**TP La communication entre agents JADE**

Pour que plusieurs agents JADE arrivent à collaborer, ils doivent s'échanger des messages. Chaque agent JADE possède une sorte de boite aux lettres qui contient les messages qui lui sont envoyés par les autres agents. Ces boites aux lettres sont sous forme d'une liste qui contient les messages selon l'ordre chronologique de leur arrivée.

**II. Format d'un message JADE**

Les agents JADE utilisent des messages conformes aux spécifications de la FIPA ([FIPA-ACL](http://www.fipa.org/specs/fipa00061/SC00061G.html) ). Les messages JADE sont des instances de la classe ACLMessage du package jade.lang.acl. Ces messages sont composés en général de :

* L'émetteur du message : un champ rempli automatiquement lors de l'envoi d'un message.
* L'ensemble des récepteurs du message : un message peut être envoyé à plusieurs agents simultanément.
* L'acte de communication : qui représente le but de l'envoi du message en cours (informer l'agent récepteur, appel d'offre, réponse à une requête,…)
* Le contenu du message.
* Un ensemble de champs facultatifs, comme la langue utilisée, l'ontologie, le timeOut, l'adresse de réponse…

**III. L'envoi d'un message**

Pour envoyer un message, il suffit de remplir les champs nécessaires (l'ensemble des récepteur et le contenu du message et l'acte de communication) d'un message JADE, puis d'appeler la méthode send() de la classe Agent. Voici un exemple d'envoi d'un message JADE.

**ACLMessage message = new ACLMessage(ACLMessage.INFORM);**

**message.addReceiver(new AID("said", AID.ISLOCALNAME));**

**message.setContent("bonjour…");**

**send(message);**

Ce message est envoyé à l'agent appelé Said pour lui dire bonjour.

**IV. La réception d'un message**

La réception d'un message est aussi simple que l'envoi. Il suffit d'appeler la méthode reveive() de la classe Agent pour récupérer le première message non encore lu de l'agent.   
Exemple : **ACLMessagemessageRecu = receive();**

Pour répondre à un message reçu on récupère l'identité de l'émetteur du message par la méthode getSender(). Voici un exemple :

**ACLMessage message = new ACLMessage(ACLMessage.INFORM);**

**message.addReceiver(messageRecu.getSender());**

**message.setContent("la reponse");**

**send(message);**

**V. L'attente d'un message**

Il se peut qu'un agent doive effectuer un certain traitement ou lancer quelques tâches après avoir reçu un message d'un autre agent. Il est possible de faire une attente active jusqu'à l'arrivé du message de la manière suivante :

**ACLMessage message =null ;**

**While (message == null){**

**Message = receive() ; //traitement à faire après avoir reçu le message.**

Mais ce genre d'attente active consomme énormément les ressources de la machine sur laquelle l'agent s'exécute. On peut aussi bloquer un Behaviour d'un agent jusqu'à la réception du message, et ceci grâce à la méthode block() de la classe Behaviour.

**Message = receive() ;**

**If (message == null) block(); //traitement à faire après avoir reçu le message.**

Le Behaviour se bloque jusqu'à la réception d'un message. A la réception d'un message tous les Behaviour bloqués de l'agent continuent leur exécution.

**VI. Choisir un message de la boite aux lettres**

Lorsqu'un agent communique simultanément avec plusieurs agents, il peut décider un moment de ne lire que les messages de la provenance d'un agent particulier ou les messages ayant un acte de communication particulier. Pour cela un agent peut définir un modèle de message à recevoir.  
Exemple :

**MessageTemplatemodele = MessageTemplate.MatchPerformative(ACLMessage.INFORM); ACLMessagemsg = myAgent.receive(modele);**

**VII. Exemple**

Voici un exemple de communication entre deux agents. Nous avons deux agents : AgentA et AgentB. AgentA produit des nombres aléatoires puis les envoies à AgentB qui cumule ces nombre aléatoires. Tant que la somme ne dépasse pas un certain seuil (50 dans l'exemple), il informe AgentA qu'il est prêt à recevoir d'autres nombres aléatoires. Dés que le seuil est dépassé, AgentB informe AgentA et s'arrête. Dès qu'AgentA reçoit le message d'arrêt, il s'arrête également.

