Université Constantine 2

Faculté des NTIC

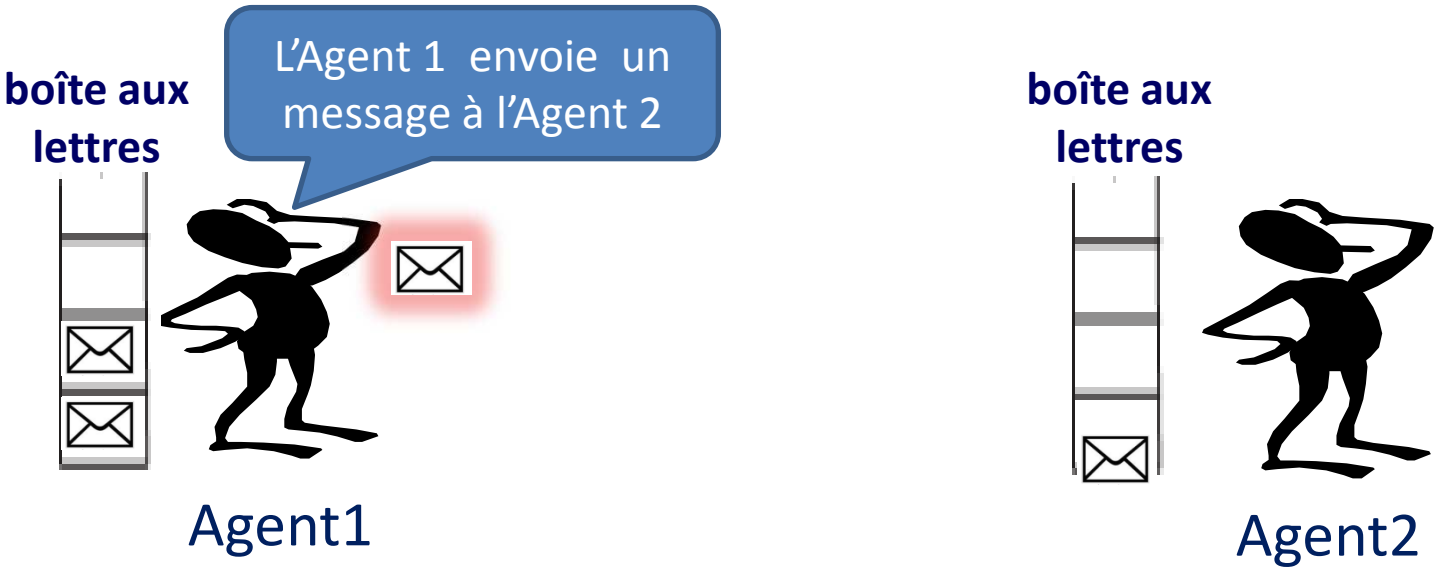
Département d'Informatique Fondamentale et ses Applications

1ère Année Master Réseaux et Systèmes Distribués
TP Algorithmes Distribués (ALDI)

