

Université Benyoucef BENKHEDDA-Alger1

Faculté des Sciences

Département Informatique



Spécialité : Master Réseaux et Systèmes embarqués

Mini projet systèmes complexes.

Communication Inter-Agents par la plateforme Jade

Réalisé par :

- LAYADI wissem
- MISSARA Razine Merouane

Professeur : MEZZOUDJ Saliha

Table des matières

Table des matières	I
Table des figures	II
Introduction	1
1 Modèle de communication entre agents	2
1.1 Système multi-agents	2
1.2 Processus de communication proposé	2
2 La plateforme JADE et la réalisation du modèle	5
2.1 Introduction	5
2.2 Premier pas dans JADE	5
2.3 Configuration Eclipse et la création des agents du modèle :	6
2.4 Description du communication entre agents	8
2.5 Comportement des agents (Behaviour)	9
2.6 Etape de l'exécution	9

Table des figures

1.1	Processus de communication	3
1.2	Communication direct	4
2.1	Gui JADE	6
2.2	Mon premier agent	7
2.3	Compilation des classes	8
2.4	Exécution AgentY Eclipse	10
2.5	Exécution AgentX Eclipse	10
2.6	Exécution AgentZ Eclipse	11

Introduction générale

L'utilisation de systèmes d'information complexes, fortement interactifs et parfois distribués, doit comporter des niveaux suffisants d'assistance. L'identification des différents niveaux d'assistance nécessaires peut conduire à concevoir un véritable système de multi-assistance dans lequel la communication homme-machine jouera un rôle majeur. Les processus dynamiques, coopératifs et autonomes nécessaires à cette interaction doivent alors intégrer une représentation des connaissances et des comportements de l'utilisateur et posséder de réelle capacité à communiquer. L'approche multi-agents offre un niveau d'abstraction adapté à cette problématique.

En effet, les systèmes multi-agents (SMA) permettent de coordonner le comportement d'agents interagissant dans une société pour réaliser des tâches ou résoudre des problèmes.

Afin de structurer la communication entre les agents, des protocoles d'interaction sont élaborés, ils permettent de décrire explicitement les échanges de messages entre les agents, ils représentent un schéma commun de conversation utilisé pour exécuter une tâche, un protocole décrit les enchaînements possibles de messages entre les agents.

L'objectif de ce travail est d'aboutir à un modèle à base d'agents pour trouver un mécanisme de communication entre agents. Afin de faciliter le développement de ce modèle, nous avons choisi la plateforme JADE sur laquelle est basée notre travail.

1 | Modèle de communication entre agents

1.1 Système multi-agents

Un agent comme étant une entité physique ou virtuelle évoluant dans un environnement dont il n'a qu'une représentation partielle et sur lequel il peut agir. Il est capable de communiquer avec d'autres agents et est doté d'un comportement autonome.

Les agents peuvent être classés selon leurs degrés d'autonomie, de coopération et d'adaptation, caractéristiques généralement considérées comme principales. Le plus haut niveau de coopération accorde à l'agent des capacités de négociation.

- Quand l'agent a un but à atteindre, il doit être suffisamment autonome pour pouvoir prendre des initiatives permettant d'atteindre ce but.
- Pour que l'ensemble des agents constitue un système cohérent chacun d'entre eux doit avoir un certain degré de coopération.
- L'agent doit agir en fonction de son environnement, c'est à dire qu'il doit s'adapter à celui-ci.

Les systèmes multi-agent constituent une nouvelle technique de modélisation qui place l'objet d'étude au centre de sa démarche. Ces modèles représentent les actions individuelles, les interactions entre les acteurs et les conséquences de ces interactions sur la dynamique du système.

1.2 Processus de communication proposé

La communication c'est un mécanisme d'interactions entre agent. Nous insistons sur le fait que pour qualifier les agents d'agents intelligents, il est essentiel de prouver leur capacité à communiquer dans un but individuel ou collectif.

Le processus de communication inter-agents dans un système multi agents est représenté par un modèle d'architecture de fonctionnements à la figure (Figure 1.1).

Le processus que nous proposons se décompose en trois phases :

La première phase : Deux agents qui communiquent entre eux (AgentX, AgentZ)

- L'agent Z envoie un message : Bonjour, merci de me générer un nombre aléatoire N.
- L'agent X répond par un nombre aléatoire N.

