Fondements des Systèmes Multi-Agents

Alexandre Le Borgne – M2 TI

1. Organisation

- src/fr/univpau/sma/projet/
  - agent/
    - DealerAgent.java
      Est la classe d'implémentation de l'agent vendeur.
    - MarketAgent.java
      Est la classe d'implémentation de l'agent marché.
    - TakerAgent.java
      Est la classe d'implémentation de l'agent preneur.
    - behaviour/
      - dealer/
        - RegisterAtMarket.java
          Cette classe est la première des classes de behaviour de l'agent Dealer. Elle contient le processus d'inscription du dealer au Market en ayant préalablement cherché via les yellow pages un service « *black-market* ».
        - fsm/
          - DealerFSMBehaviour.java
            C'est l'implémentation par FSM de l'automate du vendeur vu en cours. Cette classe contient des classes privées destinées à définir les états de l'automate :
            1. WaitForTakers
               Cette classe a pour rôle d'attendre qu'au moins un prenneur potentiel soit inscrit à l'enchère pour démarrer la vente.
            2. Announce
               Cette classe permet au vendeur de faire une annonce.
            3. WaitForBids
               Cette classe attend et compte les bids.
            4. Attribute
               Permet d'attribuer un lot.
            5. Give
               Permet de donner le lot.
            6. Withdraw
               Permet au vendeur d'abandonner la vente si le prix est trop bas.
            7. WaitForNewTaker
               Permet au vendeur d'inscrire de nouveaux preneurs en cours d'enchère.
            8. End

Termine l'enchère.

- market/
  - MarketCyclicBehaviour.java

    Permet au marché de gérer tous les messages qu'il reçoit.
  - SpreadAucionsBehaviour.java

    Permet de faire parvenir aux agents preneurs les enchères quand un nouveau vendeur s'inscrit sur le marché.
- taker/
  - RegisterAtMarket.java

    Cette classe est la première des classes de behaviour de l'agent Taker. Elle contient le processus d'inscription du dealer au Market en ayant préalablement cherché via les yellow pages un service « *black-market* ».
  - WaitForAuction.java

    Cette classe est un behaviour threadé qui écoute les nouvelles enchères arrivant du marché et s'y inscrit. Dans le mode automatique, l'agent preneur s'inscrit à toutes les enchères, mais dans le mode manuel c'est sa classe privée WaitForUserSelection qui a la charge d'écouter les choix de l'uttilisateur.
    Chaque enchère choisie est traitée dans une nouvelle machine à états afin de gérer de multiples inscriptions.
  - fsm/
    - TakerFSMBehaviour.java

      C'est l'implémentation par FSM de l'automate du preneur vu en cours. Cette classe contient des classes privées destinées à définir les états de l'automate :
      1. WaitForAnnounce

         Attend une annonce d'un vendeur. Permet aussi de gérer les message de type « to*Attribute* » et « *toWithdraw* ».
      2. Bid

         En mode automatique, permet au preneur d'envoyer un bid sur une enchère.
      3. WaitForUserBid

         En mode manuel, permet d'attendre et d'envoyer les bids de l'utilisateur.
      4. WaitForGive

         Permet d'attendre la livraison du produit lié à l'enchère.

5. Pay

   Permet au preneur d'envoyer la notification de paiment. Termine l'automate.

6. Lose

   Termine l'automate quand le preneur a perdu l'enchère. Quand le vendeur annule une enchère, le preneur considère qu'il a perdu l'enchère

- gui/
  - dealer/
    - ➔ DealerAuctionCreation.java

      Est l'écran permettant de définir le nom du vendeur et les paramètres de l'enchère.
    - ➔ DealerGUI.java

      Est l'écran d'affichage de l'activité du vendeur.
  - market/
    - ➔ MarketGUI.java

      Est l'écran d'affichage de l'activité du marché.
  - taker/
    - ➔ TakerGUI.java

      Est l'écran d'affichage et de contrôle en fonction du mode choisi de l'activité du vendeur.
    - ➔ TakerModeChoice.java

      Est l'écran permettant de définir les paramètres du preneur.
- objects/
  - Auction.java

    Est l'objet permettant de stocker les enchères.
  - DealerTakerTable.java

    Est le modèle de la table des bidders utilisée par l'interface du vendeur.
  - MarketCurrentAuctionsTable.java

    Est le modèle de la table des enchères en cours utilisée par l'interface du marché.
  - MarketPastAuctionsTable.java

    Est le modèle de la table des enchères terminées utilisée par l'interface du marché.
  - ProtocolMessage.java

    Est la classe qui étend ACLMessage et permet une communication plus aisée entre les différents acteurs du système.
  - TakerAuctionSelectionTable.java

    Est le modèle de la table de sélection des enchères utilisée

par l'interface du preneur en mode manuel.

