat=en cours]
Une commande est en cours d'envoi a la bourse
comOK
Commande [prix=20.0, choix=0, nbActions=2, nomSociete=apple, nomClient=12, nomCourtier=1, etat=effectuee]
votre solde est de : 4.0
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse

betty@BettyBismuth: ~/Bureau/SAR
Que voulez vous ?
taper 'marche' pour lire l'etat actuel du marche
taper 'vente' pour vendre des actions
taper 'achat' pour acheter des actions
taper 'resultat' pour connaitre l'etat de vos commandes
taper 'fermer' pour fermer la journee
marche
-----
Etat du marche :
SOCIETE : apple      PRIX : 20.0    ACTIONS FLOTTANTES :10
SOCIETE : amazon     PRIX : 10.0    ACTIONS FLOTTANTES :800
SOCIETE : samsung     PRIX : 40.0    ACTIONS FLOTTANTES :100
SOCIETE : peugeot     PRIX : 60.0    ACTIONS FLOTTANTES :80
-----
Que voulez vous ?
taper 'marche' pour lire l'etat actuel du marche
taper 'vente' pour vendre des actions
taper 'achat' pour acheter des actions
taper 'resultat' pour connaitre l'etat de vos commandes
taper 'fermer' pour fermer la journee
```



## Affichage de la fermeture de la journée

```
Terminal
betty@BettyBismuth: ~/Bureau/SAR
Comme [prix=20.0, choix=0, nbActions=2, nomSociete=apple, nomClien
t=12, nomCourtier=1, etat=effectuee]
la transaction a ete effectuee et on doit maintenant informer le clien
t
Aucune commande dans le tampon de la bourse
Un client du courtier 1 a ferme sa journee
[1=1]
Aucune commande dans le tampon de la bourse
Un client du courtier 1 a ferme sa journee
[1=0]
SOCIETE : apple    PRIX : 24.0    ACTIONS FLOTTANTES :8
SOCIETE : amazon  PRIX : 10.0    ACTIONS FLOTTANTES :800
SOCIETE : samsung PRIX : 40.0    ACTIONS FLOTTANTES :100
SOCIETE : peugeot PRIX : 60.0    ACTIONS FLOTTANTES :80
Journee terminee, la bourse est fermee.
-----
Aucune commande dans le tampon de la bourse
-----
Tapez connexion pour lancer une nouvelle journee

betty@BettyBismuth: ~/Bureau/SAR
resultat
-----
Votre commande pour 2 actions de la societe apple au prix de 20.0 a
abouti
-----
Votre solde est de : 682.7704287382327
-----
Votre portefeuille :
Vous possédez 0 actions de la societe amazon
Vous possédez 2 actions de la societe apple
Vous possédez 0 actions de la societe samsung
Vous possédez 0 actions de la societe peugeot
-----
Que voulez vous ?
taper 'marche' pour lire l'etat actuel du marche
taper 'vente' pour vendre des actions
taper 'achat' pour acheter des actions
taper 'resultat' pour connaître l'etat de vos commandes
taper 'fermer' pour fermer la journee
fermer

betty@BettyBismuth: ~/Bureau/SAR
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Le client 14 a ferme sa journee.
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Aucune commande à envoyer à la bourse
Le client 12 a ferme sa journee.
Plus de client, courtier ferme

betty@BettyBismuth: ~/Bureau/SAR
taper 'marche' pour lire l'etat actuel du marche
taper 'vente' pour vendre des actions
taper 'achat' pour acheter des actions
taper 'resultat' pour connaître l'etat de vos commandes
taper 'fermer' pour fermer la journee
marche
-----
Etat du marche :
SOCIETE : apple    PRIX : 20.0    ACTIONS FLOTTANTES :10
SOCIETE : amazon  PRIX : 10.0    ACTIONS FLOTTANTES :800
SOCIETE : samsung PRIX : 40.0    ACTIONS FLOTTANTES :100
SOCIETE : peugeot PRIX : 60.0    ACTIONS FLOTTANTES :80
-----
Que voulez vous ?
taper 'marche' pour lire l'etat actuel du marche
taper 'vente' pour vendre des actions
taper 'achat' pour acheter des actions
taper 'resultat' pour connaître l'etat de vos commandes
taper 'fermer' pour fermer la journee
fermer
```

## Affichage de la réouverture

[illegible]

