Universidade do Minho
Escola de Engenharia
Departamento de Informática

Mestrado Integrado em Engenharia Informática
Mestrado em Engenharia Informática
Agentes Inteligentes
2020/2021

Filipe Gonçalves, Paulo Novais, César Analide

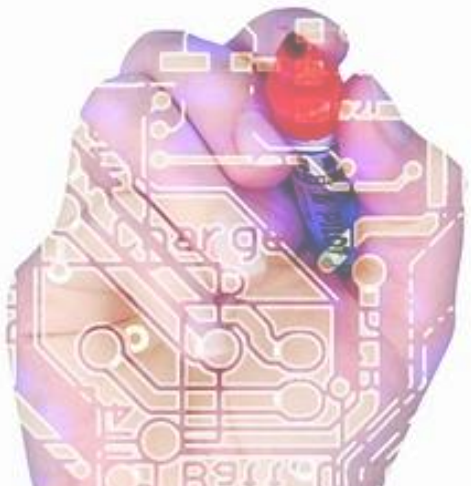
ISLab
Synthetic Intelligence Lab

Agentes Inteligentes @ 2020/2021

- Paulo Novais – pjon@di.uminho.pt
- César Analide – analide@di.uminho.pt
- Filipe Gonçalves – fgoncalves@algoritmi.uminho.pt

- Departamento de Informática
  Escola de Engenharia
  Universidade do Minho
- ISLab – (Synthetic Intelligence Lab)
- Centro ALGORITMI
  Universidade do Minho

JADE – Performatives



Performatives

```
                ACLMessage msg1 =new ACLMessage(ACLMessage.);
                msg1.setContent("ping");
                msg1.setConversationId(""+time);
                msg1.addReceiver(receiver);
                myAgent.send(msg1);
            }else{
                ACLMessage msg2 =new ACLMessage(ACLMessage.I
                msg2.setContent("not ping");
                msg2.setConversationId(""+time);
                msg2.addReceiver(receiver);
                myAgent.send(msg2);
            ACLMessage msg =new ACLMessage(ACLMessage.INFORM
            }

            block(1000);

        }
    }

}
```

```
ACCEPT_PROPOSAL : int - ACLMessage
AGREE : int - ACLMessage
CANCEL : int - ACLMessage
CFP : int - ACLMessage
CONFIRM : int - ACLMessage
DISCONFIRM : int - ACLMessage
FAILURE : int - ACLMessage
INFORM : int - ACLMessage
INFORM_IF : int - ACLMessage
INFORM_REF : int - ACLMessage
NOT_UNDERSTOOD : int - ACLMessage
PROPAGATE : int - ACLMessage
Press 'Ctrl+Space' to show Template Proposals
```

INT	*Performative*	INT	*Performative*
0	ACCEPT_PROPOSAL	11	PROPOSE
1	AGREE	12	QUERY-IF
2	CANCEL	13	QUERY-REF
3	CFP	14	REFUSE
4	CONFIRM	15	REJECT_PROPOSAL
5	DISCONFIRM	16	REQUEST
6	FAILURE	17	REQUEST_WHEN
7	INFORM	18	REQUEST_WHENEVER
8	INFORM-IF	19	SUBSCRIBE
9	INFORM-REF	20	PROXY
10	NOT_UNDERSTOOD	21	PROPAGATE

Agentes Inteligentes @ 2020/2021

```java
public class SenderAsk extends Agent{

    @Override
    protected void setup(){

        super.setup();

        this.addBehaviour(new ReceiveBehaviour());
        this.addBehaviour(new SendMessage(this,2000));


    }

    public class SendMessage extends TickerBehaviour {

        public SendMessage(Agent a, long timeout){

            super(a,timeout);
        }

        @Override
        public void onTick(){

            AID receiver=new AID();
            receiver.setLocalName("pingaponga");
            long time =System.currentTimeMillis();


            if(time % 2 == 0){
                ACLMessage msg1 =new ACLMessage(ACLMessage.PROPOSE);
                msg1.setContent("Let's play ping pong?");
                msg1.setConversationId(""+time);
                msg1.addReceiver(receiver);
                myAgent.send(msg1);
            }else{
                ACLMessage msg2 =new ACLMessage(ACLMessage.INFORM);
                msg2.setContent("Let's play ping pong.");
                msg2.setConversationId(""+time);
                msg2.addReceiver(receiver);
                myAgent.send(msg2);
            }

            block(1000);

        }
    }
```

