

TP N°1 : Initiation à la plateforme JADE (Création et exécution des agents)

1. Compétences à acquérir

Suite à ce TP, vous serez en mesure de :

- Se familiariser avec la plateforme JADE.
- Comprendre le principe de fonctionnement d'un système multi agents.
- Installer la plateforme JADE.
- Créer un agent et lancer son exécution pour afficher son nom local et son identifiant.
- Créer un agent qui reçoit des paramètres passés en arguments puis les afficher.

2. Introduction

JADE (Java Agent DEvelopment Framework) est un environnement de développement d'agent implémenté totalement dans le langage JAVA. Il facilite la mise en place d'un système multi-agent (SMA) répondant aux spécifications FIPA (Foundation for Intelligent Physical Agents) à travers un ensemble d'outils. La plateforme JADE inclut des composants qui contrôlent un système multi-agent :

- **Un runtime Environment** : l'environnement où les agents peuvent vivre, il doit être activé pour pouvoir lancer les agents.
- **Une librairie de classes** : que les développeurs utilisent pour écrire leurs programmes.
- **Une suite d'outils graphiques** : qui facilitent la gestion et la supervision de la plateforme des agents.

Chaque instance du JADE est appelée conteneur « container », elle peut contenir plusieurs agents. Un ensemble de conteneurs constituent une plateforme (voir Figure 2). Chaque plateforme doit contenir un conteneur spécial appelé « main container » et tous les autres conteneurs s'enregistrent auprès de celui-là dès leur lancement. Un « main-container » contient trois agents spéciaux (voir Figure 1) appelés **AMS** (Agent Management System), **DF** (Directory Facilitator) et **ACL** (Agent Communication Channel) qui sont créés automatiquement au lancement du « main-container ».



Figure 1 : Architecture du main-container.

- **AMS** (Agent Management System) permet de gérer le cycle de vie des agents, maintenir une liste de tous les agents qui résident sur la plateforme (pages blanches) et contrôler l'accès ainsi que l'utilisation du canal de communication des agents.
- **DF** (Directory Facilitator) permet d'enregistrer les services offerts par les agents (Pages jaunes) et rechercher les services offerts par les agents.
- **ACL** (Agent Communication Channel) permet de gérer les communications entre les agents.

Dans la plateforme JADE, chaque agent est identifié par un nom unique qui est l'**AgentIdentifier (AID)**, chaque agent peut joindre ou quitter librement la plateforme JADE et rentrer en contact avec chacun des autres agents.

La figure ci-dessous présente un exemple constitué de deux plateformes connectées à travers un réseau de communication. Les « container » 1 et 2 appartiennent au même « main container » de la plateforme 1. Les agents A2 et A3 appartiennent au même « container » alors que les agents A4 et A3 n'appartiennent pas au même « container ». Les agents A4 et A5 n'appartiennent pas à la même plateforme.

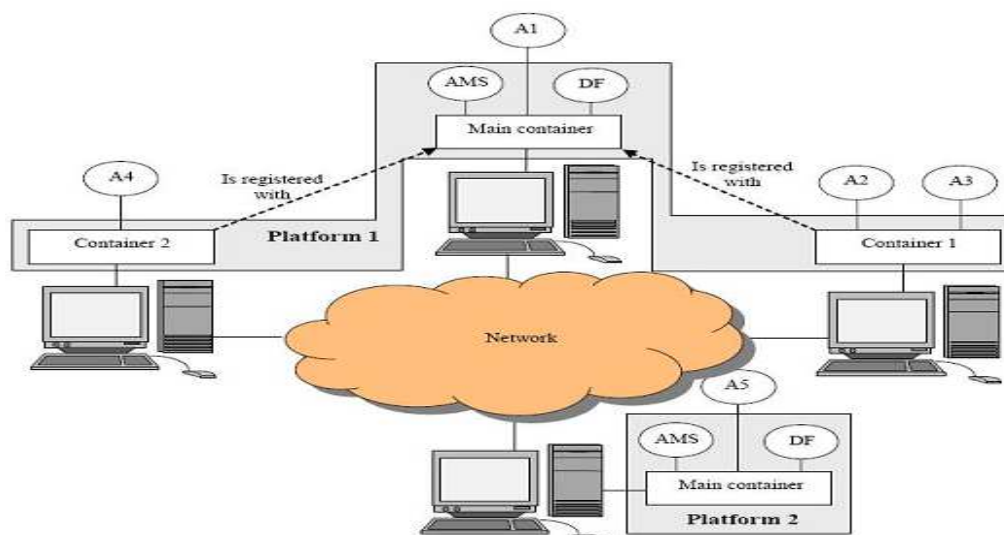
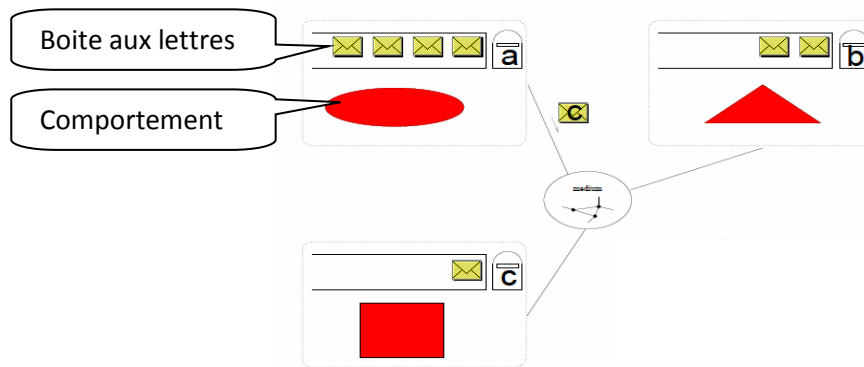


