Intelligence Artificielle (I-INFO-026)

Travaux Pratiques

TP 2 : Programmation d'agents pour la vente de pièces automobiles

- **Introduction :**

L'objectif de ce TP est de programmer des agents intelligents pour la vente de pièces automobiles. Il faudra doter l'agent **'Courtier'** d'une intelligence qui lui rend capable de chercher la meilleure offre parmi deux offres proposées par deux autres agents de type ***'Vendeur'***. Notons que les offres sont formulées en fonction des demandes de l'agent 'Client'. Le but serait donc que l'agent Client dispose d'une interface graphique simple (voir Figure 1) qui lui permet d'encoder le nom d'un courtier (agent intermédiaire), l'identifiant de la pièce auto demandée (exemple : 1 pour les plaquettes,2 pour les pièces de suspensions, etc.), le nombre de pièces désiré et un ***TextArea*** pour l'affichage des interactions entre agents (déroulement d'une transaction).

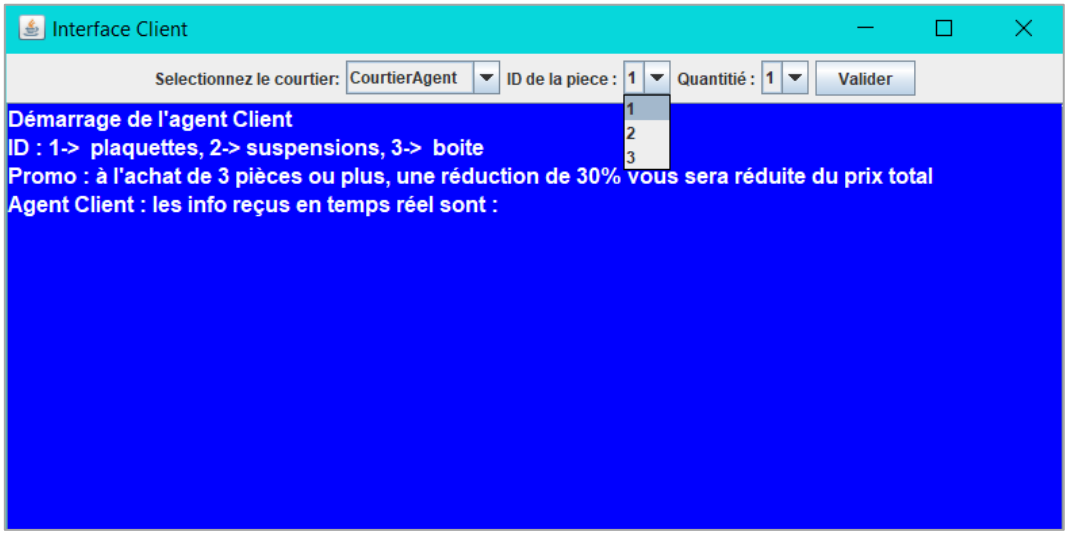

Figure 1 : Interface graphique de l'agent « Client »

Chaque Vendeur disposera aussi d'une simple interface graphique permettant d'afficher toutes les interactions reçues et envoyées. Chacun propose un service en fonction de la demande de l'agent courtier et dispose d'un nombre défini de pièces. Si le nombre demandé par le courtier (pour satisfaire la demande du client) est supérieure à au nombre de pièces disponible, la demande doit être refusée. L'agent vendeur doit répondre via un message de type CFP à l'agent courtier en donnant une réponse indiquant la disponibilité ou pas de la pièce avec le prix.

Afin de réaliser les différentes interactions, il faudra utiliser plusieurs actes de communication. Voilà les actes de communication les plus importants :