TP N°03 : Initiation à JADE
Communication entre les agents

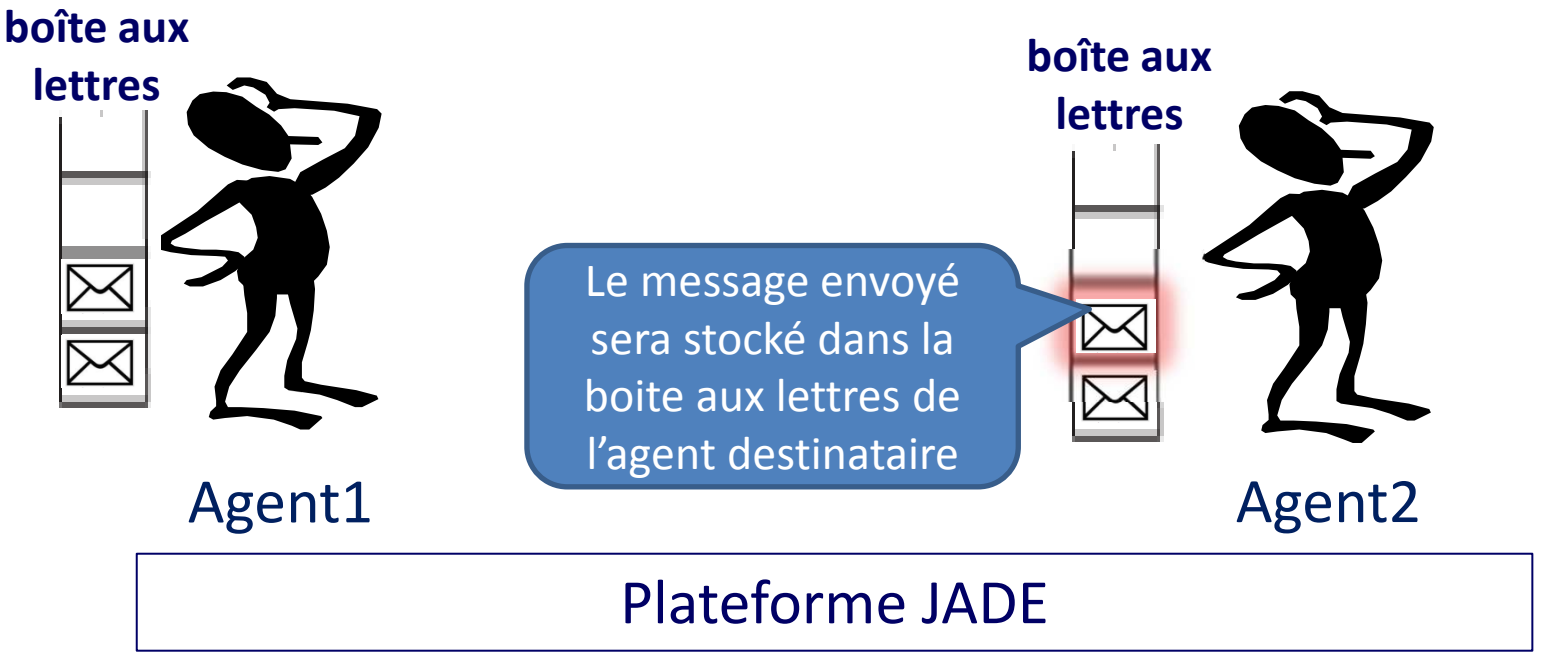
Introduction

- ✓ 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.



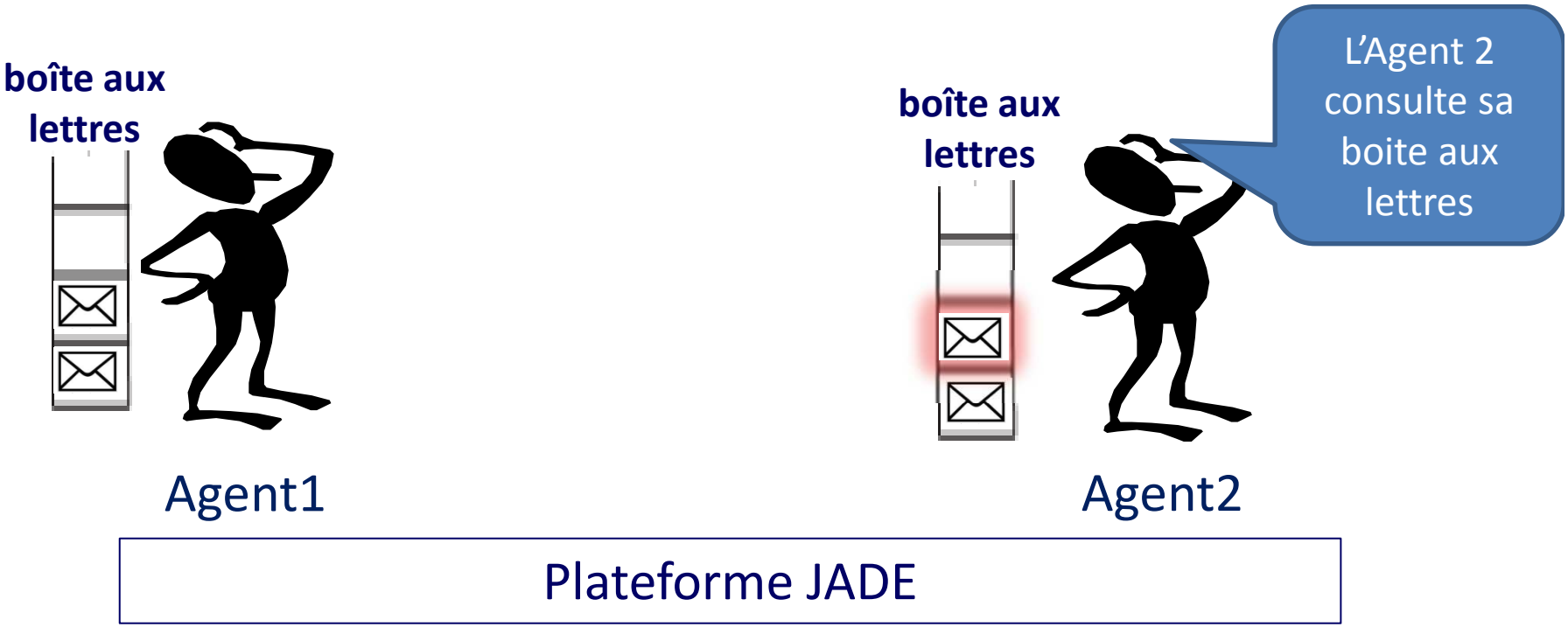
Introduction

- ✓ 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.



