



Université Abdelhamid Mehri Constantine 2- Algérie
Faculté des Nouvelles Technologies de l'Information et de la Communication
Département d'Informatique Fondamentale et ses Applications



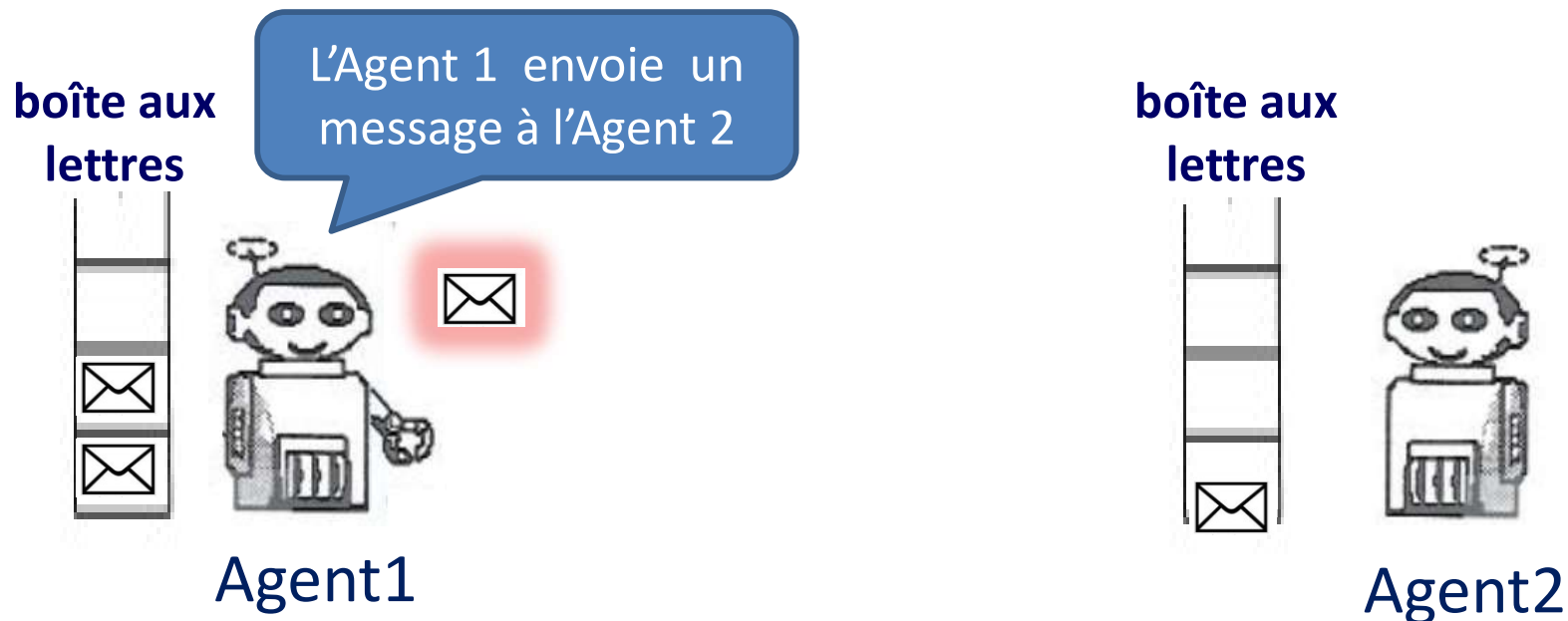
**1^{ère} Année Master Sciences et Technologies de l'information et de la
Communication (M1STIC)**
TP Algorithmes distribués (ALDI)

TP 03 : Initiation à la plateforme JADE (Communication entre les agents)

Année universitaire : 2023/2024

Introduction

- ✓ La communication entre les agents dans la plateforme JADE se fait par l'envoi de messages asynchrones,
- ✓ Chaque agent possède une **boîte aux lettres** qui contient les **messages** envoyés par d'autres agents,
- ✓ Les messages reçus sont stockés selon l'ordre chronologique de leur arrivée.

