Master IHM

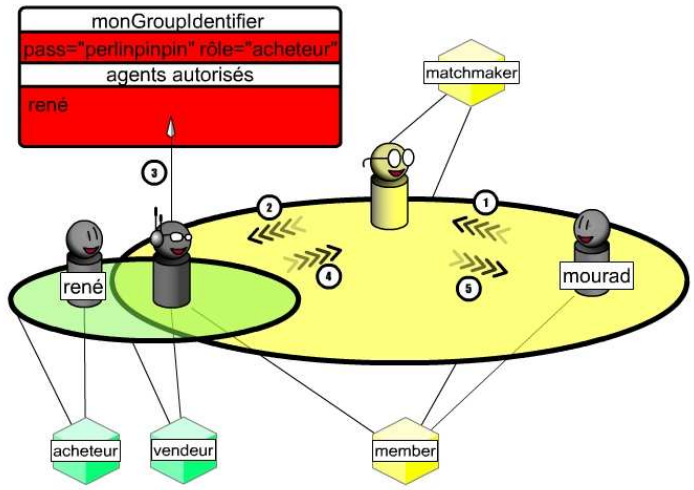
Projet Protocole d'Enchères
Systèmes Multi-Agents avec MadKit

Objectifs :
a) spécification et création d'une organisation multi-agents dans MadKit
b) implémentation d'un matchmaker et d'un protocole d'enchères
c) créer une IHM en adéquation avec les besoins d'un utilisateur du système

Présentation du problème
On souhaite réaliser un système multi-agents simulant "la vente à la criée". Ce système est organisé autour deux types de groupes et de trois rôles. Le groupe de type *marché* accueille tous les agents, et un groupe de type « *salle de vente* » accueille un vendeur et les acheteurs d'un type de poisson donné. Il peut exister autant de salles de ventes que de types de poissons. Le rôle du matchmaker est de mettre en relation des acheteurs et des vendeurs d'un même type de poisson. Dès qu'une salle contient un vendeur et au moins deux acheteurs, les enchères peuvent débuter **dans cette salle** selon le protocole suivant :

- le vendeur propose un prix
- si un et un seul acheteur fait part de son intérêt pour l'enchère proposée alors le poisson lui est attribué
- si plus d'un acheteur sont intéressés alors le vendeur augmente son prix.
- si aucun acheteur n'est intéressé alors le vendeur baisse son prix.

Au départ de la simulation, **le matchmaker** doit être lancé et il crée le marché qu'on appellera "FishMarket" situé dans la communauté **"DESS IHM".** Ensuite, on peut introduire les acheteurs et les vendeurs.



Lorsqu **vendeur** est créé, il détermine aléatoirement un lot de poisson (thon, mérou, sardine) à vendre ainsi qu'un prix de départ, il entre dans le marché, et enfin crée une *salle de ventes où se passeront les enchères*. Ensuite, Il doit se déclarer au près du matchmaker comme vendeur en lui fournissant les informations suivantes : son adresse (AgentAddress), le type de poisson qu'il vend, le nom de la salle des ventes, et la carte d'accès à cette salle. Le matchmaker ne référence qu'un seul vendeur par type de poissons donné.
Lorsqu **acheteur** est créé, il détermine aléatoirement un type de poisson qu'il souhaite acquérir et le prix maximum d'achat, et entre dans le marché. Il cherche ensuite une salle des ventes en s'adressant au matchmaker et en lui indiquant le poisson recherché. Si une telle salle des ventes existe, le matchmaker prévient le vendeur de la salle afin que celui ci permette l'accès à l'acheteur. Si le vendeur accepte, il en informe le matchmaker qui fournit la carte d'accès à l'acheteur(cf. dessin). L'acheteur peut alors entrer dans la salle des ventes pour assister aux enchères.

Démarche
1) Lire l'énoncé
2) Assurer vous d'avoir compris l'énoncé
3) Spécification sur papier de l'organisation (sans les enchères)
    a. Spécifier le matchmaker (attributs et principes algorithmiques de l'activation, la vie et la mort)
    b. Spécifier le vendeur

c. Spécifier l'acheteur

4) Correction collective de 3)
5) Implémentation de 4 dans l'ordre matchmaker, vendeur, acheteur
6) Implémentation des enchères.