- ***INFROM*** : c'est le type de message le plus utilisé, nous pouvons l'utiliser dans plusieurs contextes comme (envoyer une information, confirmation ou un accusé de réception) ;
- ***REQUEST*** : c'est un type de message qui permet de faire une demande initiale. L'agent qui envoie ce message doit recevoir une notification de type ***INFORM*** par exemple ;
- ***CFP*** : c'est l'abréviation de (*Call for Proposal*) ce type peut être utilisé pour les messages que les agents envoient pour demander une offre. Ce message doit recevoir comme réponse un message de type ***PROPOSE ;***
- ***PROPOSE :*** comme son nom l'indique, c'est un message qui doit avoir comme contenu une proposition (soit un prix, un texte qui contient le nom d'un article, etc.). Ce message doit recevoir comme réponse soit (***ACCEPT_PROPOSAL*** ou ***REFUSE***) ;
- ***ACCEPT_ PROPOSAL :*** pour accepter l'offre et terminer l'interaction ;
- ***REFUSE :*** pour refuser l'offre, l'agent qui reçoit ce type de message aura le droit de proposer une autre offre (***PROPOSE***). Ce type de message peut être remplacé par ***REJECT_PROPOSAL*** au cas où l'agent ne veut pas recevoir une nouvelle offre.

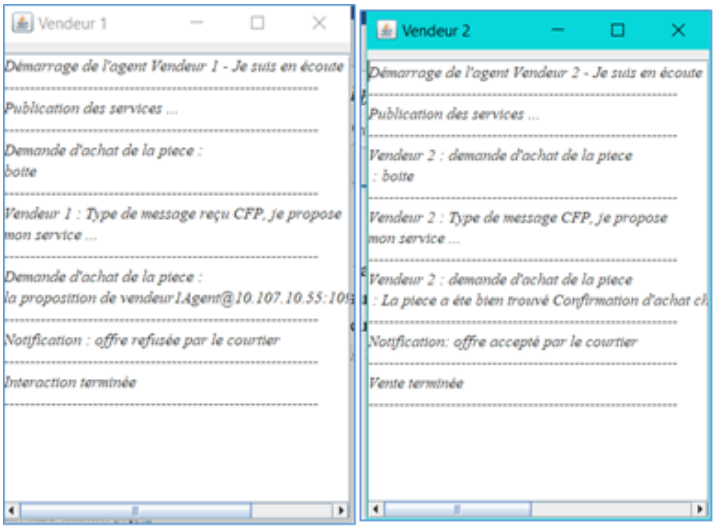

Figure 2 : Interfaces graphiques des deux agents 'vendeur'

- **Exemples :**

- Spécification d'un type de message ACL (de type ***REQUEST***) :

```
ACLMessage aclMsg=new ACLMessage(ACLMessage.REQUEST) ;
```

- Vérification si un message reçu s'agit d'un **CFP** (call for Proposal)

```
if(msg.getPerformative() == ACLMessage.CFP);
```

- **Enoncé du TP :**

- **Question 1 :** Créer le conteneur principal « MainContainer »
- **Question 2 :** Créer trois autres conteneurs : Clients, Vendeurs et Courtiers dans lesquels il faudra inclure un agent client (dans le conteneur Clients), deux agents vendeur (dans le conteneur Vendeurs) et un agent courtier (dans le conteneur Courtiers)
- **Question 3 :** l'agent client doit envoyer une demande à l'agent courtier pour acheter une pièce de voiture, en encoudant l'identifiant et nombre de pièces souhaitées. Lors du clic sur le bouton **« envoyer »,** la fonction **onGuiEvent(GuiEvent ev)** doit s'exécuter dans un comportement **OneShotBehaviour** dans la classe de l'agent Client. Dans la classe « ClientAgent.java », implémenter le transfert du message collecté depuis l'interface graphique vers l'agent courtier ?
- **Question 4 :** une fois que la demande du client est reçue par le courtier, ce dernier doit transférer la demande (de type CFP) **aux deux agents vendeurs** (créés précédemment). Dans la classe « CourtierAgent », implémenter cet envoi ?
- **Question 5 :** chaque agent vendeur reçoit le message du courtier (**CFP**) et proposera en retour son offre si le stock est suffisant avec le prix total de la commande. Dans les classes « Vendeur1Agent.java » et « Vendeur2Agent.java », implémenter :
    - L'envoi d'une réponse au courtier selon le stock disponible chez chaque vendeur ;
    - Si le nombre de pièces > 3, une réduction de 30% est appliqué au prix total.
- **Question 6 :** l'agent courtier doit sélectionner l'offre (parmi les deux offres venant des agents vendeurs) selon la demande du client et le prix total proposé avant d'envoyer un message de type ***ACCEPT_ PROPOSAL*** au bon vendeur et **REFUSE** au vendeur qui a proposé la mauvaise offre :
    - Dans la classe ***CourtierAgent***, implémenter l'envoi des deux messages ;
    - Dans les classes « Vendeur1Agent.java » et « Vendeur2Agent.java », implémenter la confirmation (ou non) de la vente en affichant un message sur l'interface graphique.
- **Question 7 :** Après avoir choisi bon vendeur, le courtier doit confirmer la transaction d'achat de la pièce au client. Implémenter l'envoi du message au client pour confirmer l'achat.