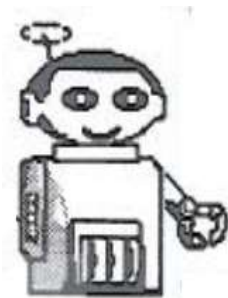
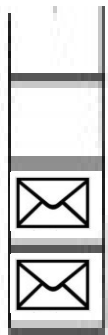


Plateforme JADE

Introduction

- ✓ La communication entre les agents dans la plateforme JADE se fait par l'envoi de messages asynchrones,
- ✓ Chaque agent possède une **boîte aux lettres** qui contient les **messages** envoyés par d'autres agents,
- ✓ Les messages reçus sont stockés selon l'ordre chronologique de leur arrivée.

boîte aux lettres



Agent1

boîte aux lettres



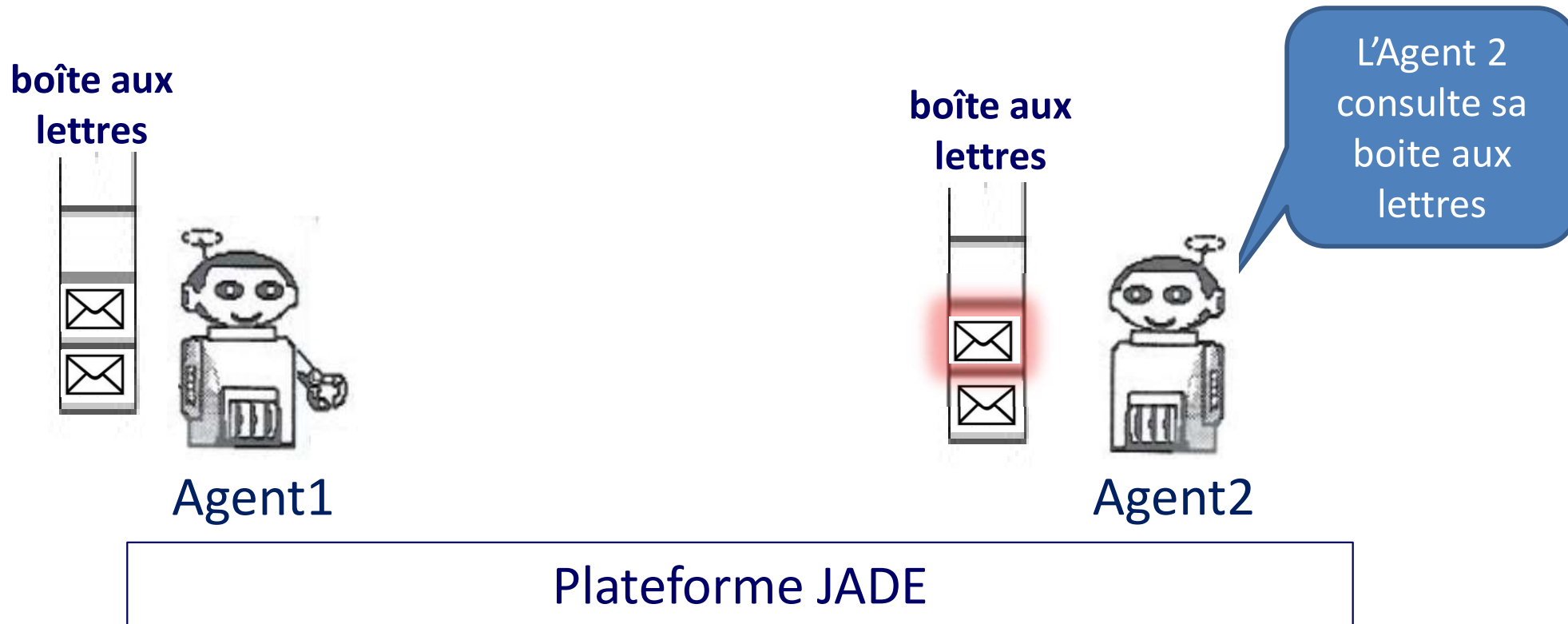
Agent2

Le message envoyé
sera stocké dans la
boite aux lettres de
l'agent destinataire

Plateforme JADE

Introduction

- ✓ La communication entre les agents dans la plateforme JADE se fait par l'envoi de messages asynchrones,
- ✓ Chaque agent possède une **boîte aux lettres** qui contient les **messages** envoyés par d'autres agents,
- ✓ Les messages reçus sont stockés selon l'ordre chronologique de leur arrivée.