Performatives

```java
public class ReceiveBehaviour extends CyclicBehaviour {

    @Override
    public void action(){
        ACLMessage msg =receive();
        if(msg != null){

            if(msg.getPerformative()==0){

                System.out.println("Recebi uma mensagem de "+msg.getSender()+".Conteúdo: Ok! Let's play ping pong.");


            }else System.out.println("Recebi uma mensagem de "+msg.getSender()+".Conteúdo: The offer to play ping pong was not accepted.");

        }


        block();
    }
}
```

0: ACCEPT_PROPOSAL

- New Behaviour in pingaponga agent.

```
public class ReceiveBehaviourProposal extends CyclicBehaviour {

    @Override
    public void action(){
        ACLMessage msg =receive();
        if(msg != null){
            ACLMessage resp=msg.createReply();
            if(msg.getPerformative()==11){

                System.out.println("Recebi uma mensagem de "+msg.getSender()+".Conteúdo: "+msg.getContent());

                resp.setContent("Yes");
                resp.setPerformative(ACLMessage.ACCEPT_PROPOSAL);

            }else{
                System.out.println("Recebi uma mensagem de "+msg.getSender()+".Conteúdo: "+msg.getContent());

                resp.setContent("No");
                resp.setPerformative(ACLMessage.NOT_UNDERSTOOD);

            }

            send(resp);

        }

        block();
    }
}
```

11: PROPOSE

- Configurations:
  - -container -agents pingaponga:agents.PingPong;sender:agents.SenderAsk

```
pingaponga a começar!
Recebi uma mensagem de ( agent-identifier :name sender@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Let's play ping pong.
Recebi uma mensagem de ( agent-identifier :name pingaponga@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: The offer to play ping pong was
Recebi uma mensagem de ( agent-identifier :name sender@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Let's play ping pong.
Recebi uma mensagem de ( agent-identifier :name pingaponga@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: The offer to play ping pong was
Recebi uma mensagem de ( agent-identifier :name sender@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Let's play ping pong?
Recebi uma mensagem de ( agent-identifier :name pingaponga@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Ok! Let's play ping pong.
Recebi uma mensagem de ( agent-identifier :name sender@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Let's play ping pong?
Recebi uma mensagem de ( agent-identifier :name pingaponga@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Ok! Let's play ping pong.
Recebi uma mensagem de ( agent-identifier :name sender@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Let's play ping pong.
Recebi uma mensagem de ( agent-identifier :name pingaponga@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: The offer to play ping pong was
Recebi uma mensagem de ( agent-identifier :name sender@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Let's play ping pong?
Recebi uma mensagem de ( agent-identifier :name pingaponga@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Ok! Let's play ping pong.
Recebi uma mensagem de ( agent-identifier :name sender@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Let's play ping pong?
Recebi uma mensagem de ( agent-identifier :name pingaponga@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Ok! Let's play ping pong.
Recebi uma mensagem de ( agent-identifier :name sender@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Let's play ping pong?
Recebi uma mensagem de ( agent-identifier :name pingaponga@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Ok! Let's play ping pong.
```

ISLab
Synthetic Intelligence Lab

Agentes Inteligentes @ 2020/2021

- Filter Messages from Message Pool

```
public class SenderTestFilter extends Agent {


    @Override
    protected void setup(){

        super.setup();

        this.addBehaviour(new SendMessage(this,2000));

    }

    public class SendMessage extends TickerBehaviour {

        public SendMessage(Agent a, long timeout){

            super(a,timeout);
        }
```

```
    public class SendMessage extends TickerBehaviour {

        public SendMessage(Agent a, long timeout){

            super(a,timeout);
        }

        @Override
        public void onTick(){
            AID receiver=new AID();
            receiver.setLocalName("pingaponga");
            long time =System.currentTimeMillis();

            if(time % 2 == 0){
                ACLMessage msg1 =new ACLMessage(ACLMessage.PROPOSE);
                msg1.setOntology("event");
                msg1.setContent("Let's play ping pong.");
                msg1.setConversationId(""+time);
                msg1.addReceiver(receiver);
                myAgent.send(msg1);
            }else{
                ACLMessage msg2 =new ACLMessage(ACLMessage.INFORM);
                msg2.setContent("Let's play ping pong.");
                msg2.setConversationId(""+time);
                msg2.addReceiver(receiver);
                myAgent.send(msg2);
            }

            block(1000);

        }
    }
}
```

The pingaponga agent will only process messages with performative PROPOSE.