Figure 2 : Exemple de « container » (Environnement d'exécution pour les agents)

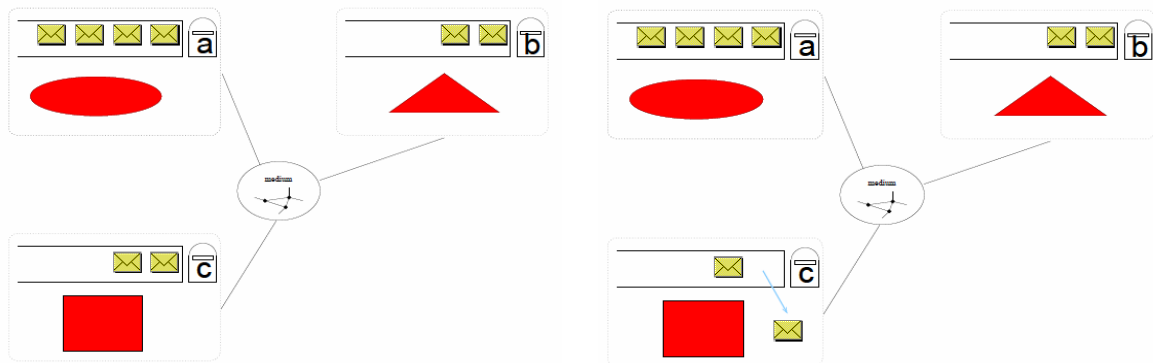
3. Système multi agents

Un système multi agents est un système distribué composé d'un ensemble d'agents. Un agent est une entité autonome réelle ou abstraite, possède un cycle de vie et peut **communiquer avec d'autres agents au moyen de messages**, les **messages** reçus sont stockés dans une **boîte aux lettres (file d'attente)**. Un agent possède des compétences et offres des services, il est caractérisé par un ou plusieurs **comportements** qui décrivent la **réaction** de l'agent à un message reçu ou à une observation de ses connaissances. Ce **comportement** peut changer pendant son exécution.

La figure ci-dessous montre un exemple d'un système multi agents composé de 3 agents « a », « b » et « c » qui sont connectés à travers un réseau de communication. Chaque agent possède une boîte aux lettres et un comportement représenté par une forme géométrique (ellipse, triangle, rectangle, ...etc.).

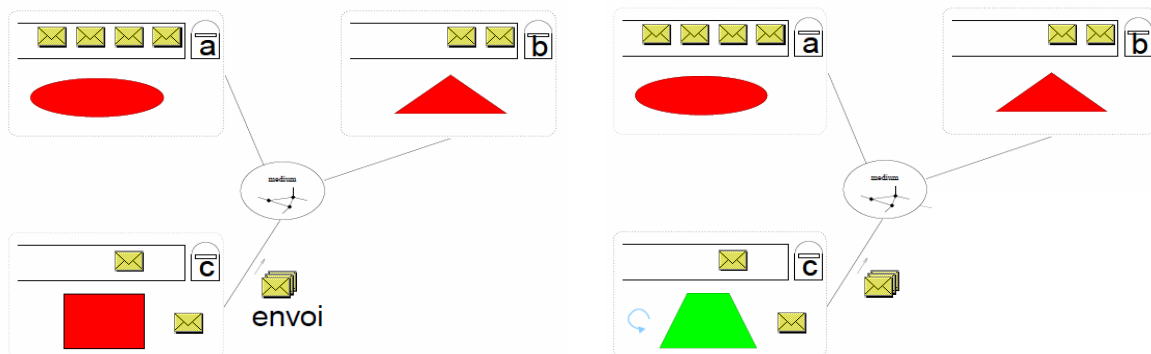


(a) l'agent « a » envoie un message à l'agent « c »



(b) l'agent « c » reçoit le message et le stocke dans sa boîte aux lettres

(c) l'agent « c » consulte le 1er message de sa boîte aux lettres



(d) l'agent « c » (après la consultation du 1er message) envoie des messages aux autres agents

(e) l'agent « c » (après la consultation du 1er message) change son comportement

Figure 3 : Exemple d'un système multi agents.

4. Installation de la plateforme JADE

Pour installer la plateforme JADE sur un ordinateur, il faut suivre les étapes décrites ci-dessous :

1. Lancer votre navigateur et taper l'adresse **<http://www.jade.tilab.com>**



Figure 4 : Page d'accueil du site <http://www.jade.tilab.com>.

2. Télécharger le fichier **JADE-all-4.5.0.zip** à partir du bouton **Download** (encadré par un rectangle rouge dans la Figure 4).
3. Mettre le fichier **JADE-all-4.5.0.zip** dans une partition de votre disque dur (par exemple : C:, D: ou E:).
4. Décompresser le fichier **JADE-all-4.5.0.zip**. (Faire un clic droit sur le nom du fichier puis cliquer sur **Extraire vers JADE-all-4.5.0.zip** (voir Figure 5.a)).
5. Accéder au dossier **JADE-all-4.5.0.zip** et décompresser les fichiers (JADE-bin-4.5.0.zip, JADE-doc-4.5.0.zip, JADE-examples-4.5.0.zip, JADE-src-4.5.0.zip (voir Figure 5.b)).