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("Agent1", AID.ISLOCALNAME));
```

//3. Définir le contenu du message

```
message.setContent("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 **1er 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();
    //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:

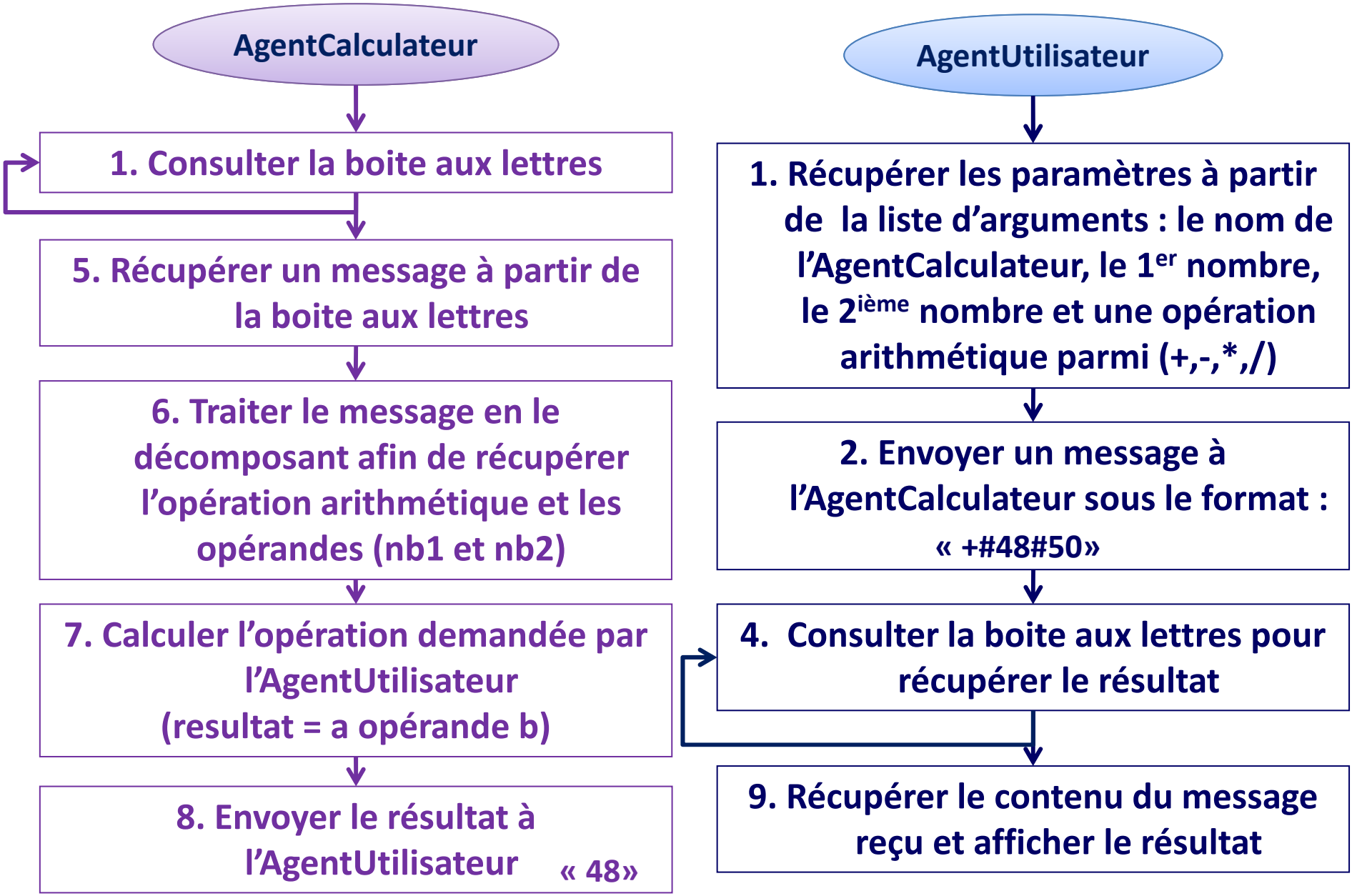
```
//1. Consulter la boite aux lettres
ACLMessage msgRecu = receive();
//2. Si le message est différent de null
if (msgRecu != null) {
  //2.1. Déclarer une instance de la classe ACLMessage
        ACLMessage msgEnvoi = new ACLMessage(ACLMessage.INFORM);
  //2.2. Récupérer l'identité de l'agent émetteur et l'ajouter au message à envoyer
        msgEnvoi.addReceiver(msgRecu.getSender());
  //2.3. Définir le contenu du message
        msgEnvoi.setContent("Merci pour l'information");
  //2.4. Envoyer le message
        send(msgEnvoi );
}
```