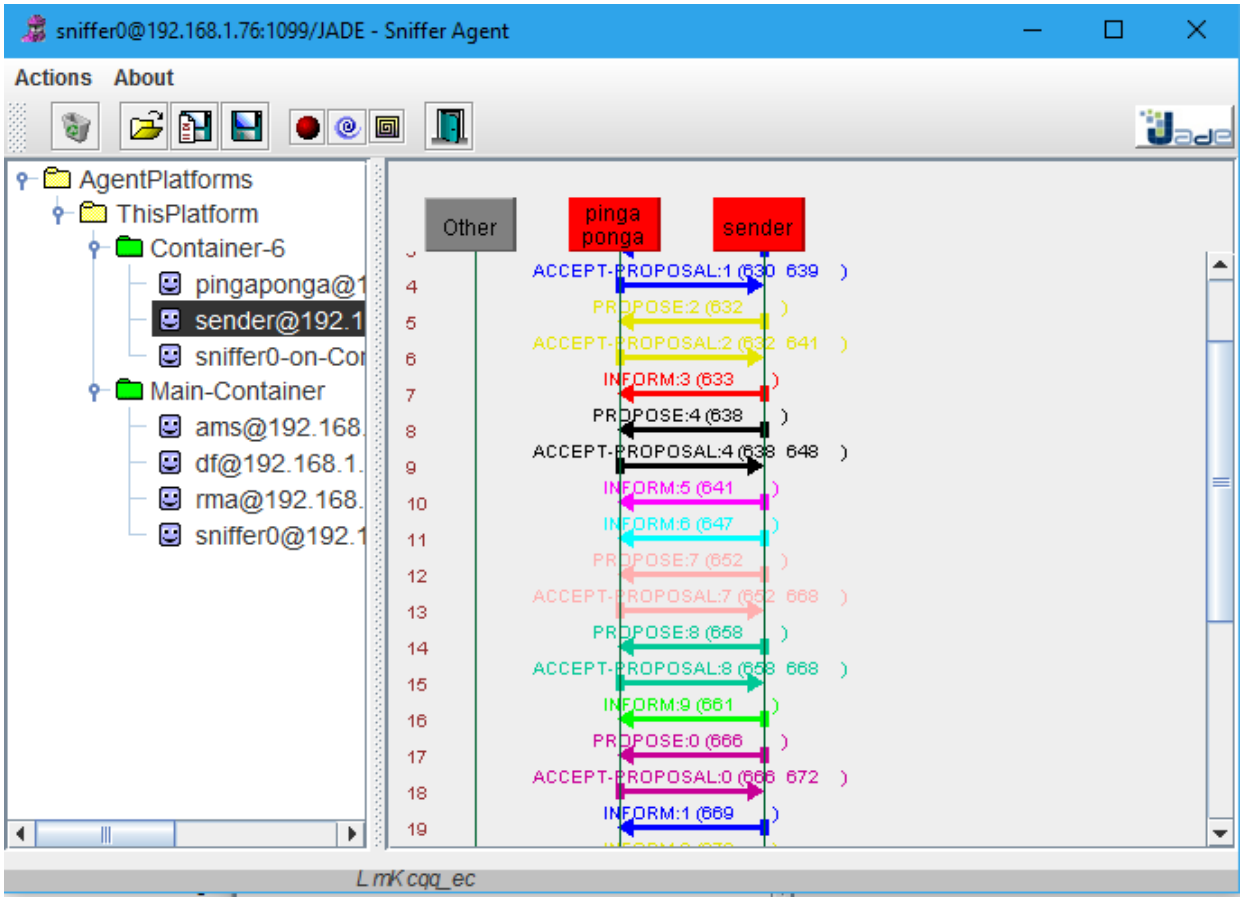
- Filter Messages from Message Pool

```
public class ReceiveBehaviourFilter extends CyclicBehaviour {

    @Override
    public void action(){
        MessageTemplate mt1= MessageTemplate.MatchPerformative(ACLMessage.PROPOSE);
        MessageTemplate mt2= MessageTemplate.MatchOntology("event");
        MessageTemplate mt3= MessageTemplate.and(mt1, mt2);
        ACLMessage msg =receive(mt1);
        if(msg != null){
            ACLMessage resp=msg.createReply();
                System.out.println("Recebi uma mensagem de "+msg.getSender()+".Conteúdo: "+msg.getContent());
                resp.setContent("Yes");
                resp.setPerformative(ACLMessage.ACCEPT_PROPOSAL);
                send(resp);

        }

        block();
    }
}
```