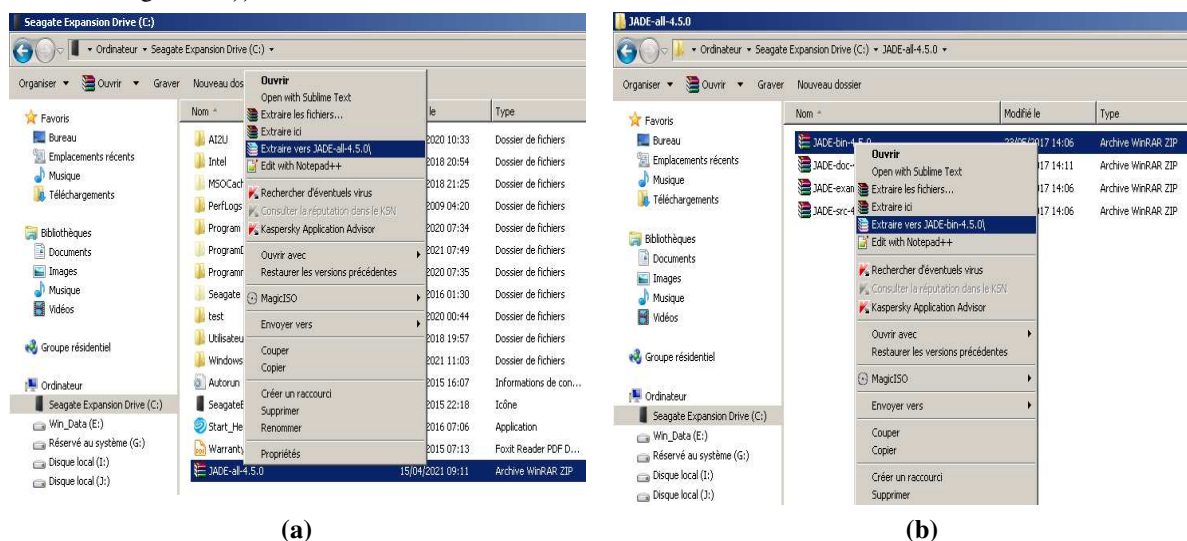


Figure 5 : Etapes de décompression de la plateforme JADE.

5. Création et exécution du premier agent sous la plateforme JADE

5.1. Création du premier agent

Afin de créer un agent avec JADE et en utilisant le langage de programmation JAVA, il faut suivre les étapes présentées ci-dessous :

5.1.1. Sous Eclipse

1. Lancer Eclipse.
2. Créer un nouveau projet en cliquant sur : **File/New/Java Project**.
3. Donner un nom au projet créé (**TPALDI01** par exemple).
4. Cliquer sur **Next**, puis sur l'onglet **Libraries**, ensuite sur le bouton **Add External JARs**, puis double cliquer sur le fichier **jade.jar** qui se trouve dans le répertoire (**C:/JADE-all-4.5.0/JADE-bin-4.5.0/jade/lib** (voir Figure 6)).

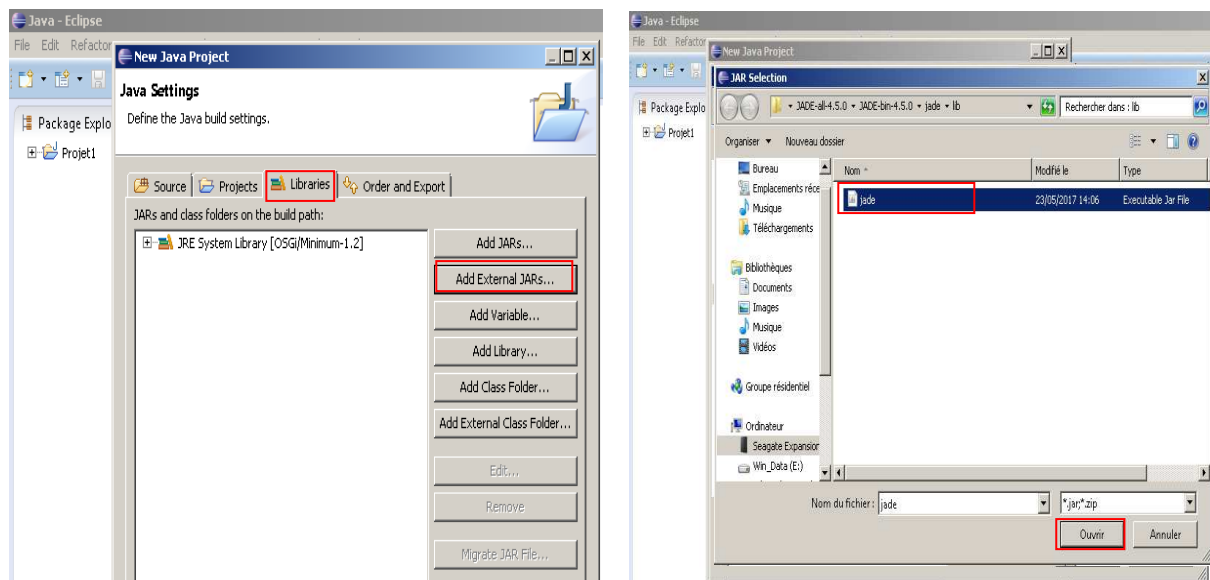


Figure 6 : Ajout de la plateforme JADE sous Eclipse.