- TakerCurrentAuctionsTable.java

  Est le modèle de la table des enchères en cours utilisée par l'interface du preneur.

- TaketPastAuctions.java

  Est le modèle de la table des enchères terminées utilisée par l'interface du preneur.

2. Execution

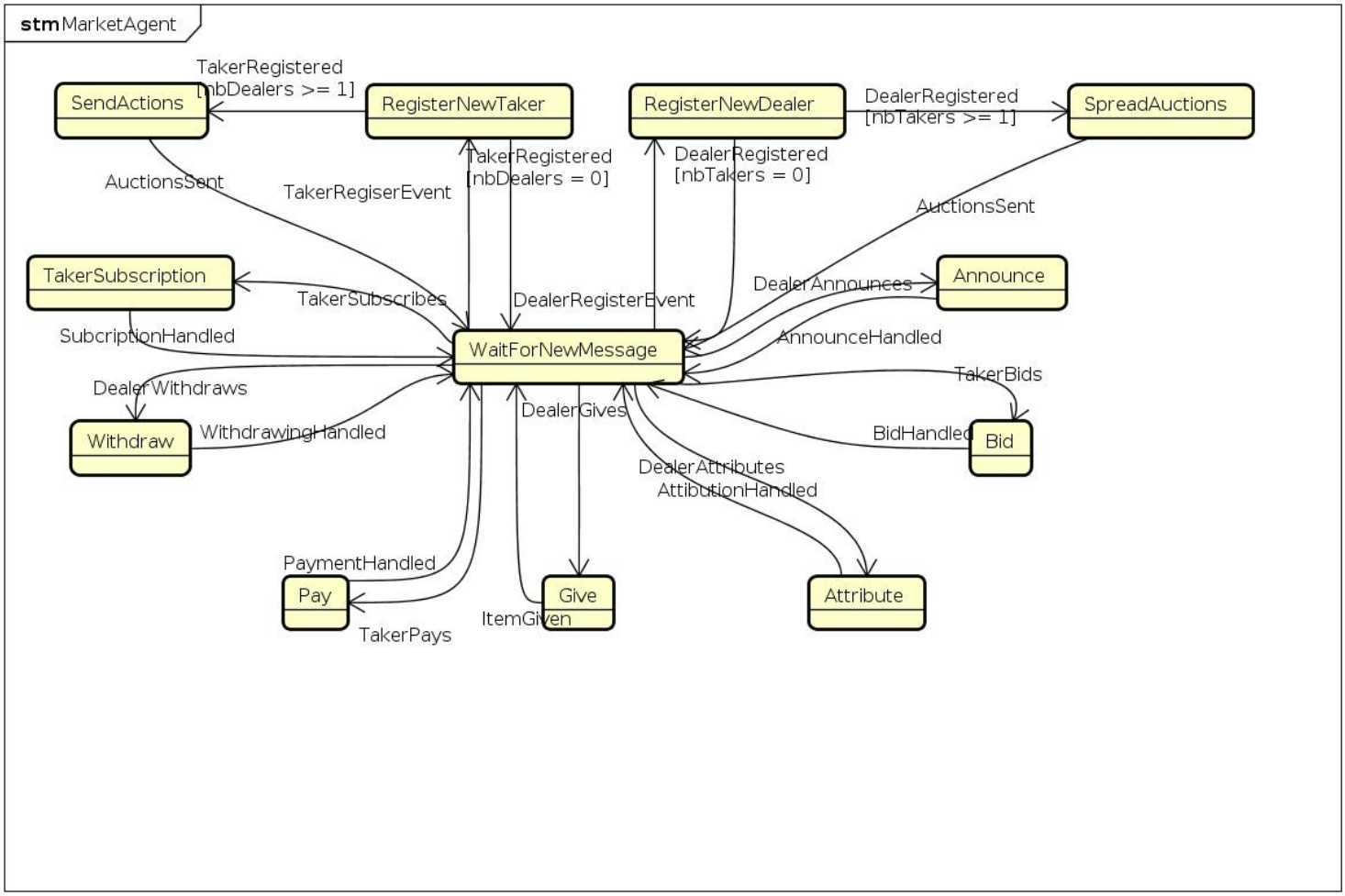
Afin de faciliter la compilation et l'exécution du projet, deux scripts ont été créés : compile.sh et launch.sh.
Il est à noter que lors du lancement du programme, seule l'interface du marché apparaît. C'est à ce stade le seul agent présent sur la plate-forme jade.
La création agent vendeurs et preneurs se fait dynamiquement via le menu de l'interface market ou grâce à des raccourcis clavier :