Format d'un message JADE

- ✓ Un message JADE est une instance de la classe **ACLMessage** du package **jade.lang.acl**.
- ✓ Chaque message contient les champs suivants :

L'émetteur du message	L'ensemble des récepteurs du message	L'acte de communication	Le contenu du message	Un ensemble de champs facultatifs
-----------------------	--------------------------------------	-------------------------	-----------------------	-----------------------------------

Nom de l'agent qui va envoyer le message

Un message peut être envoyé à plusieurs agents simultanément

Représente le but de l'envoi du message en cours

L'information à envoyer

➤ **REQUEST (requête) :**

L'agent émetteur demande à l'agent récepteur d'exécuter une action

➤ **INFORM (informer) :**

L'agent émetteur informe l'agent récepteur à propos d'un fait

➤ **PROPOSE ou CFP (Call for Proposals – Appel d'offre):**

L'agent émetteur désire rentrer en négociation avec l'agent récepteur

Envoi d'un message

L'envoi d'un message se fait par les instructions suivantes:


//1. Déclarer une instance de la classe **ACLMessage** et définir l'acte de communication


ACLMessage message = new ACLMessage(ACLMessage.INFORM);

//2. Ajouter le nom de l'agent récepteur


message.addReceiver(new AID(NomAgentRecepteur, AID.ISLOCALNAME));

//3. Définir le contenu du message


message.setContent("TP ALDI a commencé");
message.setContentObject("TP ALDI a commencé");

//4. Envoyer le message


send(message);

Consultation de la boite aux lettres

La consultation de la boite aux lettres se fait par les instructions suivantes:

//1. Déclarer une instance de la classe **ACLMessage** et appeler la méthode
// **receive()** qui permet de récupérer le **1^{er} message de la boite aux lettres**,
// elle retourne **null** si la boite aux lettres est **vide**

```
ACLMessage msgRecu = receive();
```

//2. Traiter le message s'il est différent de **null**

```
if (msgRecu != null) {
```

```
    //2.1. Récupérer le contenu du message
```

```
    String messContenu = msgRecu.getContent();
```

```
ou String messContenu = (String)msgRecu.getContentObject();
```

```
    //2.2. Traiter le message
```

```
    .....
```

```
}
```

Envoi d'une réponse à un agent émetteur

L'envoi d'une réponse à un message reçu se fait les instructions suivantes:

//1. Consulter la boîte aux lettres

```
ACLMessage msgRecu = receive();
```

//2. Si le message est différent de **null**

```
if (msgRecu != null) {
```

//2.1. Déclarer une instance de la classe **ACLMessage**

```
ACLMessage msgEnvoi = new ACLMessage(ACLMessage.INFORM);
```

//2.2. Récupérer l'identité de l'agent émetteur et l'ajouter au message à envoyer

```
msgEnvoi.addReceiver(msgRecu.getSender());
```

//2.3. Définir le contenu du message

```
msgEnvoi.setContent("Merci pour l'information");
```

```
ou msgEnvoi.setContentObject("Merci pour l'information");
```