5. Cliquer sur le bouton **Finish**.
6. Créer une nouvelle classe en cliquant sur : **File/New/Class**.
7. Donner un nom à la classe créée (**AgentTest** par exemple).
8. Cliquer sur le bouton **Finish**.

5.1.2. Sous NetBeans

1. Lancer NetBeans.
2. Créer un nouveau projet en cliquant sur : **Fichier/Nouveau projet.../Java**.
3. Cliquer sur **Suivant**.
4. Donner un nom au projet créé (**TPALDI01** par exemple).
5. Dans l'arborescence du projet, faire un clic droit sur **Bibliothèques**, puis cliquer sur **Ajouter un fichier JAR ou un dossier...**, ensuite double cliquer sur le fichier **jade.jar** qui se trouve dans le répertoire (**C:/JADE-all-4.5.0/JADE-bin-4.0.1/jade/lib**) (voir Figure 7).

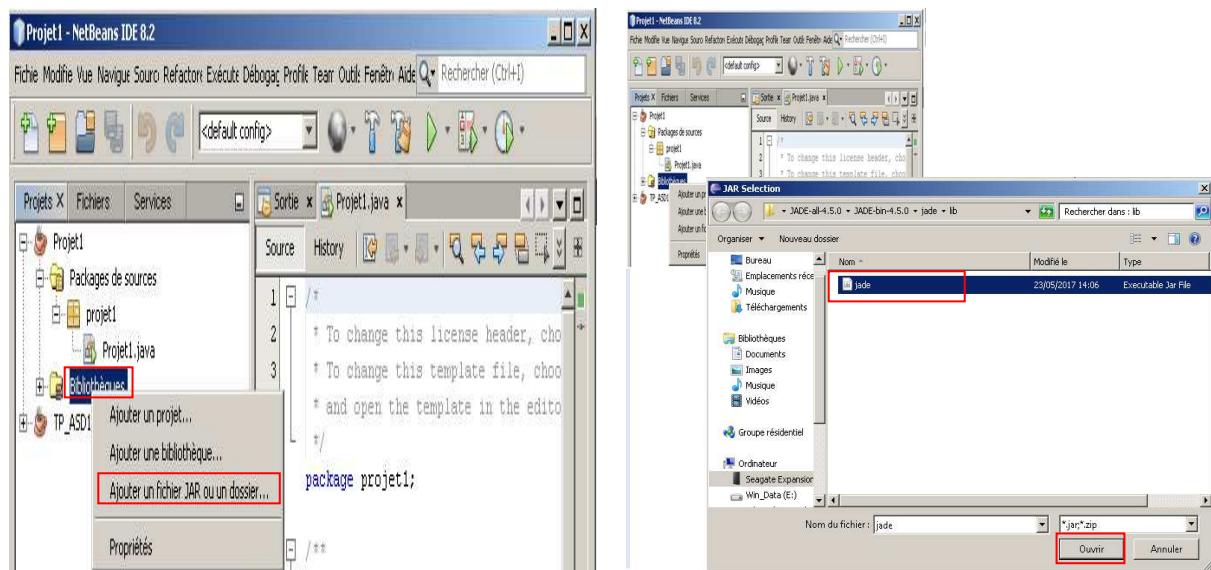


Figure 7 : Ajout de la plateforme JADE sous NetBeans

6. Créer une nouvelle classe en cliquant sur : **Fichier/Nouveau fichier.../Java Class.**
7. Cliquer sur **Suivant.**
8. Donner un nom à la classe créée (**AgentTest** par exemple).
9. Cliquer sur le bouton **Terminer.**

Une fois que la classe dédiée à l'agent sera créée, le code qui sera exécuté par l'agent doit être écrit dans la classe créée. Dans notre cas, nous tapons le code suivant qui permet d'afficher le nom de l'agent :

Chaque agent **hérite** de la classe **jade.core.Agent**

```

import jade.core.Agent;
public class AgentTest extends Agent{
    protected void setup () {
        System.out.println("Je suis l'agent : "+this.getLocalName());
        System.out.println("Je suis l'agent : "+this.getName());
    }
}
        
```

Chaque agent doit avoir une méthode obligatoire **void setup()** dans laquelle, il :

- Récupère la liste des paramètres passée en arguments.
- Enregistre les Services offerts par l'agent auprès du DF.
- Lance ses comportements (behaviours).

Un agent est identifié par un nom **unique** l'**AgentIdentifier (AID)** qui a la forme suivante :

nom_local@nom_plateforme

Exemple :

AgentProcess@192.168.56.1:1099/JADE

- La méthode **getLocalName()** permet de récupérer le **nom local** de l'agent.
- La méthode **getName()** permet de récupérer l'**AgentIdentifier** de l'agent.