La deuxième phase : Deux agents qui communiquent entre eux (AgentY, AgentZ)

- L'agent Z envoie un message : Bonjour, merci de me calculer le résultat en utilisant une formule avec un indice i.
- L'agent Y renvoi le résultat de calcul.

La troisième phase : L'agent Z va calculer la maximum entre les deux résultats qui sont générés par Agent Y et Agent X et il va l'afficher.

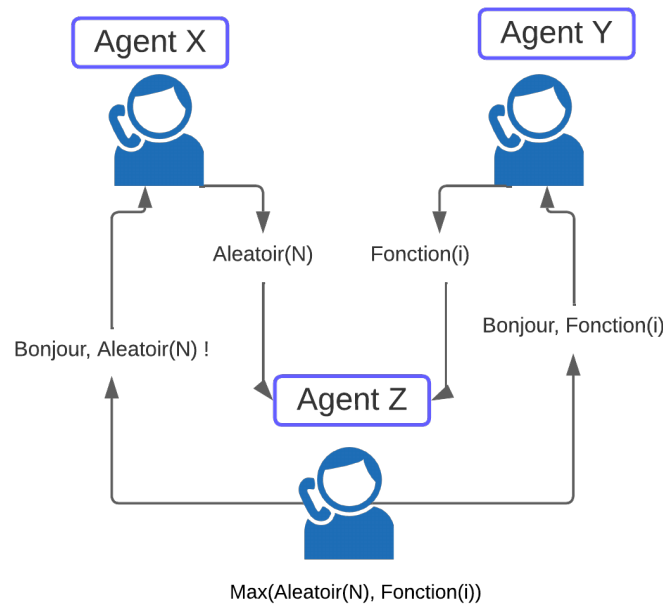


FIGURE 1.1 – Processus de communication

Cette communication est une communication direct car elle est intentionnelle et se fait par l'envoi de messages à plusieurs destinataires (mode diffusion). Elle se base sur trois éléments essentiels : (Voir figure 1.2)

- Le langage de communication : il permet de structurer les messages échangés entre les agents.

- L'ontologie : elle permet de rajouter un aspect sémantique pour les messages échangés entre les agents.
- Protocole d'interaction : il permet de définir un ordre sur les messages échangés entre les agents.

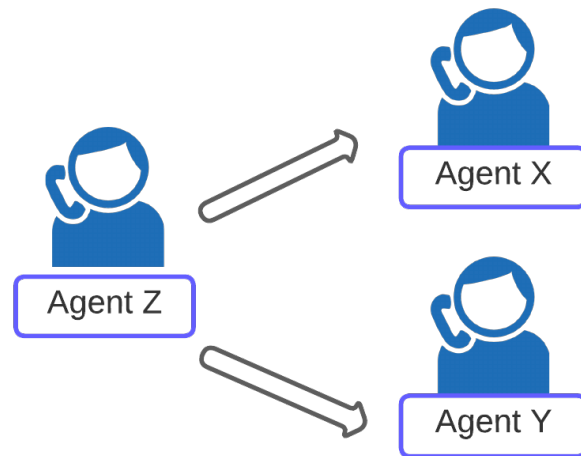


FIGURE 1.2 – Communication direct

2 | La plateforme JADE et la réalisation du modèle

2.1 Introduction

Il existe une multitude de plates-formes multi-agents dédiées à différents modèles d'agent. Les plates-formes fournissent une couche d'abstraction permettant de facilement implémenter les concepts des systèmes multi-agents.

Parmi ces plateformes, nous avons choisi JADE (Java Agent DEvelopment framework) qui permet de développer et d'exécuter des applications distribuées basées sur le concept d'agents. Les agents dans JADE sont implémentés selon 6 propriétés (Autonomie, Réactivité, Aspects sociaux, Dynamicité, Offre de service...)

2.2 Premier pas dans JADE

Après avoir installé le document *jade-all.zip*, on va le décompresser puis on doit mettre à jour le chemin de la librairie *jade.jar* dans la variable classpath afin de permettre, lors du démarrage de la plateforme depuis une invite de commandes, de retrouver les fichiers jar (contenant les classes compilées) de jade nécessaire à son lancement.