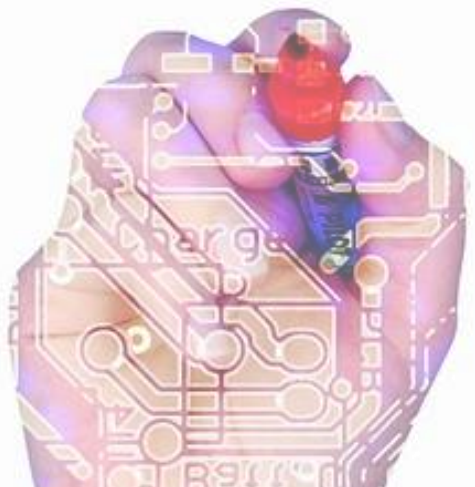
- Configuration:
  - -container -agents pingaponga:agents.PingPong;sender:agents.SenderTestFilter

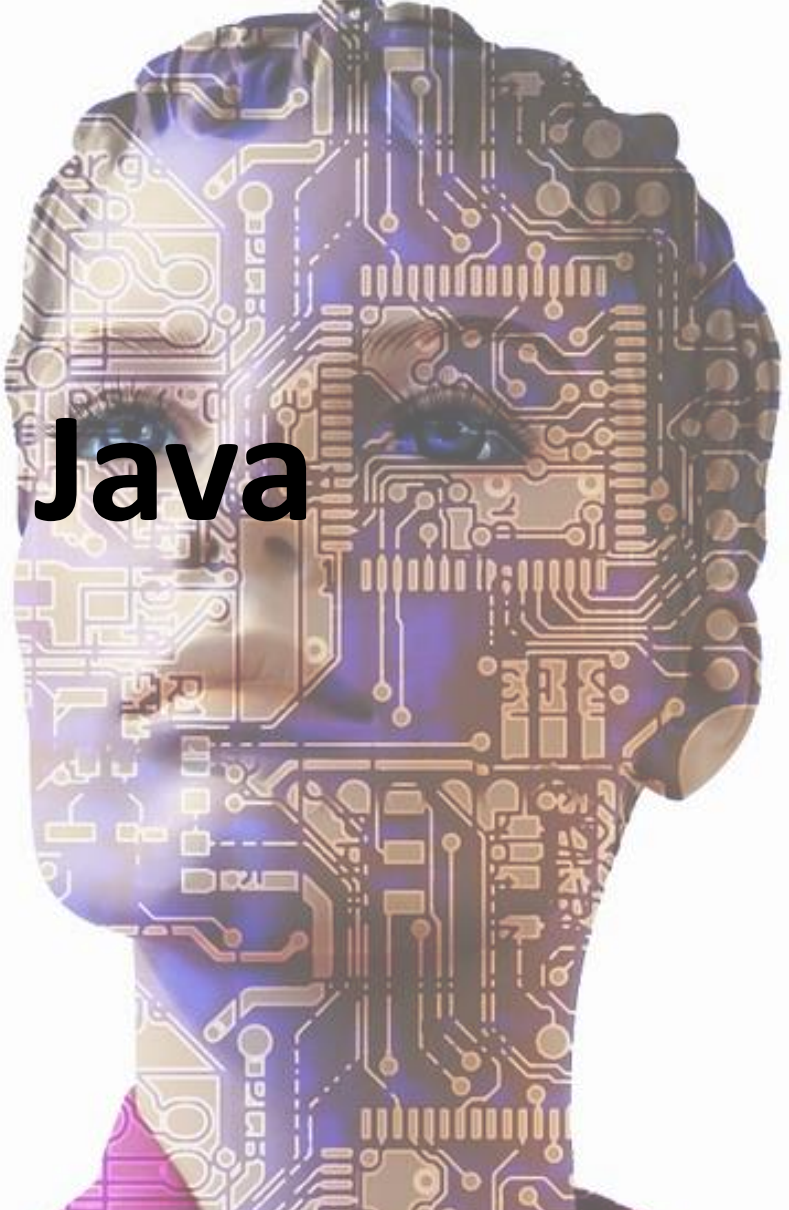
```
pingaponga a começar!
Recebi uma mensagem de ( agent-identifier :name sender@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Let's play ping pong?
Recebi uma mensagem de ( agent-identifier :name sender@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Let's play ping pong?
Recebi uma mensagem de ( agent-identifier :name sender@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Let's play ping pong?
Recebi uma mensagem de ( agent-identifier :name sender@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Let's play ping pong?
Recebi uma mensagem de ( agent-identifier :name sender@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Let's play ping pong?
Recebi uma mensagem de ( agent-identifier :name sender@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Let's play ping pong?
Recebi uma mensagem de ( agent-identifier :name sender@192.168.1.76:1099/JADE  :addresses (sequence http://Laptop.lan:7778/acc )).Conteúdo: Let's play ping pong?
```