- Ctrl + T : Nouveau preneur automatique
- Ctrl + Shift + T : Nouveau preneur manuel
- Ctrl + Shift + D : Nouveau vendeur

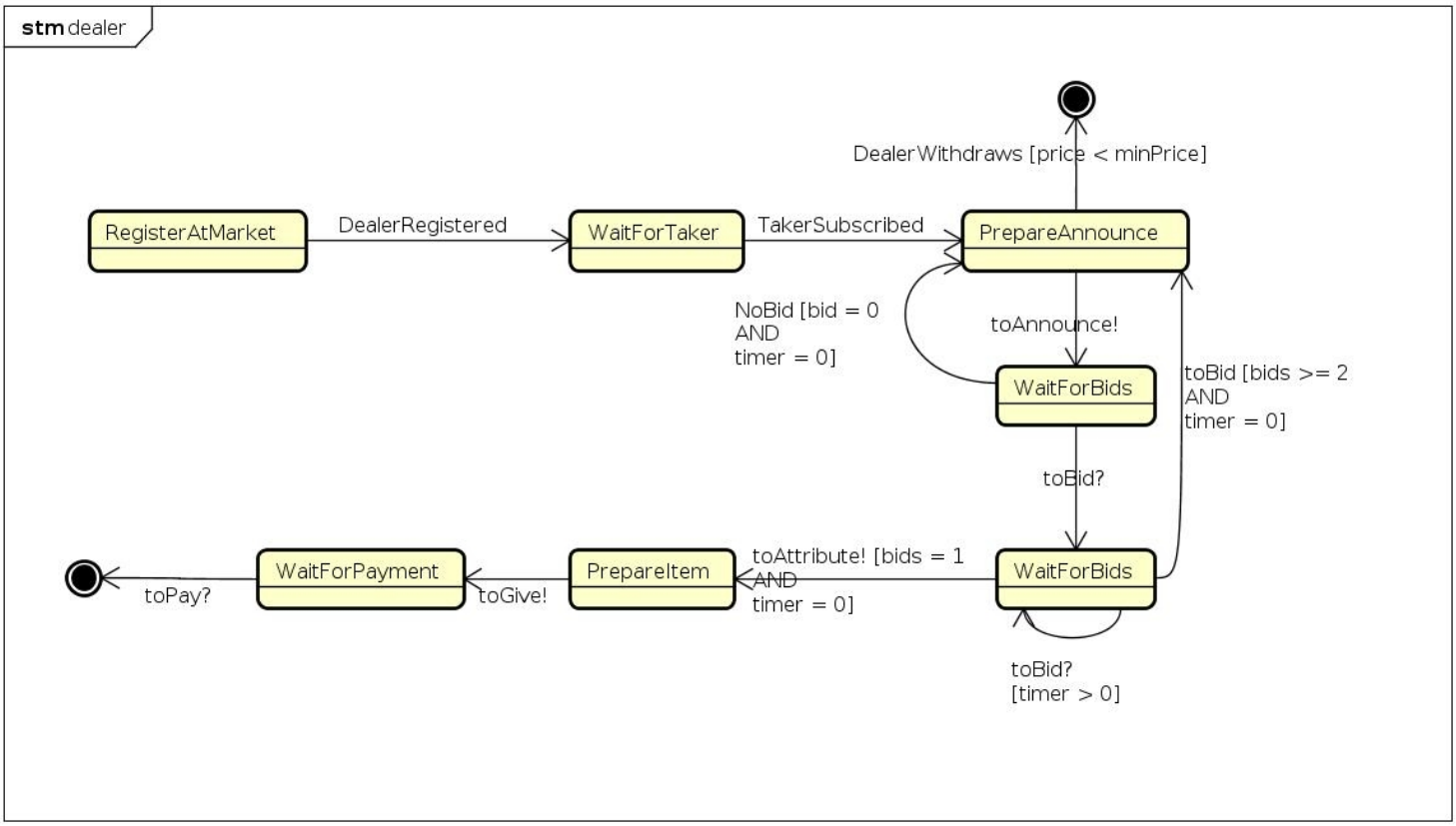
Le programme est capable de gérer de multiples vendeurs ayant de multiples preneurs (automatiques ou non) inscrits.

3. Automates

- Agent Market



- Agent Dealer



- Agent Taker