Pour vérifier que l'opération est bien réalisée, on tape dans l'invite de commande la commande suivante :

```
1 Java jade.Boot -gui
```

Premièrement, voici comment la plateforme JADE, une fois lancée, se présente :

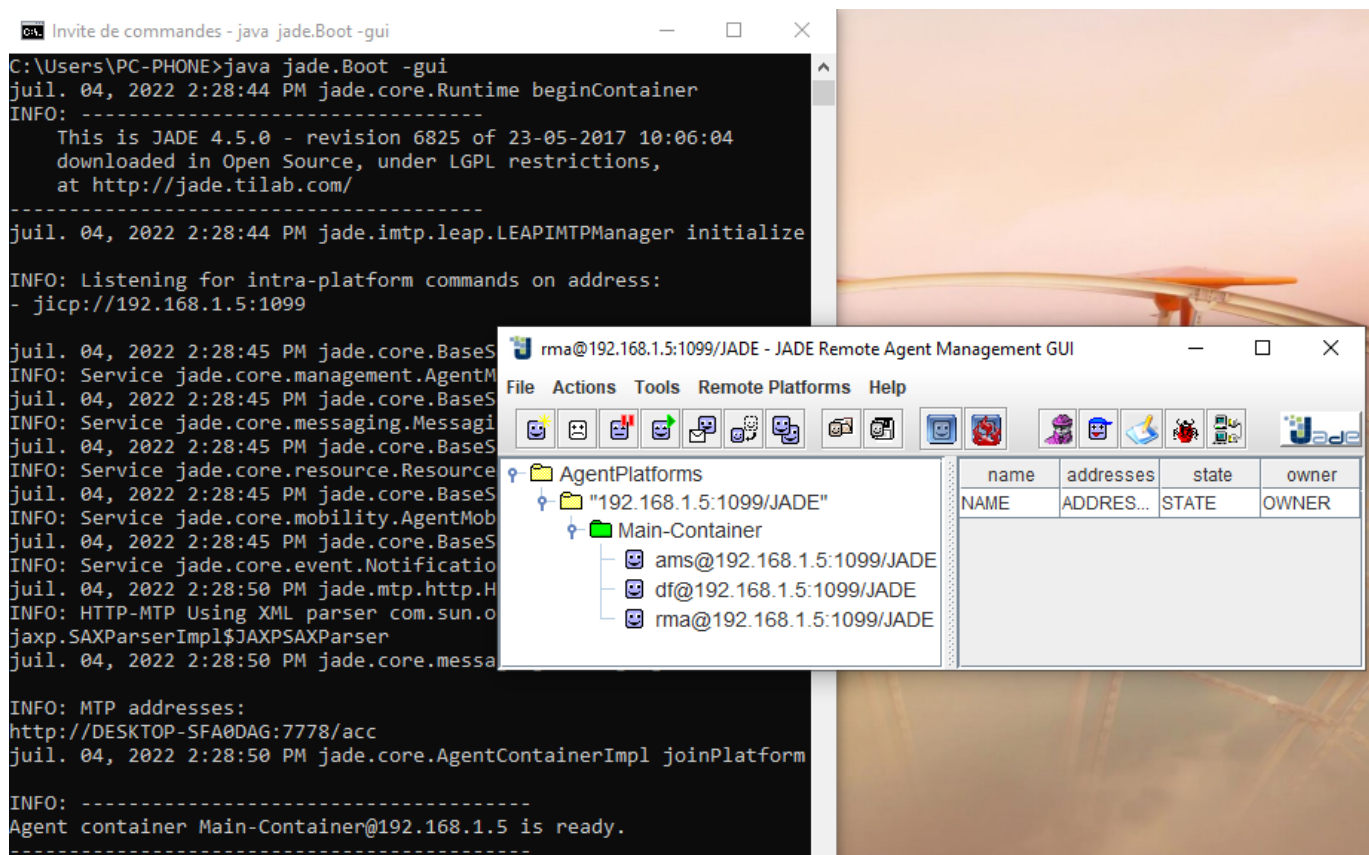


FIGURE 2.1 – Gui JADE

2.3 Configuration Eclipse et la création des agents du modèle :

Après l'installation de la plateforme JADE, on passe à la configuration :

1. Lancer Eclipse
2. Créer un nouveau projet TpJade
3. Créer les classes nécessaire

Pour cette démonstration, on va créer une classe agent qui sera en charge d'écrire dans le log « Je suis nomAgent » (voir Figure 2.2)

- L'Agent JADE possède une méthode `setup()` qui est invoquée dès la création de l'agent.
- La méthode `takedown()` qui est invoquée lors de la fin d'exécution de l'agent.