//2.4. Envoyer le message

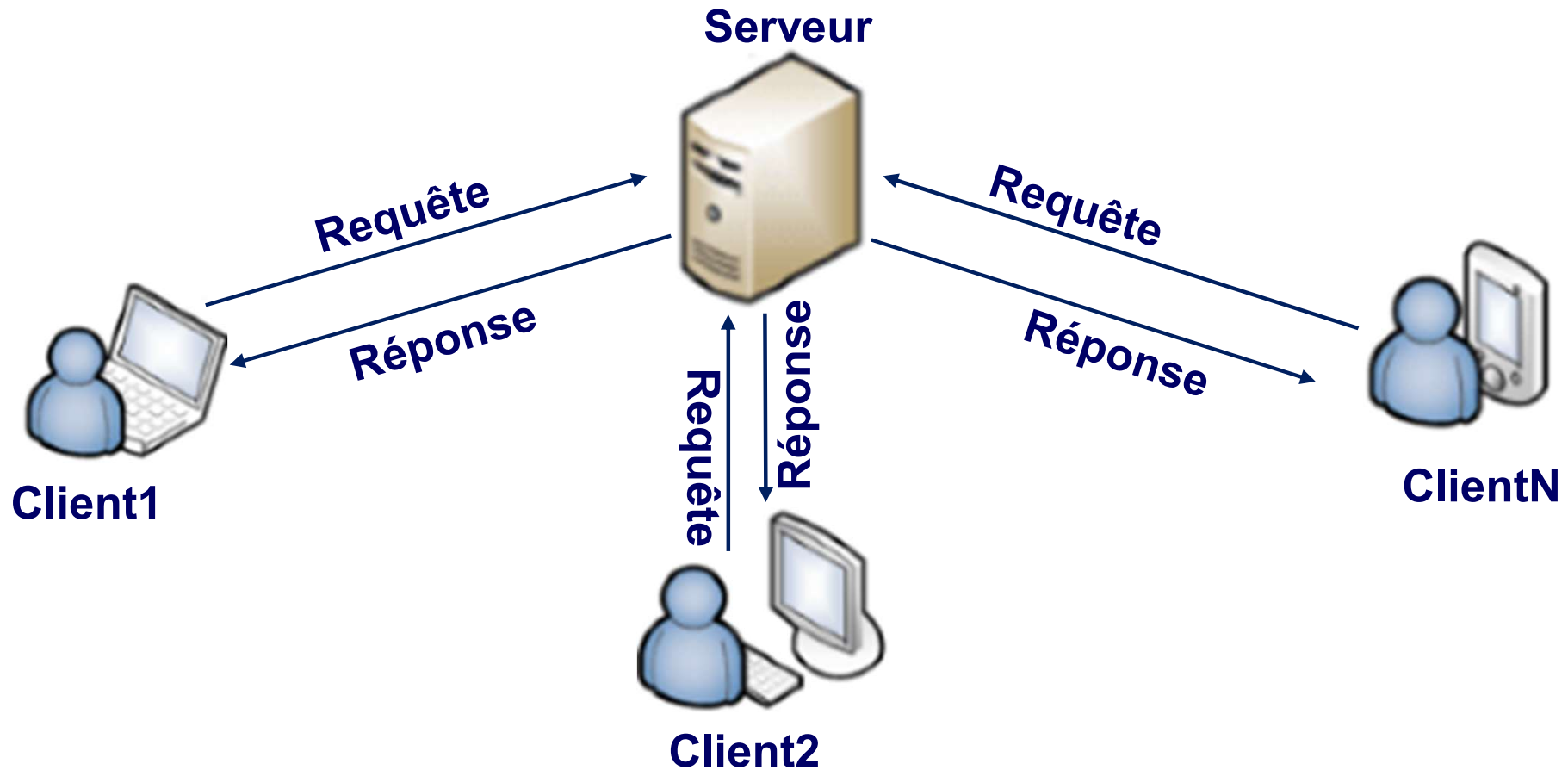
```
send(msgEnvoi);
```

```
}
```


Exemple d'application

On suppose qu'il existe de type d'agents :

- **Serveur** : il reçoit des requêtes et envoie des réponses aux requêtes.
- **Client** : il envoie des requêtes à un Serveur puis attend une réponse.



Exemple



1. Consulter la boîte aux lettres

2. Récupérer le 1^{er} message à partir de la boîte aux lettres

3. Saisir la réponse et l'envoyer au Client



1. Récupérer les paramètres à partir de la liste d'arguments

2. Saisir un message et l'envoyer au Serveur

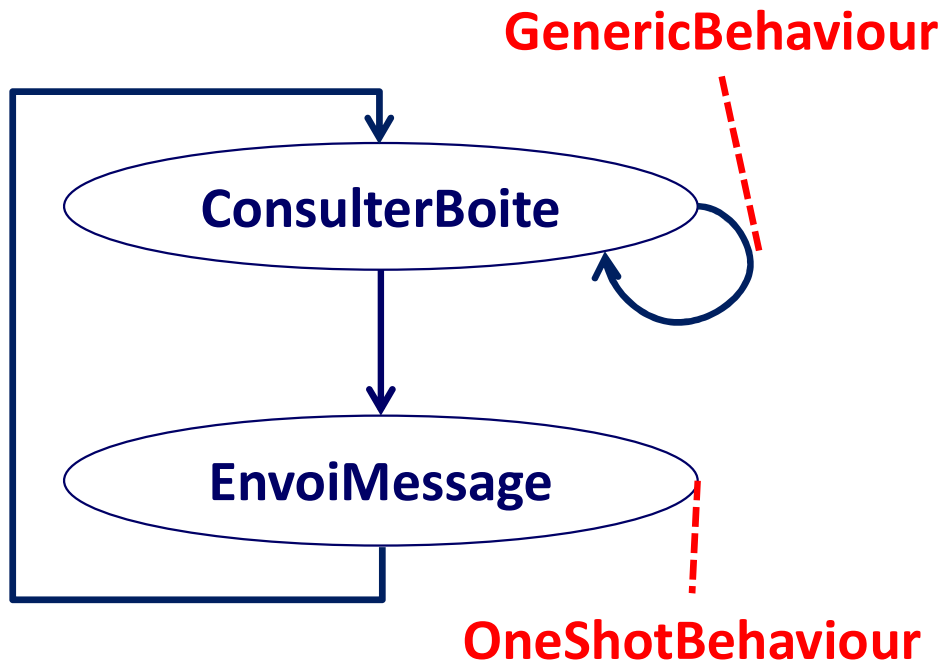
3. Consulter la boîte aux lettres pour récupérer la réponse