- **Remarque 1 :** fin d'interactions : (il faut utiliser le comportement ***Behaviour*** qui donne droit à la modification de la fonction *done*() pour arrêter le comportement).
- **Remarque 2 :** Vous pouvez utiliser l'outil « Sinffer » qui permet de suivre les interactions entre les agents (Figure 3).

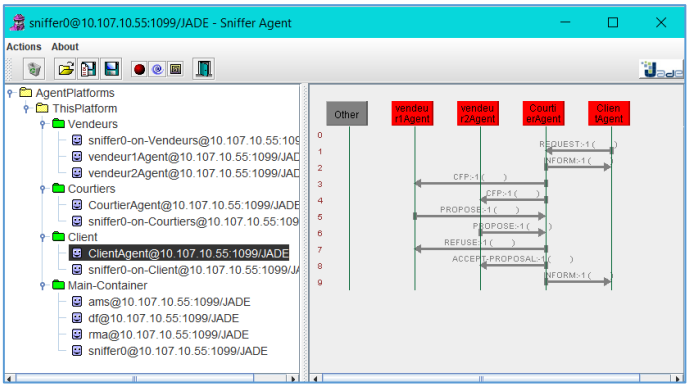

Figure 3 : l'outils Sniffer permettant de suivre les interactions des agents (à voir dans le tp)

- **Questions facultatives :**

  - **Question 8 :** après confirmation de chaque commande, le courtier doit enregistrer le nom du vendeur avec le nom de pièce. Si le client veut racheter la même pièce, le courtier doit réagir intelligemment et contacter directement le vendeur en question sans demander aux autres.

  - **Question 9 :** ajouter un 3ème agent « *vendeur3Agent* », qui propose la même pièce que le « vendeur2Agent », mais avec un prix plus élevé. Ce troisième agent propose une promotion de 50% lors de l'achat de plus de 3 pièces au lieu de 30%. Implémenter cela de telle manière que le client puisse toujours avoir le prix le plus bas.

  - **Question 10 :** augmenter l'intelligence du courtier afin qu'il puisse prendre en compte en plus du prix d'autres critères tels que, la qualité (marque), les avis d'utilisateurs, etc. Implémenter la prise en compte de ces critères pour le choix du courtier.

  - **Question 11 :** Implémenter la publication des services de chaque vendeur en utilisant la librairie JADE *DFService.* L'agent courtier commencera par consulter et comparer entre tous les services publiés par les différents vendeurs. Ensuite il sélectionnera le vendeur qui a publié le meilleur service correspondant à la commande en donnant aussi (Figure 4)

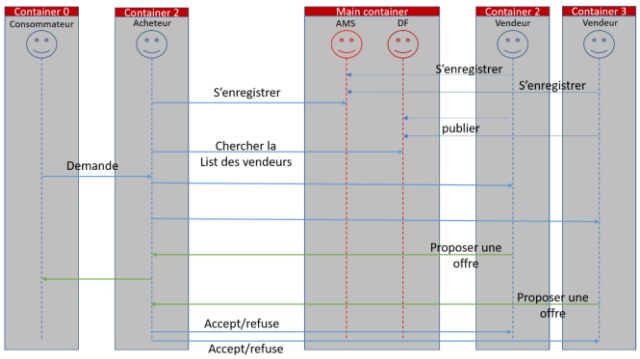

Figure 4 : Figure diagramme d'interaction : questions supplémentaires