Pour ce faire voici la structure du code de la classe créé

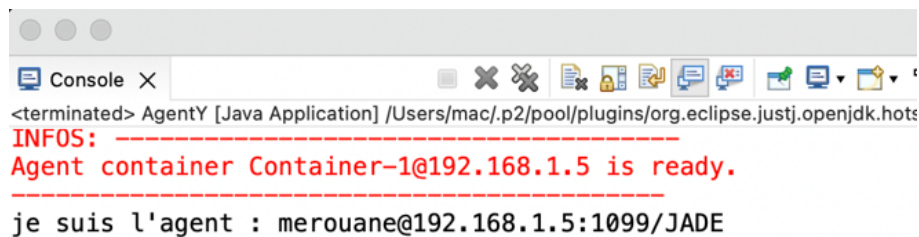
```

1 public class AgentX extends Agent{
2     protected void setup()
3     { //Première méthode exécutée lors du lancement de l'agent
4         System.out.println("je suis l'agent : " + this.getName()); }
5     /*D'autre action ...*/
6     protected void takeDown()
7     { //passe à l'état détruit
8         System.out.println("sa marche");}}

```

4. Pour exécuter l'agent via eclipse :

- Ouvrir le panel Run /Run configurations..
- Dans l'onglet "main", définir la "Main class" à **jade.Boot**
- Dans l'onglet "(x)= Arguments", définir le "Program arguments" à : **java jade.Boot -container merouane :AgentX**
- Cliquer sur "Apply", puis "run" (Voir Figure 2.2)



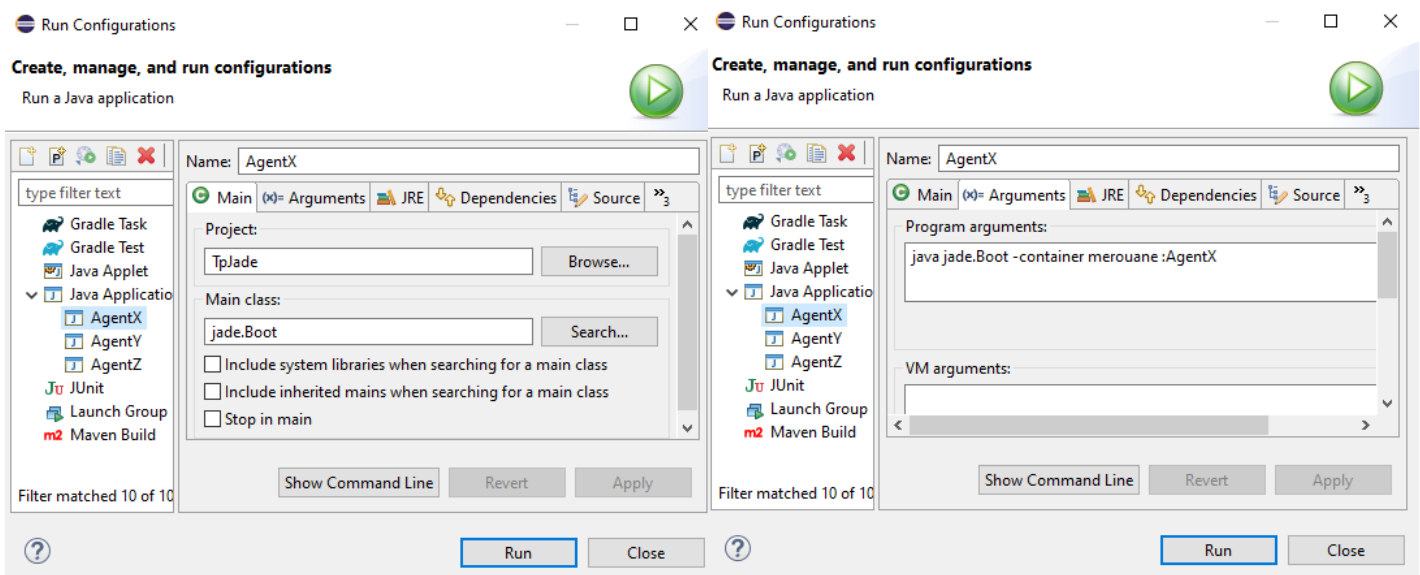
```

<terminated> AgentY [Java Application] /Users/mac/.p2/pool/plugins/org.eclipse.justj.openjdk.hot
INFOS: -----
Agent container Container-1@192.168.1.5 is ready.
-----
je suis l'agent : merouane@192.168.1.5:1099/JADE

```

FIGURE 2.2 – Mon premier agent

5. vous trouverez ci-après des captures d'écran liées à la configuration.



