Jérémie Bosom

Léo DELLOUVE

Luc Poncet

Rendu Première Partie - SMA

Un Réseau Simple

version 1

2014-2015

Table des matières

Présentation du travail

Lancement et exécution du programme

Pour lancer le projet nous vous invitons à lancer l'exécutable à l'aide de la commande :
java -jar test.jar

Lors du lancement du programme,

- un producteur est crée,
- un ensemble de consommateur aléatoire variant de 1 à 10 est créé. Certains d'eux sont des consommateurs producteurs. Leur consommation et leur production respective sont fixées à 10 pour l'instant mais nous la ferons varier dans les rendus futurs.
- un observateur est crée.

Agents présents

Afin d'éviter la duplicité de notre code nous avons réalisé une classe AbstractAgent dont héritent tous nos autres agents. Elle enregistre les agents au DF et permet d'appliquer les GlobalBehaviour, comportements communs à certains agents.

Nous avons de plus réalisé les 3 agents demandés, à savoir :

- ConsommateurAgent, l'agent consommateur. Son but est de s'abonner au producteur qui lui propose le tarif le moins cher. Pour cela il envoi toutes les secondes une demande de prix aux producteurs après avoir consulté le DL pour récupérer leurs noms. Il sélectionne ensuite celui qui lui propose le tarif le plus intéressant et s'y abonne. Il peut, si l'occasion se présente, se désabonner de son producteur initial pour s'abonner à un autre producteur. De plus, lorsque tous les mois le producteur lui demande de payer sa facture il envoie l'argent au producteur.
- ProducteurAgent, l'agent producteur. Il envoie toutes les secondes la demande de paiement à ses clients, et répond à leurs demandes d'abonnement, de désabonnement et de prix. Ses clients sont stockés dans une liste.
- ObservateurAgent, l'agent observateur consulte toutes les secondes les fournisseurs qu'il a référencé via le DF pour connaître leurs bénéfices et prix respectifs. Il affiche continuellement via une fenêtre JFrame les informations du marchés.

Le contenu des messages est différent pour chaque type de requêtes.