Indications

- **Communication**
  - Les agents communiquent par des messages typés **ActMessage(performatif, contenu)**
  - Le *performatif* est une *String*, le *contenu* peut être une *String* ou un *Object*
  - On utilisera le **même protocole d'interaction** pour communiquer :
    - *Vendeur*
      - ActMessage**("informer gagnant", null)** : informer le destinataire qu'il a remporté l'enchère.
      - ActMessage**("informer fin", AgentAddress gagnant)** : informer le destinataire que l'enchère a été remporté par le *gagnant*.
      - ActMessage**("informer suicide",null)** : informer le matchmaker de son suicide
      - ActMessage**("demander publier", InfoCard)** : demander au matchmaker de publier les informations de l'*InfoCard*
      - ActMessage**("proposer", String prix)** :proposer un prix d'enchère
      - ActMessage**("AR demander entrer groupe", AgentAddress acheteur**) : accuser réception d'un message dont le performatif est "demander entrer groupe"(à valeur de réponse positive).
    - *Acheteur*
      - ActMessage**("demander existence salle", String typeDePoisson)** : demander au matchmaker si il existe une salle où l'on vend un certain type de poisson
      - ActMessage**("accepter proposition",null)** : accepter la dernière offre émise par le vendeur
    - *Matchmaker*
      - ActMessage**("répondre publier", String reponse='oui'/'non')** : répondre à une demande de publication.
      - ActMessage**("demander entrer groupe", AgentAddress demandeur)** : demander au vendeur destinataire si il peut laisser entrer l'agent demandeur dans son groupe
      - ActMessage**("répondre existence salle", InfoCard infoCard)** : réponse à une demande d'existence de salle et donner une carte d'entrée.

- **Gestion des cartes d'informations**
  - Le *matchmaker* doit gérer les informations fournies par les vendeurs.
  - Le matchmaker utilisera pour ceci un objet de type InfoCard(AgentAddress vendeur, String placeDeVente,String typePoisson, String accessCard) qui vous est fourni.
  - Que ce passe-t-il lorsqu'un vendeur meurt ?

- **Gestion sécurisée de l'entrée dans les salles de ventes :**
  - les vendeurs utiliseront un objet de type **MonGroupFilter** . Initialisez l'objet comme suit : *new MonGroupFilter(String accessCard, String role)* où *accessCard* est le password pour entrer dans le groupe et *role* est le type de rôle que l'on désire laissé entré : ici un "*acheteur".*

- Lorsqu'un vendeur créer un groupe (*createGroup()*) passer lui l'objet MonGroupFilter en dernier paramètre. Le noyau MadKit vérifiera automatiquement si un agent à le droit d'entrée dans ce groupe en se référant à cet objet.
- Pour autoriser un *acheteur* à entrer dans votre salle de vente utiliser la méthode de MonGroupFilter *allowAgentToTakeRole(AgentAddress requester, String roleName, Object memberCard).*
- Pour plus d'infos regarder la documentation de la class *madkit.kernel.GroupIdentifier.*

Conseils d'implémentation

Les soucis que vous risquez de rencontrer, outre les bugs classiques, vont certainement être au niveau de l'interaction donc n'hésitez pas à jouer du *println* en numérotant les étapes.

IHM

L'interface que l'on vous demande de créer doit au moins permettre :

-pour l'acheteur : la sélection des types de poissons et le prix maximum d'achat

-pour le vendeur : la sélection des types de poissons et le prix de départ

Le suivi des interactions est également un des points indispensables de l'interface : l'utilisateur doit pouvoir savoir à **tout moment** ce que font ses agents.

Pour le reste vous êtes libre de mettre à profit vos talent de créateur d'interfaces.

Conditions de réalisation de ce long TP

Ce long TP est un travail en binôme et donne lieu à un dossier (10 pages maximum sans le programme source) et à une démonstration (1/4 d'heure). La présentation et la remise des dossiers sont prévues la dernière semaine de Février à une date et à des horaires qui seront définis plus tard.

Contenu du dossier

1. Une description de votre application (services offerts, interface, diagramme de classe, …on pourra s'aider de la notation UML)
3. Description des tests effectués pour s'assurer que le programme est correct. On indiquera également les cas non traités ainsi que les améliorations possibles.
4. Un manuel utilisateur, destiné à un utilisateur n'ayant pas participé au projet.
5. Les sources commentés.
6. Un CD commun à tous les groupes contenant les sources.

Une grande attention sera apportée à la clarté de la présentation : rédaction claire et concise, page de garde, pagination, sommaire.