Après avoir lancé le GUI jade et configuré Eclipse, maintenant nous pouvons effectuer diverses actions sur les agents. Nous disposons de l'arborescence de la plateforme. Le « Main-Container » contient les agents spéciaux AMS, RMA et DF (voir figure 2.1) et les « Container » qui contiennent les agents (agentX, agentY et agentZ) (voir figure 2.2)

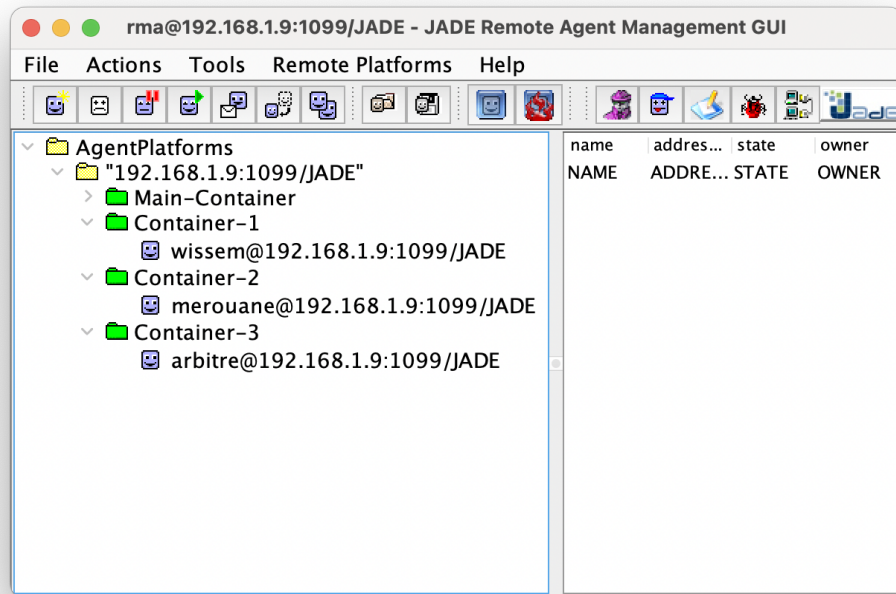


FIGURE 2.3 – Compilation des classes

2.4 Description du communication entre agents

Nous avons déjà expliqué dans le chapitre précédent notre architecture proposée, et dans ce qui suit nous verrons comment peuvent les agents de communiquer entre eux.

Principe :

- ✓ L'agent Z (sous le nom arbitre) envoie un message à l'agent X (sous le nom merouane) demandant qu'un nombre aléatoire soit généré, et ce dernier répond par un nombre N chaque 2 secondes. (Voir Figure 2.5)
- ✓ L'agent Z (sous le nom arbitre) envoie un message à l'agent X (sous le nom wissem) demandant le résultat de la fonction avec l'indice i. L'agent Y chaque 4 secondes calculera la fonction pour chaque i en utilisant la formule : $F=3*m*m + 5*m + 9$. (Voir Figure 2.4)
- ✓ Après avoir récupéré les valeurs générées par l'agent X et l'agent Y, l'agent Z pour chaque message reçu choisira la valeur la plus élevée et l'affichera. (Voir Figure 2.6)

2.5 Comportement des agents (Behaviour)

Chaque action réalisée par un agent est représenté comme un comportement de l'agent. Pour définir un nouveau comportement, il faut créer une nouvelle classe qui hérite de la classe *jade.core.behaviours.Behaviour*. Une fois le comportement défini, il faut l'appeler depuis la classe de l'agent (dans la méthode **setup()**) qui est censé le réaliser. Pour ajouter le comportement, la méthode à utiliser est **addBehaviour()** .

Une classe désignant un comportement doit toujours avoir la méthode suivante :

Action() – code du comportement qui sera exécuté par l'agent.