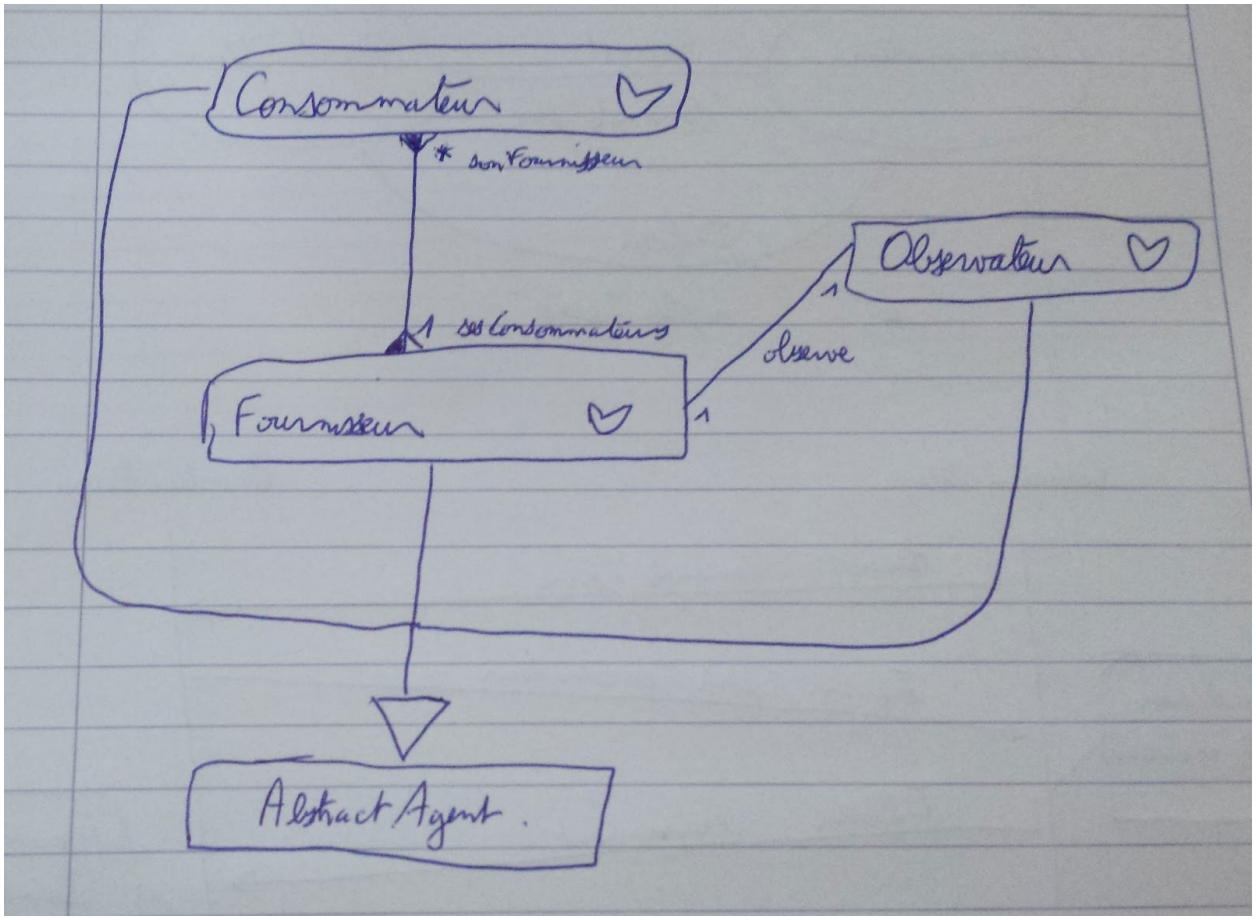
Comportements

Nous leur avons associé les comportements (Behaviour) suivants :

- Globaux
    - Nous avons créé un comportement global permettant de récupérer les AID de tous les services correspondant à une certaine description. Pour utiliser ce Behaviour un service doit hériter de la classe AbstractAgent.
- Consommateur
    - Le comportement MessageListenner permet de traiter les différents messages reçus par le consommateur et d'y répondre. C'est ici que les prix des fournisseurs sont reçus et que le consommateur décide de s'abonner ou non. Il s'agit d'un CyclicBehaviour.
    - Le comportement AskPrixFournisseur demande le prix de tous les fournisseurs susceptibles d'alimenter le consommateur en leur envoyant un message à chacun. Il travail sur les AID récupérées via le comportement global. Il s'agit d'un TickerBehaviour avec une période de 1 seconde dans notre simulation.
- Producteur
    - Le comportement FactureClient qui demande aux abonnés d'un fournisseur de payer leur créances. Il s'agit d'un TickerBehaviour avec une période de 1 seconde dans notre simulation.
- Observateur
    - Le comportement AskPrixFournisseur agit globalement de la même manière que celui du consommateur mais avec une période de 1 secondes afin d'avoir toujours les bonnes valeurs.

L'interface graphique Swing (Java) vous permettra de visualiser les champs demandés, à savoir les bénéfices réalisés par le producteur, son tarif et le nombre de client.