5.2. Exécution du premier agent

Pour pouvoir lancer l'exécution de votre premier agent, vous pouvez utiliser l'une des deux méthodes suivantes :

5.2.1. 1^{ère} méthode

1. Cliquer sur **Run/Run Configuration ...**
2. Dans la partie gauche de la fenêtre, double cliquer sur : **Java Application.**
3. Dans l'onglet Main :
 - Cliquer sur **Browse...** et choisir le projet en cours.
 - Cocher la case : «**Include libraries when searching for a main class**».
 - Cliquer sur **Search...** puis sur «**Boot - jade**»
4. Dans l'onglet arguments, taper la commande :

-cp jade.boot agent1:AgentTest

Nom local de l'agent Nom de la classe de l'agent

Remarque

Si vous avez un package dans votre projet alors la commande s'écrit sous cette forme : **-cp jade.boot NomAgent:NomPackage.NomClasse**

5. Cliquer sur **Apply** pour ne pas refaire cette configuration plusieurs fois pour le même projet.
6. Cliquer sur **Run.**

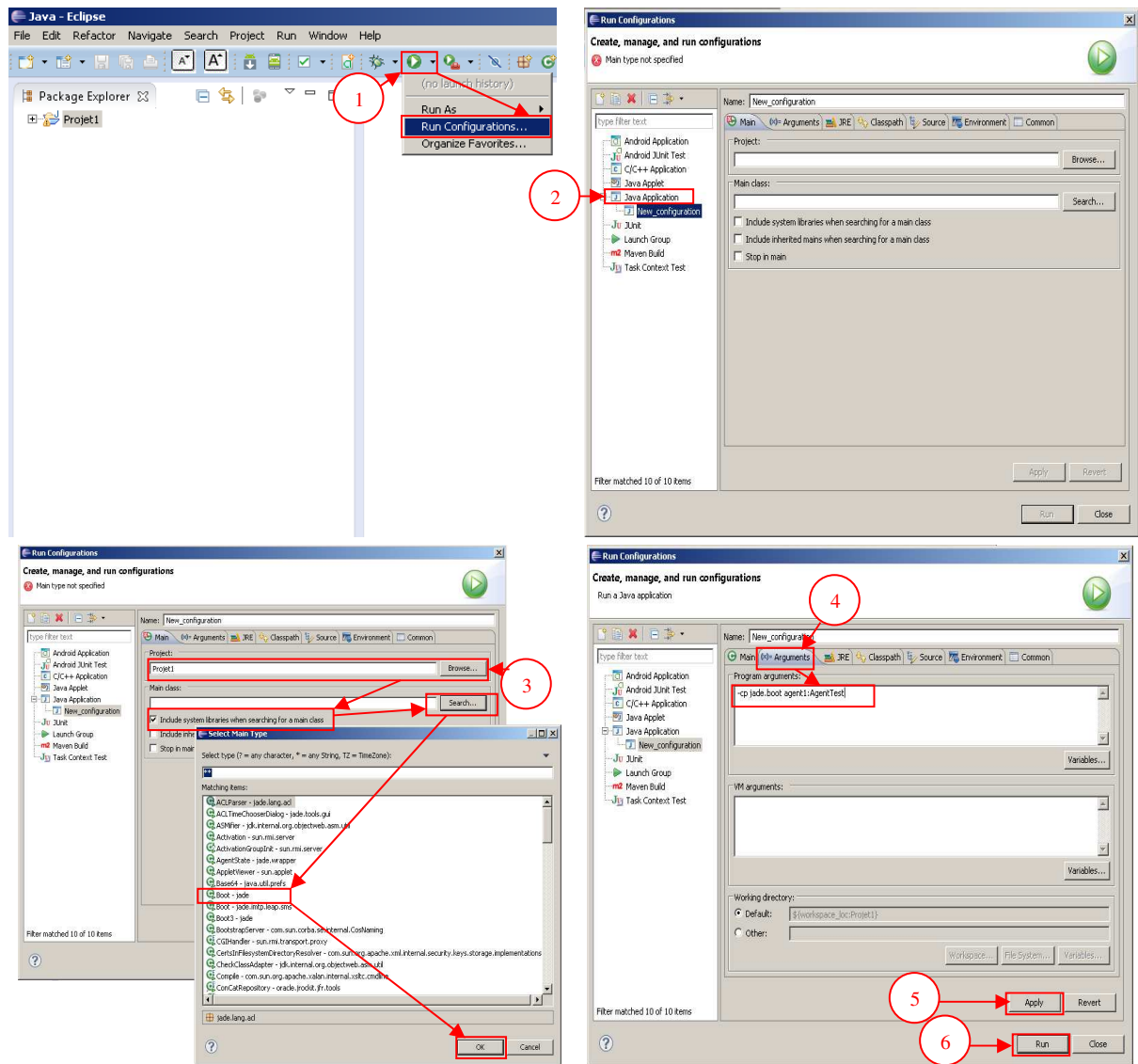


Figure 8 : Etapes d'exécution d'un programme écrit sous la plateforme JADE en utilisant l'IDE Eclipse.