Les trois catégories d'agents ont un comportement cyclique, leur méthode **action()** exécute la même opération chaque fois qu'elle est appelée. Comme exemple, voici un code représentant l'implantation d'un comportement au sein d'une classe AgentX.

```

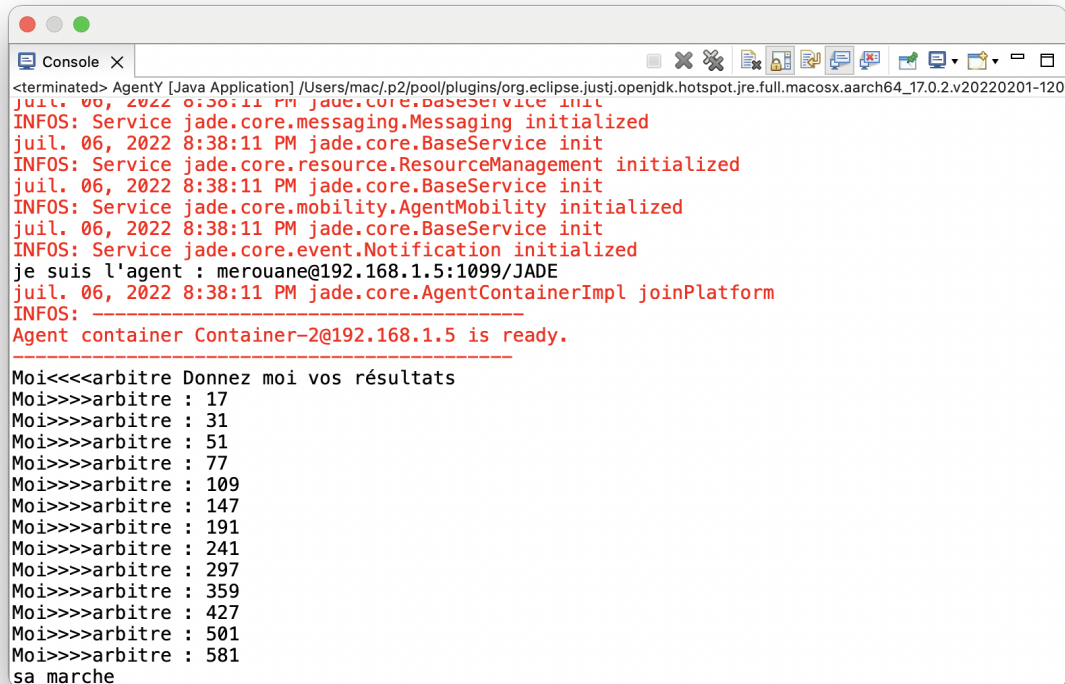
1  protected void setup() {
2
3      System.out.println("je suis l'agent : " + this.getName());
4      this.addBehaviour(new CyclicBehaviour() {
5          public void action() {
6              jade.lang.acl.ACLMessage msg = myAgent.receive();
7              if(msg == null) this.block();
8              else {
9                  System.out.println("Moi<<<<" + msg.getSender().getLocalName() + msg.getContent());
10             };
11             myAgent.addBehaviour(new CyclicBehaviour() {
12                 String mess = new String();
13                 Random rd = new Random();
14                 public void action() {
15                     jade.lang.acl.ACLMessage msgr = new jade.lang.acl.ACLMessage(jade.lang.acl.
16                     ACLMessage.INFORM);
17                     int NN=rd.nextInt(1000);
18                     mess=String.valueOf(NN);
19                     msgr.addReceiver(msg.getSender());
20                     msgr.setContent(mess);
21                     System.out.println("Moi>>>>" + msgr.getSender().getLocalName() + " : " + mess);
22                     myAgent.send(msgr);
23                     myAgent.doWait(1000);
24                 }
25             });
26         }
27     });
28 }

```

2.6 Etape de l'exécution

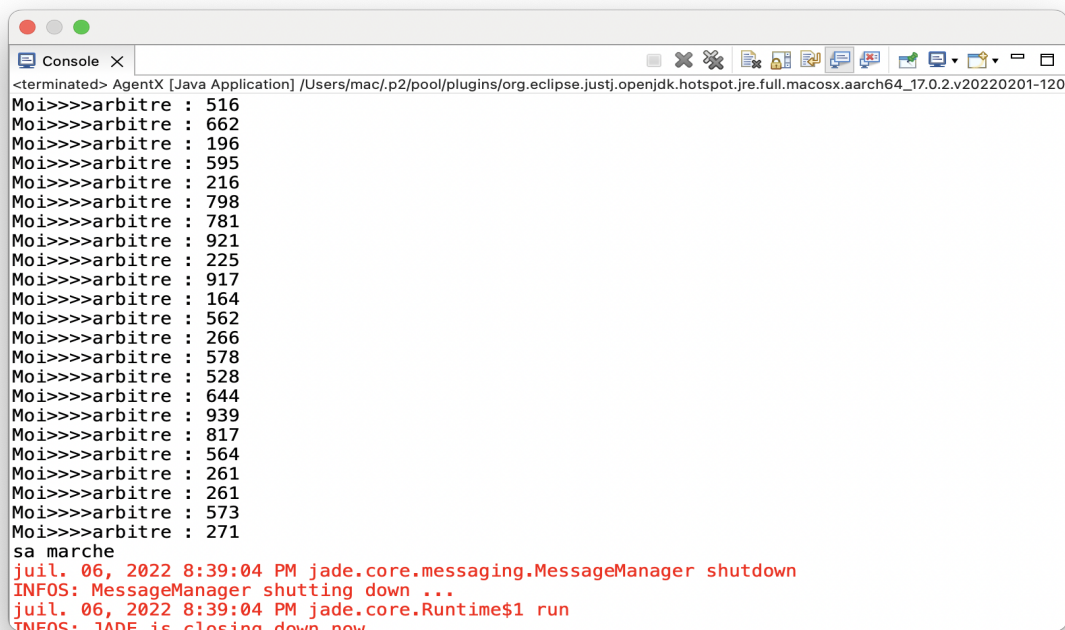
Après avoir créé les trois classes (JadeTP, AgentX, AgentY), nous passons à l'exécution.