Protocoles AUML - les agents



AUML - Les agents présents

Schémas de comportement

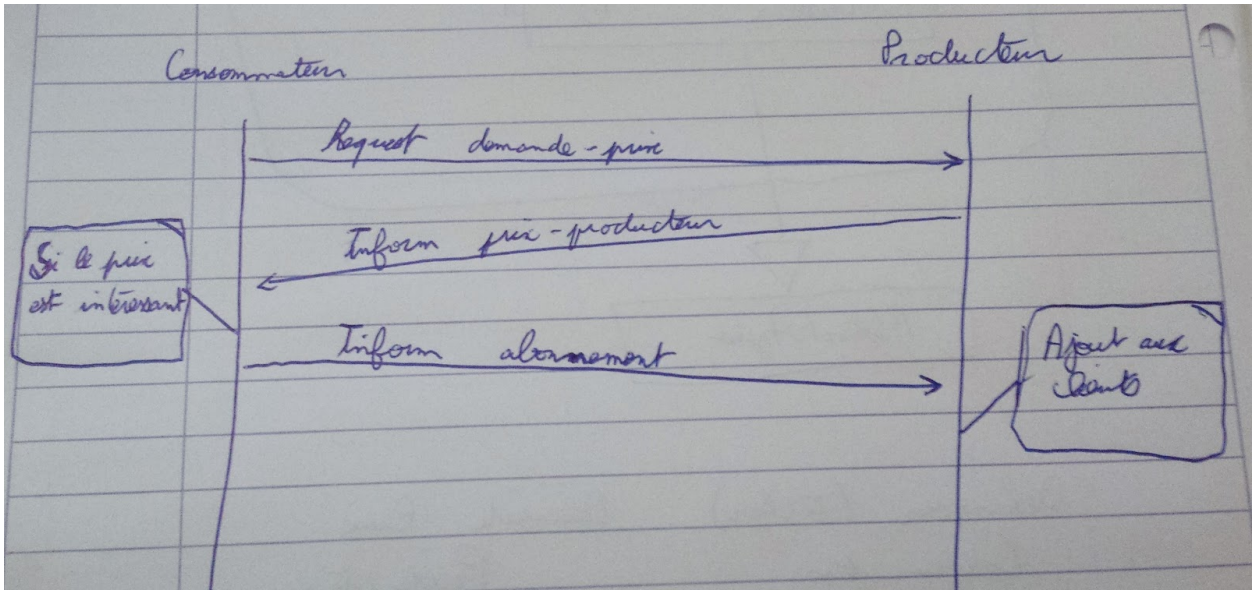


Diagramme de comportement - demande de prix acceptée par le client

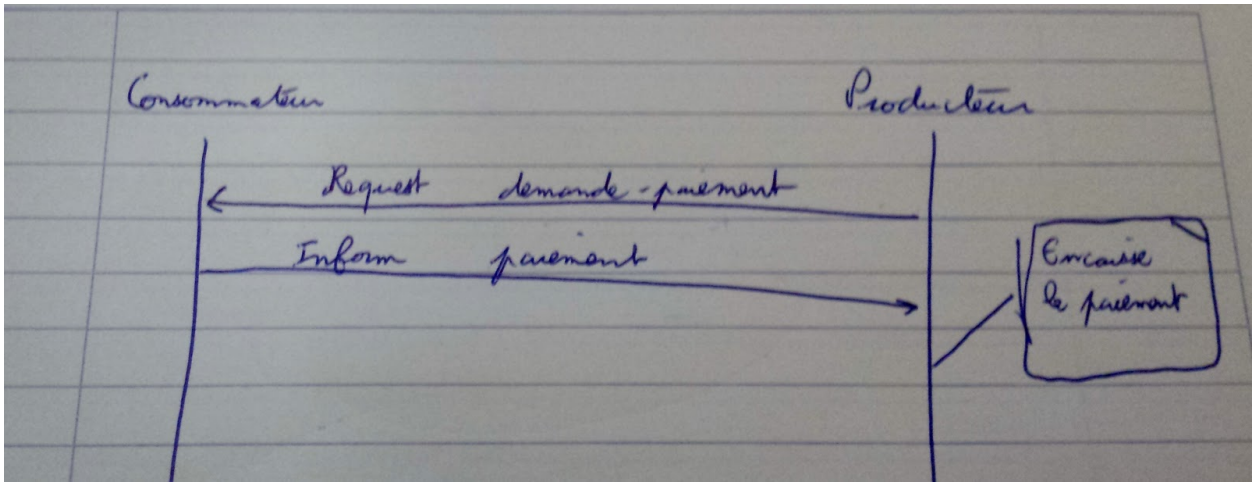


Diagramme de comportement - demande paiement