4. Récupérer la réponse et l'afficher

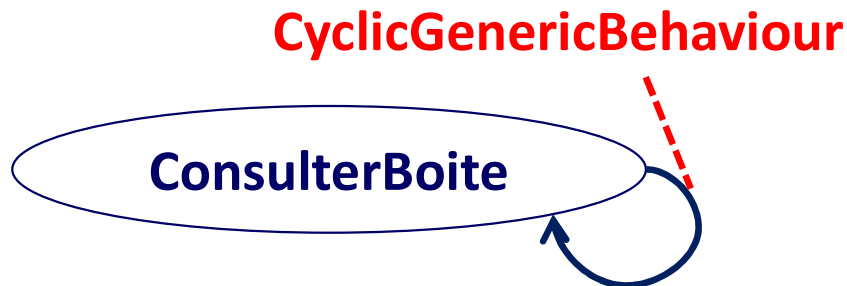
Implémentation de l'exemple

Modélisation des comportements

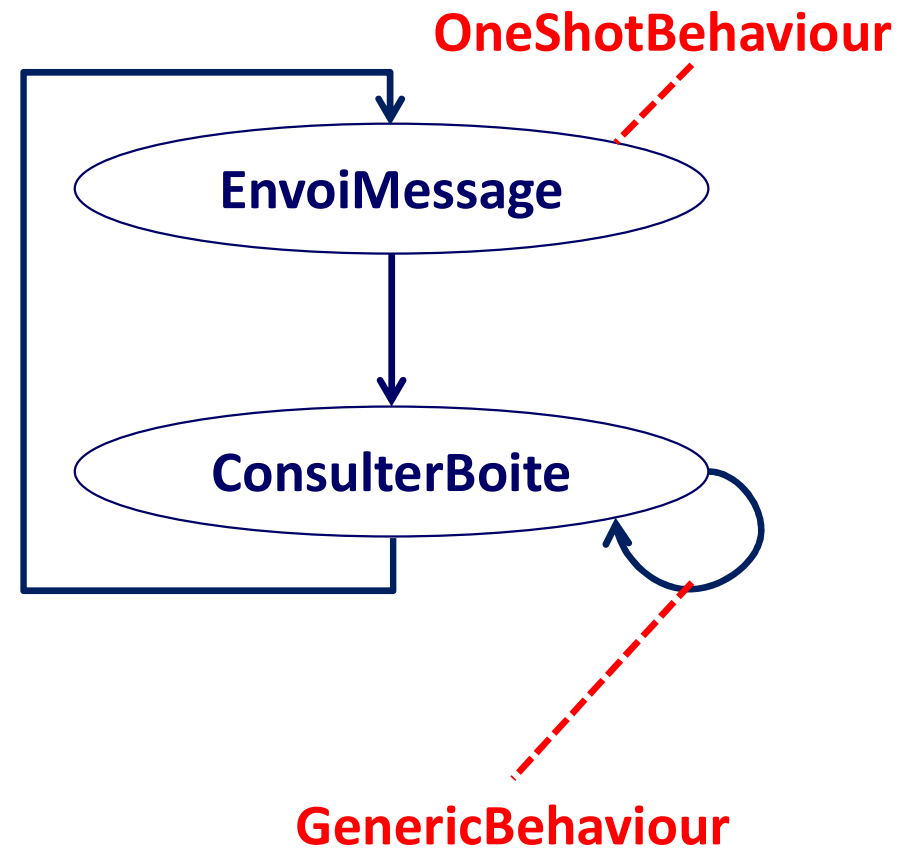
Serveur



Ou bien



Client



Implémentation de l'exemple

```
import java.util.Scanner;
import jade.core.*;
import jade.core.behaviours.OneShotBehaviour;
import jade.lang.acl.ACLMessage;

public class Client extends Agent{
    String nomServeur;
    public void setup() {
        System.out.println("Je suis l'agent "+getLocalName());
        Object [] args = this.getArguments();
        if (args != null) {
            nomServeur = args[0].toString();
            addBehaviour(new EnvoiMessage());
        }
    } //setup
    public class EnvoiMessage extends OneShotBehaviour{
        public void action() {
            Scanner sc = new Scanner(System.in);
            System.out.println("Agent "+getLocalName()+" Veuillez saisir un texte ");
            String contenu = sc.nextLine();
            ACLMessage msgEnvoi = new ACLMessage(ACLMessage.INFORM);
            msgEnvoi.setContent(contenu);
            msgEnvoi.addReceiver(new AID(nomServeur,AID.ISLOCALNAME));
            send(msgEnvoi);
            System.out.println("Agent "+getLocalName()+" j'ai envoye "+contenu+" à l'agent "+nomServeur);
            //addBehaviour(new ConsulterBoite());
        }
    }
}
```

Implémentation de l'exemple

```
import jade.core.Agent;
import jade.core.behaviours.*;
import jade.lang.acl.ACLMessage;

public class Serveur extends Agent{
    public void setup() {
        System.out.println("Je suis l'agent "+getLocalName());
        addBehaviour(new ConsulterBoite());
    }//methode setup
    public class ConsulterBoite extends CyclicBehaviour{
        public void action() {
            ACLMessage msgRecu = receive();
            if (msgRecu != null) {
                String contenu;
                String nomEmetteur;
                contenu = msgRecu.getContent();
                nomEmetteur = msgRecu.getSender().getLocalName();
                System.out.println("Agent "+getLocalName()+" j'ai reçu "+contenu+ " de la part de l'agent "+nomEmetteur);
            }
        }//methode action
    }//Comportement ConsulterBoite
}//Class Serveur