1. On passe à la compilation et l'exécution qui commence par le démarrage du main container (voir Figure 2.3) de la classe AgentX et l'agentY sous eclipse(voir Figure 2.4 et Figure 2.5)



```
<terminated> AgentY [Java Application] /Users/mac/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.aarch64_17.0.2.v20220201-120
jul. 06, 2022 8:38:11 PM jade.core.BaseService init
INFO: Service jade.core.messaging.Messaging initialized
jul. 06, 2022 8:38:11 PM jade.core.BaseService init
INFO: Service jade.core.resource.ResourceManagement initialized
jul. 06, 2022 8:38:11 PM jade.core.BaseService init
INFO: Service jade.core.mobility.AgentMobility initialized
jul. 06, 2022 8:38:11 PM jade.core.BaseService init
INFO: Service jade.core.event.Notification initialized
je suis l'agent : merouane@192.168.1.5:1099/JADE
jul. 06, 2022 8:38:11 PM jade.core.AgentContainerImpl joinPlatform
INFO: -----
Agent container Container-2@192.168.1.5 is ready.
-----
Moi<<<arbitre Donnez moi vos résultats
Moi>>>arbitre : 17
Moi>>>arbitre : 31
Moi>>>arbitre : 51
Moi>>>arbitre : 77
Moi>>>arbitre : 109
Moi>>>arbitre : 147
Moi>>>arbitre : 191
Moi>>>arbitre : 241
Moi>>>arbitre : 297
Moi>>>arbitre : 359
Moi>>>arbitre : 427
Moi>>>arbitre : 501
Moi>>>arbitre : 581
sa marche
```