Exemple

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

- **AgentCalculateur :** il permet de faire des calculs arithmétiques.
- **AgentUtilisateur** : il demande à un **AgentCalculateur** de lui faire des calculs arithmétique puis il récupère le résultat.

Exemple


AgentCalculateur
1. Consulter la boite aux lettres
5. Récupérer un message à partir de la boite aux lettres
6. Traiter le message en le décomposant afin de récupérer l'opération arithmétique et les opérandes (nb1 et nb2)
7. Calculer l'opération demandée par l'AgentUtilisateur (resultat = a opérande b)
8. Envoyer le résultat à l'AgentUtilisateur « 48»
AgentUtilisateur
1. Récupérer les paramètres à partir de la liste d'arguments : le nom de l'AgentCalculateur, le 1er nombre, le 2ième nombre et une opération arithmétique parmi (+,-,*,/)
2. Envoyer un message à l'AgentCalculateur sous le format : « +#48#50»
4. Consulter la boite aux lettres pour récupérer le résultat
9. Récupérer le contenu du message reçu et afficher le résultat

Implémentation de l'exemple

```
public class AgentUtilisateur extends Agent{
  String nom_agent, op;
  int nb1, nb2;
  protected void setup(){
    System.out.println("Je suis l'agent : "+getLocalName());
    //Récupérer la liste des arguments
    Object[] args = getArguments();
    if (args != null) {
      nom_agent=args[0].toString();
      nb1=Integer.parseInt(args[1].toString());
      nb2=Integer.parseInt(args[2].toString());
      op=args[3].toString();
    }
    //Ajouter le comportement
    SequentialBehaviour ComportSeq = new SequentialBehaviour();
    ComportSeq.addSubBehaviour(new Envoi());
    ComportSeq.addSubBehaviour(new ConsulterBoiteAuxLettres());
    addBehaviour(ComportSeq);
  }//setup
```

cp jade.boot calcul:AgentCalculateur;
P:AgentUtilisateur(calcul,48,50,+)

Implémentation de l'exemple

```
public class Envoi extends OneShotBehaviour{
  public void action(){
    //Rédiger le contenu du message à envoyer
    String  msgContenu=op+"#"+nb1+"#"+nb2;
    //Déclarer une instance de la classe ACLMessage et définir l'acte de communication
    ACLMessage message = new ACLMessage(ACLMessage.REQUEST);
    //Ajouter le nom de l'agent récepteur
    message.addReceiver(new AID(nom_agent, AID.ISLOCALNAME));
    //Ajouter le contenu du message
    message.setContent(msgContenu);
    //Envoyer le message
    send(message);
    System.out.println("Je suis l'agent : "+getLocalName()+" : j'ai envoyé le message :
    "+msgContenu+" à l'agent "+nom_agent);
  } // action
} // Comportement Envoi
```

Implémentation de l'exemple

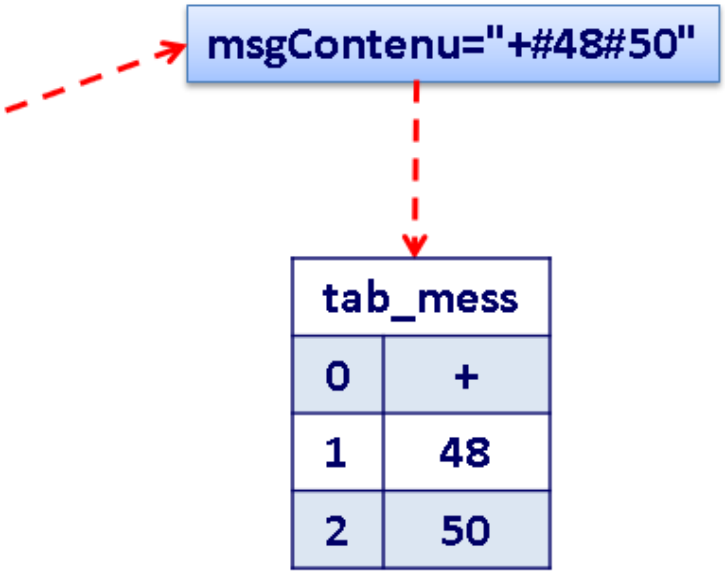
```
public class ConsulterBoiteAuxLettres extends Behaviour{
  ACLMessage msgRecu = receive();
  public void action(){
    msgRecu = receive(); //Appeler la méthode receive() qui permet de récupérer le
    if (msgRecu != null){// 1er message de la boite aux lettres
      //Récupérer le contenu du message
      double resultat = Double.parseDouble(msgRecu.getContent());
      System.out.println("Je suis l'agent : "+getLocalName()+" : j'ai reçu le message : "+
      resultat+" de la part de : "+msgRecu.getSender().getLocalName());
    } //if
  } //méthode action
  public boolean done(){
    return msgRecu != null;
  } //done
} // comportement ConsulterBoiteAuxLettres
} //class AgentUtilisateur
```