**AgentA.java**

**importjade.core.AID;**

**import jade.core.Agent;**

**import jade.core.behaviours.FSMBehaviour;**

**import jade.core.behaviours.OneShotBehaviour;**

**importjade.lang.acl.ACLMessage;**

**public class AgentA extends Agent {**

**protected void setup(){**

**System.out.println("---------------------------------------");**

**System.out.println("----------------agent A----------------");**

**System.out.println("---------------------------------------");**

**FSMBehaviour agentA\_beh= new FSMBehaviour();**

**agentA\_beh.registerFirstState(new attendreAgentB(), "attendreAgentB");**

**agentA\_beh.registerState(new envoiChiffre(), "envoiChiffre");**

**agentA\_beh.registerLastState(new fin(), "fin");**

**agentA\_beh.registerDefaultTransition("attendreAgentB", "envoiChiffre");**

**agentA\_beh.registerTransition("envoiChiffre", "attendreAgentB",0);**

**agentA\_beh.registerTransition("envoiChiffre", "fin", 1);**

**addBehaviour(agentA\_beh); }**

**private class attendreAgentB extends OneShotBehaviour{**

**@Override**

**public void action() {**

**System.out.println("en attente de l agent B");**

**block(); }}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**private class envoi Chiffre extends OneShotBehaviour{**

**int valeur Retour = 0;**

**@Override**

**public void action() {**

**ACLMessage message Recu = receive();**

**if (messageRecu.getContent().equalsIgnoreCase("pret") )**

**valeur Retour=0;**

**else valeur Retour=1;**

**int chiffre = (int)(Math.random()\*10);**

**System.out.println("envoi de la valeur "+ chiffre);**

**ACLMessage message = new ACLMessage(ACLMessage.INFORM);**

**message.addReceiver(messageRecu.getSender());**

**message.setContent(chiffre+"");**

**send(message); }**

**public intonEnd(){**

**returnvaleurRetour; } }**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**private class fin extends OneShotBehaviour{**

**@Override**

**public voidaction() {**

**System.out.println("arret de l'agent");**

**myAgent.doDelete(); } }}**

**AgentB.java**

**importjava.security.acl.Acl;**

**import jade.core.AID;**

**import jade.core.Agent;**

**import jade.core.behaviours.FSMBehaviour;**

**import jade.core.behaviours.OneShotBehaviour;**

**importjade.lang.acl.ACLMessage;**

**public class AgentB extends Agent {**

**int somme = 0;**

**boolean stop = false;**

**protected void setup(){**

**System.out.println("----------------agent B----------------");**

**FSMBehaviour agentB\_beh= new FSMBehaviour();**

**agentB\_beh.registerFirstState(new attendrechiffre(), "attendrechiffre");**

**agentB\_beh.registerState(new afficher(), "afficher");**

**agentB\_beh.registerState(new fin(), "fin");**

**agentB\_beh.registerTransition("attendrechiffre", "afficher",0);**

**agentB\_beh.registerTransition("attendrechiffre", "fin",1);**

**agentB\_beh.registerDefaultTransition("afficher", "attendrechiffre");**

**addBehaviour(agentB\_beh); }**

**private class attendrechiffre extends OneShotBehaviour{**

**int valeur Retour = 0;**

**public void action() {**

**ACLMessage message = new ACLMessage(ACLMessage.INFORM);**

**message.addReceiver(new AID("AgentA", AID.ISLOCALNAME));**

**if(!stop){**

**message.setContent("pret");**

**send(message);**

**valeur Retour=0;**

**block();}**

**else{**

**message.setContent("arret");**

**send(message);**

**valeur Retour=1; } }**

**public int onEnd(){**

**return valeur Retour;**

**}**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**private class afficher extends OneShotBehaviour{**

**@Override**

**public voidaction() {**

**ACLMessagemessageRecu = receive();**

**somme+= Integer.parseInt(messageRecu.getContent());**

**System.out.println("message recu= "+ messageRecu.getContent());**

**System.out.println("la somme actuelle = "+somme);**

**if (somme > 50) stop = true;} }**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**private class fin extends OneShotBehaviour{**

**@Override**

**public voidaction() {**

**System.out.println("fin de l'agent");**

**myAgent.doDelete(); } }}**