5.2.2. 2^{ème} méthode

1. Créer une nouvelle classe et lui donner un nom (**Test** par exemple).
2. Taper le code suivant :

```
public class Test {
    public static void main(String[] args) {
        String [] commande = new String[3];
        String argument = "";
        argument = argument+"agent1:AgentTest";
        commande [0]="-cp";
        commande [1]="jade.boot";
        commande [2]= argument;
        jade.Boot.main(commande);
    }
}
```

Le nom de la classe dans laquelle vous avez tapé le code de l'agent

Le nom que vous allez donner à l'agent

3. Cliquer sur **Run** pour exécuter la méthode main de la classe **Test**.

Remarque

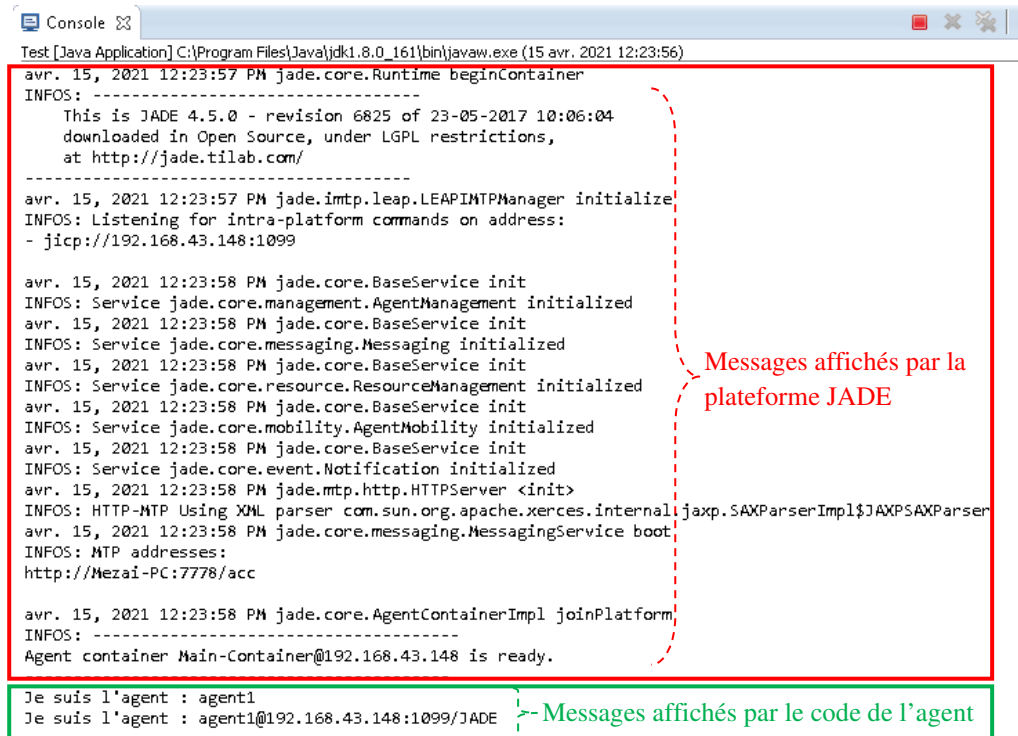
Si les classes sont dans un package alors utiliser l'instruction :

argument=argument+"agent1:**NomPackage.AgentTest**";

à la place de l'instruction :

argument = argument+"agent1:AgentTest";

Une fois que le programme est exécuté, le résultat suivant sera affiché au niveau de la console :



```
Test [Java Application] C:\Program Files\Java\jdk1.8.0_161\bin\javaw.exe (15 avr. 2021 12:23:56)
avr. 15, 2021 12:23:57 PM jade.core.Runtime beginContainer
INFOS: -----
This is JADE 4.5.0 - revision 6825 of 23-05-2017 10:06:04
downloaded in Open Source, under LGPL restrictions,
at http://jade.tilab.com/
-----
avr. 15, 2021 12:23:57 PM jade.imtp.leap.LEAPIIMPManager initialize
INFOS: Listening for intra-platform commands on address:
- jicp://192.168.43.148:1099

avr. 15, 2021 12:23:58 PM jade.core.BaseService init
INFOS: Service jade.core.management.AgentManagement initialized
avr. 15, 2021 12:23:58 PM jade.core.BaseService init
INFOS: Service jade.core.messaging.Messaging initialized
avr. 15, 2021 12:23:58 PM jade.core.BaseService init
INFOS: Service jade.core.resource.ResourceManagement initialized
avr. 15, 2021 12:23:58 PM jade.core.BaseService init
INFOS: Service jade.core.mobility.AgentMobility initialized
avr. 15, 2021 12:23:58 PM jade.core.BaseService init
INFOS: Service jade.core.event.Notification initialized
avr. 15, 2021 12:23:58 PM jade.mtp.http.HTTPServer <init>
INFOS: HTTP-NTP Using XML parser com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl$JAXPSAXParser
avr. 15, 2021 12:23:58 PM jade.core.messaging.MessagingService boot
INFOS: MTP addresses:
http://Mezai-PC:7778/acc

avr. 15, 2021 12:23:58 PM jade.core.AgentContainerImpl joinPlatform
INFOS: -----
Agent container Main-Container@192.168.43.148 is ready.


Je suis l'agent : agent1
Je suis l'agent : agent1@192.168.43.148:1099/JADE
```

Messages affichés par la plateforme JADE

Messages affichés par le code de l'agent

Figure 9 : Exemple d'exécution d'un agent.