Implémentation de l'exemple

```
import jade.lang.acl.ACLMessage;
import jade.core.behaviours.*;
import jade.core.*;
public class AgentCalculateur extends Agent{
  protected void setup(){
    System.out.println("Je suis l'agent : "+getLocalName());
    addBehaviour(new calculer());
  }
  public class calculer extends CyclicBehaviour{
    public void action(){
      //Déclarer une instance de la classe ACLMessage  et consulter la boite aux lettres
      ACLMessage msgRecu = receive();
      if (msgRecu != null){
        //Récupérer le contenu du message
```

Implémentation de l'exemple

```
String msgContenu  =  msgRecu.getContent();
System.out.println("Je suis l'agent : "+getLocalName()+" : j'ai reçu le message : "+
msgRecu.getContent()+" de la part de : "+msgRecu.getSender().getLocalName());
//Utiliser la méthode split pour extraire les différentes parties du message reçu
//Le critère d'extraction se base sur la recherche du caractère "#" qui sépare les
//différentes parties du contenu du message reçu et seront sauvegardées dans un
//tableau tab_mess
String[]  tab_mess  =  msgContenu.split("#");
//Récupération de l'opération
String operation = tab_mess[0];
//Récupération de la 1ère opérande
int  a = Integer.parseInt(tab_mess[1]);
//Récupération de la 2ème opérande
int  b = Integer.parseInt(tab_mess[2]);
```



Implémentation de l'exemple

```
//Effectuer l'opération arithmétique demandée par l'AgentUtilisateur
double resultat = 0;
if (operation.equals("+")){
    resultat = a + b;
   System.out.println("Agent:"+getLocalName()+":j'ai calculé:"+a+"+"+b+"="+ resultat );
}
if (operation.equals("-")){
    resultat = a - b;
   System.out.println("Aagent:"+getLocalName()+": j'ai calculé : "+a+"-"+b+"="+ resultat );
}
if (operation.equals("*")){
    resultat = a * b;
   System.out.println("Agent : "+getLocalName()+" : j'ai calculé : "+a+"*"+b+"="+ resultat )
}
if (operation.equals("/")){
   if (b != 0) resultat = a / (double) b;
   System.out.println("Agent : "+getLocalName()+" : j'ai calculé : "+a+"/"+b+"="+ resultat );
}
```

Implémentation de l'exemple

```
//Envoyer le résultat à l'agent concerné
//Déclarer une instance de la classe ACLMessage
ACLMessage message = new ACLMessage(ACLMessage.INFORM);
//Récupérer l'identité de l'agent émetteur
message.addReceiver(msgRecu.getSender());
//Ajouter le contenu du message (on envoie le contenu de la variable resultat)
message.setContent(""+ resultat );
//Envoyer le message
send(message);
System.out.println("Agent : "+getLocalName()+" : j'ai envoyé un message : " +
resultat + " à l'agent "+msgRecu.getSender().getLocalName().toString());
}//if (msgRecu != null)
}// méthode action
}// comportement calculer
}//class AgentCalculateur
```

Exécution de l'exemple

Tester l'exemple en utilisant la classe Test suivante :

```
public class Test {
    public static void main(String[]  args) {
            String [] commande = new String[3];
            String argument ="";
            argument = argument+"calcul:AgentCalculateur";
            argument = argument+";";
            argument = argument+"P1:AgentUtilisateur(calcul,48,50,+)";
            //argument = argument+";";
            //argument = argument+"P2:AgentUtilisateur(calcul,50,5,/)";
            commande [0]="-cp";
            commande [1]="jade.boot";
            commande [2]= argument;
            jade.Boot.main(commande);
    }
}
```