```

Implémentation de l'exemple

```
public class Test {  
    public static void main(String[] args) {  
        String [] commande = new String[3];  
  
        String argument = "";  
        argument = argument+"Serveur1:Serveur";  
        argument = argument+";Client1:Client(Serveur1)";  
        //argument = argument+";Client2:Client(Serveur1)";  
  
        commande [0]="-cp";  
        commande [1]="jade.boot";  
        commande [2]= argument;  
        jade.Boot.main(commande);  
    }  
}
```


Exécution partielle de l'exemple

Executer le programme en lançant :

- 1 Serveur et 1 Client

```
Retrieving CommandDispatcher for platform null
Nov 11, 2023 12:52:40 AM jade.imtp.leap.LEAPIMTPManager initialize
INFO: Listening for intra-platform commands on address:
- jicp://My-Dell:1099

Nov 11, 2023 12:52:40 AM jade.core.BaseService init
INFO: Service jade.core.management.AgentManagement initialized
Nov 11, 2023 12:52:40 AM jade.core.BaseService init
INFO: Service jade.core.messaging.Messaging initialized
Nov 11, 2023 12:52:40 AM jade.core.BaseService init
INFO: Service jade.core.mobility.AgentMobility initialized
Nov 11, 2023 12:52:40 AM jade.core.BaseService init
INFO: Service jade.core.event.Notification initialized
Nov 11, 2023 12:52:40 AM jade.core.messaging.MessagingService clearCachedSlice
INFO: Clearing cache
Nov 11, 2023 12:52:40 AM jade.mtp.http.HTTPServer <init>
INFO: HTTP-MTP Using XML parser com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl$JAXPSAXParser
Nov 11, 2023 12:52:40 AM jade.core.messaging.MessagingService boot
INFO: MTP addresses:
http://My-Dell:7778/acc
Je suis l'agent Client1
Je suis l'agent Serveur1
Nov 11, 2023 12:52:41 AM jade.core.AgentContainerImpl joinPlatform
INFO: -----
Agent container Main-Container@My-Dell is ready.
-----
Agent Client1 Veuillez saisir un texte
TP ALDI a commencé
Agent Client1 j'ai envoye TP ALDI a commenc   l'agent Serveur1
Agent Serveur1 j'ai reçu TP ALDI a commenc   de la part de l'agent Client1
```

Travail demandé

1. Compléter l'implémentation des classes **Client** et **Serveur**
2. Executer le programme en lançant :
 - 1 Serveur et 1 Client
 - 1 Serveur et 2 Clients
 - 1 Serveur et 3 Clients
 - 1 Serveur et 4 Clients