Remarque

Pour faire une deuxième exécution d'un programme écrit sous la plateforme JADE, il faut d'abord arrêter la première exécution en cliquant sur le bouton **Terminate**  qui se trouve au de la console. Cela est dû au fait que sur une machine, nous ne pouvons créer plusieurs Main-Container.

5.3. Exécution de plusieurs agents ayant la même classe

Pour pouvoir lancer l'exécution de plusieurs agents ayant avec la même classe, utiliser l'une des deux méthodes suivantes :

5.3.1. 1^{ère} méthode

Suivre les mêmes étapes présentées dans la section 5.2.1. sauf que la commande à écrire doit être sous la forme suivante :

-cp jade.boot agent1:AgentTest;agent2:AgentTest

Nom du 1^{er} agent Nom du 2^{ème} agent

5.3.2. 2^{ème} méthode

Suivre les mêmes étapes présentées dans de la section 5.2.2 en apportant à la classe Test les modifications suivantes :

```
public class Test {  
    public static void main(String[] args) {  
        String [] commande = new String[3];  
        String argument = "";  
        argument = argument+"agent1:AgentTest";  
        argument = argument+";";  
        argument = argument+"agent2:AgentTest"; ← Ajout du 2ème agent  
        commande [0]="-cp";  
        commande [1]="jade.boot";  
        commande [2]= argument;  
        jade.Boot.main(commande);  
    }  
}
```

Une fois que le programme est exécuté, le résultat suivant sera affiché au niveau de la console :

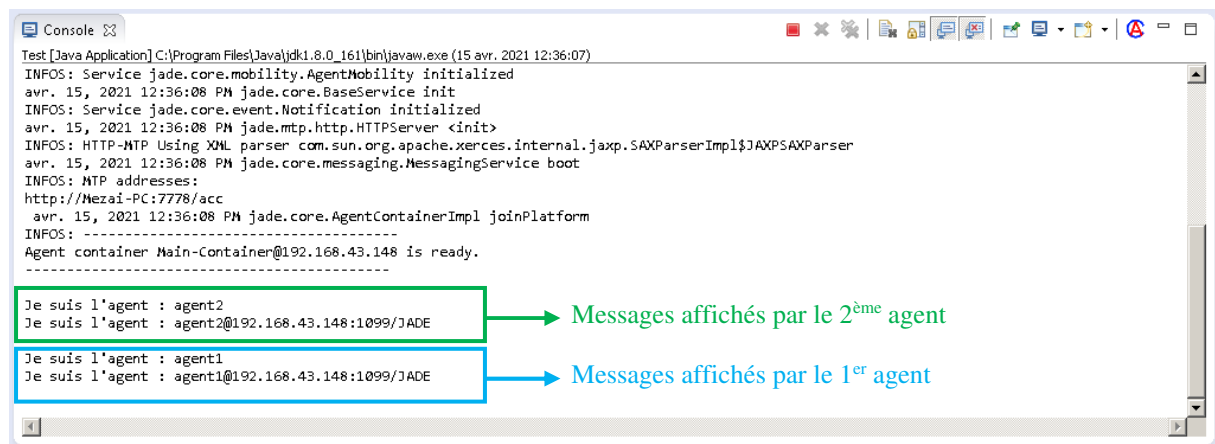


Figure 10 : Exemple d'exécution de deux agents ayant la même classe.

6. Récupération des paramètres passés en arguments

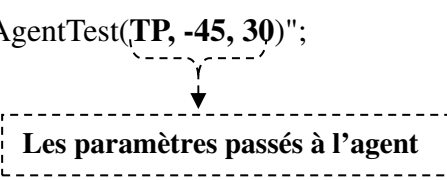
Dans la plateforme JADE, nous pouvons faire passer des paramètres aux agents. Chaque agent récupère ses paramètres en utilisant la méthode **getArguments()**. Puis, il pourra les convertir selon un format adéquat et les utiliser dans son code.

Voici un exemple de code qui montre comment récupérer et afficher les paramètres passés en arguments :

```
import jade.core.Agent;
public class AgentTest extends Agent{
    protected void setup() {
        System.out.println("Je suis l'agent:"+this.getLocalName());
        System.out.println(" Mes arguments sont :");
        Object[] args = getArguments(); //récupérer la liste des paramètres
                                         // et la mettre dans un tableau d'Objets « args »
        if (args != null) { //tester si le tableau « arg » est non vide
            for (int i = 0; i < args.length; ++i){//parcourir le tableau args
                System.out.println(args[i]); //afficher le paramètre
                                                //qui se trouve à l'indice « i »
            }
        }
    }
}
```

Pour pouvoir tester le programme présenté ci-dessus, créer une classe Test et taper le code suivant :

```
public class Test {
    public static void main(String[] args) {
        String [] commande = new String[3];
        String argument = "";
        argument = argument+"agent1:AgentTest(TP, -45, 30)";
        commande [0]="-cp";
        commande [1]="jade.boot";
        commande [2]= argument;
        jade.Boot.main(commande);
    }
}
```