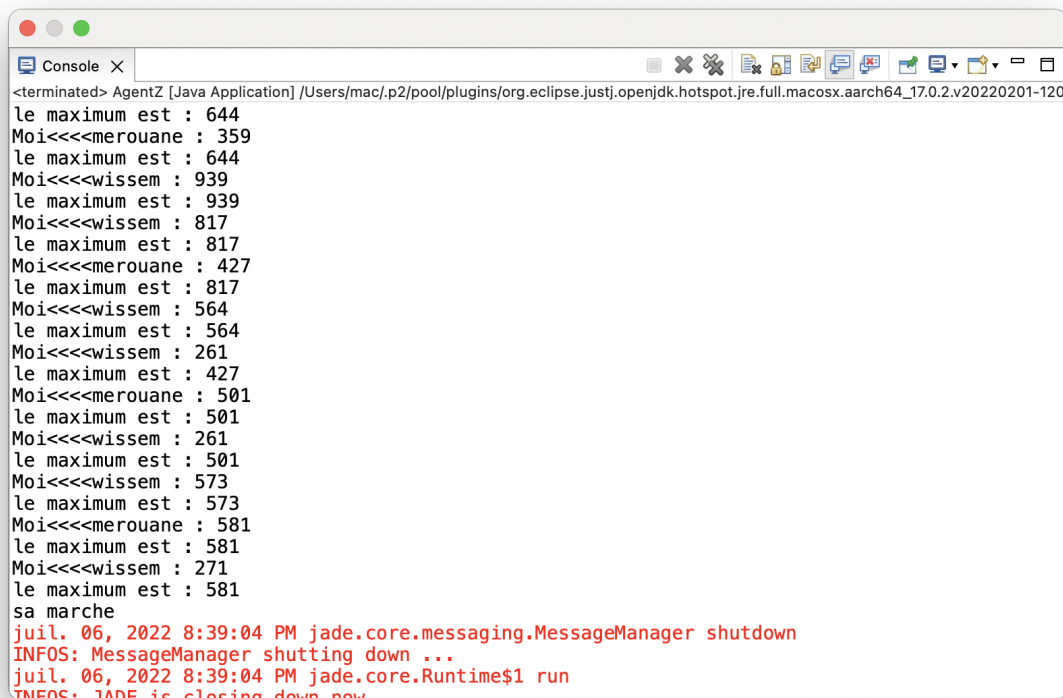
FIGURE 2.4 – Exécution AgentY Eclipse



```
<terminated> AgentX [Java Application] /Users/mac/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.aarch64_17.0.2.v20220201-120
Moi>>>arbitre : 516
Moi>>>arbitre : 662
Moi>>>arbitre : 196
Moi>>>arbitre : 595
Moi>>>arbitre : 216
Moi>>>arbitre : 798
Moi>>>arbitre : 781
Moi>>>arbitre : 921
Moi>>>arbitre : 225
Moi>>>arbitre : 917
Moi>>>arbitre : 164
Moi>>>arbitre : 562
Moi>>>arbitre : 266
Moi>>>arbitre : 578
Moi>>>arbitre : 528
Moi>>>arbitre : 644
Moi>>>arbitre : 939
Moi>>>arbitre : 817
Moi>>>arbitre : 564
Moi>>>arbitre : 261
Moi>>>arbitre : 261
Moi>>>arbitre : 573
Moi>>>arbitre : 271
sa marche
jul. 06, 2022 8:39:04 PM jade.core.messaging.MessageManager shutdown
INFO: MessageManager shutting down ...
jul. 06, 2022 8:39:04 PM jade.core.Runtime$1 run
INFO: JADE is closing down now
```

FIGURE 2.5 – Exécution AgentX Eclipse

2. Il nous reste que lancer l'agentZ (JadeTP) pour afficher les résultats de calcul avec *java jade.Boot -container JadeTP :AgentZ .* (Voir Figure 2.6)



```
<terminated> AgentZ [Java Application] /Users/mac/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.aarch64_17.0.2.v20220201-120
le maximum est : 644
Moi<<<<merouane : 359
le maximum est : 644
Moi<<<<wissem : 939
le maximum est : 939
Moi<<<<wissem : 817
le maximum est : 817
Moi<<<<merouane : 427
le maximum est : 817
Moi<<<<wissem : 564
le maximum est : 564
Moi<<<<wissem : 261
le maximum est : 427
Moi<<<<merouane : 501
le maximum est : 501
Moi<<<<wissem : 261
le maximum est : 501
Moi<<<<wissem : 573
le maximum est : 573
Moi<<<<merouane : 581
le maximum est : 581
Moi<<<<wissem : 271
le maximum est : 581
sa marche
juil. 06, 2022 8:39:04 PM jade.core.messaging.MessageManager shutdown
INFO: MessageManager shutting down ...
juil. 06, 2022 8:39:04 PM jade.core.Runtime$1 run
INFO: JADE is closing down now
```

FIGURE 2.6 – Exécution AgentZ Eclipse

Conclusion

Tout au long de ce mini projet nous avons pu renforcer nos connaissances en conception de système multi agent et cela en implémentant un modèle de communication entre trois agents en utilisant la librairie JADE qui permet à collaborer, corrdonner et s'échanger des messages.

Bibliographie

- [1] DEVELOPPEZ.COM. Créez votre premier agent avec JADE et ECLIPSE [en ligne].
[http ://djug.developpez.com/java/jade/creation-agent/](http://djug.developpez.com/java/jade/creation-agent/) (consulté le 27 mai 2022)
- [2] [https ://www.irit.fr/~Chihab.Hanachi/Cours/SMA/CoursAgentsI.pdf](https://www.irit.fr/~Chihab.Hanachi/Cours/SMA/CoursAgentsI.pdf), Introduction aux Systèmes Multi-Agents, consulté le 3 juillet 2022
- [3] SIEVERING, Johann, Semantic approach in the information systems, 2003, Genève.
- [4] Une architecture à base d'agents pour le workflow inter-organisationnel lâche, These de doctorat , consulté le 29 juin 2022