Une fois que le programme est exécuté, le résultat suivant sera affiché au niveau de la console :

```

Test (1) [Java Application] C:\Program Files\Java\jdk1.8.0_161\bin\javaw.exe (15 avr. 2021 12:42:03)
avr. 15, 2021 12:42:04 PM jade.core.BaseService init
INFOS: Service jade.core.mobility.AgentMobility initialized
avr. 15, 2021 12:42:04 PM jade.core.BaseService init
INFOS: Service jade.core.event.Notification initialized
avr. 15, 2021 12:42:04 PM jade.mtp.http.HTTPServer <init>
INFOS: HTTP-MTP Using XML parser com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl$JAXPSAXParser
avr. 15, 2021 12:42:04 PM jade.core.messaging.MessagingService boot
INFOS: MTP addresses:
http://Mezai-PC:7778/acc
avr. 15, 2021 12:42:04 PM jade.core.AgentContainerImpl joinPlatform
INFOS: -----
Agent container Main-Container@192.168.43.148 is ready.
Je suis l'agent:agent1
Mes arguments sont :
TP
-45
30
  
```

Figure 11 : Exemple d'exécution d'un agent qui reçoit une liste de paramètres passée en arguments.

Pour pouvoir tester l'exemple précédent avec plusieurs agents, modifier à la classe Test comme suit :

```

public class Test {
    public static void main(String[] args) {
        String [] commande = new String[3];
        String argument = "";
        argument = argument+"agent1:AgentTest(TP, -45, 30)";
        argument = argument+";";
        argument = argument+"agent2:AgentTest(ALDI, 5, 13)";
        commande [0]="-cp";
        commande [1]="jade.boot";
        commande [2]= argument;
        jade.Boot.main(commande);
    }
}
  
```

Les paramètres passés au 1^{er} agent

Les paramètres passés au 2^{ème} agent

Une fois que le programme est exécuté, le résultat suivant sera affiché au niveau de la console :

```

Test (1) [Java Application] C:\Program Files\Java\jdk1.8.0_161\bin\javaw.exe (15 avr. 2021 12:45:47)
INFOS: HTTP-MTP Using XML parser com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl$JAXPSAXParser
avr. 15, 2021 12:45:48 PM jade.core.messaging.MessagingService boot
INFOS: MTP addresses:
http://Mezai-PC:7778/acc
avr. 15, 2021 12:45:48 PM jade.core.AgentContainerImpl joinPlatform
INFOS: -----
Agent container Main-Container@192.168.43.148 is ready.
Je suis l'agent:agent2
Mes arguments sont :
ALDI
5
13
Je suis l'agent:agent1
Mes arguments sont :
TP
-45
30
  
```

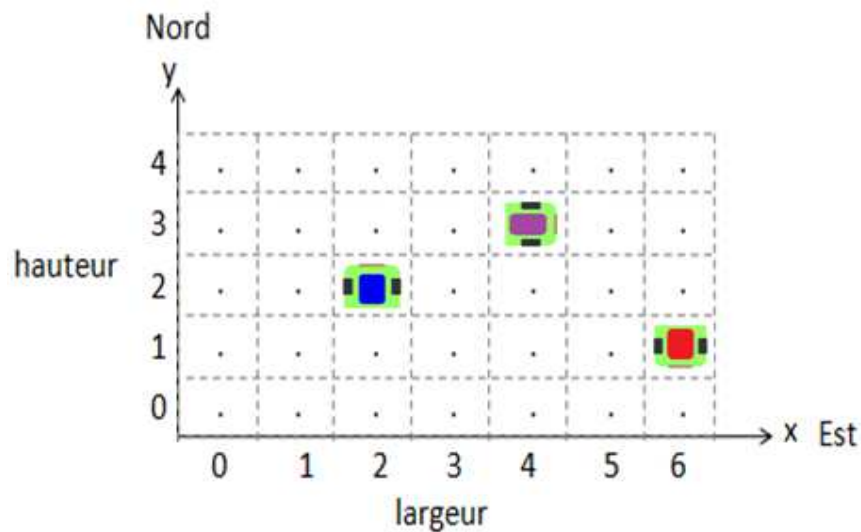
Figure 12 : Exemple d'exécution de deux agents qui reçoivent une liste de paramètres passée en arguments.

7. Test de connaissances

Afin de tester les connaissances acquises dans ce TP, nous vous proposons de faire cet exercice et le remettre dans un délai d'une semaine.

Enoncé

Nous désirons développer une application pour simuler le comportement des robots mobiles sur un plan orthonormé à 4 directions. Un robot sera considéré comme un agent qui reçoit dans sa liste de paramètres les données suivantes : les dimensions (hauteur, largeur) du plan de déplacement, la position initiale (donnée par les coordonnées x et y) et une direction (indiquée par l'une des valeurs : « Nord », « Sud », « Est » et « Ouest »). Chaque robot doit récupérer ses paramètres et les convertir selon un format adéquat pour les stocker dans des variables puis les afficher.



Travail demandé

1. Proposer une implémentation de la classe Robot.
2. Tester la classe Robot en utilisant un seul robot puis deux robots.

Exemple de Robot avec paramètres :

Robucar:Robot(15, 20, 7, 10, Est)

Atlas:Robot(15 , 20, 4, 3, Nord);NAO:Robot(15, 20, 12, 5, Sud)

3. Envoyer par mail le classe Robot et les captures d'écran des 2 exécutions.

Remarque

Pour transformer une chaîne de caractères en un nombre entier, utiliser la méthode **Integer.parseInt(...)**.

Exemple

```
String st = "123";  
int val = Integer.parseInt(st);
```