- Filter Messages from Message Pool



JADE - Serialized Java Objects





- If agents are to communicate in a way that makes sense for them, they must share the same language, vocabulary and protocols
- JADE already supports a certain degree of commonality
- However, it is required to define your own vocabulary and semantics for the content of the messages exchanged between agents

- JADE provides two ways to implement communication between agents:
  1. (Used until now) Using strings to represent the content of messages
     - convenient when the content of messages is atomic data, but not in the case of abstract concepts, objects or structured data
     - String needs to be parsed for the agent to understand
  2. Transmit *Serialized* Java objects directly as the content of messages
     - convenient method for a local application where all agents are implemented in Java.
     - however, these messages are not readable by humans

- The two types of message content are reflected on the types of message content
- Different methods are used to **set** and **get** content:

Content type	Getting content	Setting content
Strings	getContent()	SetContent()
Java Objects	getContentObject()	SetContentObject()

The Bank example:

- Two agents are created which implement the client and server roles for a bank with savings accounts
- The **BankServerAgent** class acts as a server and the **BankClientAgent** class acts as client
- Each class imports the **Bank Operation** classes which represent the terms that constitute the specific language of the agents

The Bank example:

- Client agent sends a REQUEST message to the server agent, following a simple protocol:
  - create an account
  - make an operation
- The server agent responds either with:
  - an INFORM after processing the request
  - a NOT_UNDERSTOOD if it cannot decode the content of the message
- To query information about a specific account, the client agent sends a QUERY_REF to the server agent which responds with either:
  - an INFORM after processing the query
  - a NOT_UNDERSTOOD if it cannot decode the content of the message

Messages with serialized Java objects:

- If objects are to be sent as parts of messages, it is required to implement the **java.io.Serializable** interface
  - Otherwise the serialization of the content of the messages before sending will fail

- Classes that are used in the Bank application:
  - **Account**: concept of a bank savings account
  - **Operation**: concept of a bank operation
  - **MakeOperation**: action of making an operation such as deposit or withdrawal
  - **OperationList**: concept of the list of last operations
  - **CreateAccount**: action of creating an account
  - **Information**: concept of querying information about an account such as the balance and the list of last operations
  - **Problem**: result of an action that fails

- Sending Messages:

```
class MakeOperation implements java.io.Serializable {
// ----------------------------------------------------

   private String accountId;
   private int type;
   private float amount;

   public String getAccountId() {
      return accountId;
   }

   public int getType() {
      return type;
   }

   public float getAmount() {
     return amount;
   }

   public void setAccountId(String accountId) {
      this.accountId = accountId;
   }

   public void setType(int type) {
      this.type = type;
   }

   public void setAmount(float amount) {
      this.amount = amount;
   }
}
```

```
MakeOperation mo = new MakeOperation();
mo.setAccountId(acc.getId());
mo.setType(command);
mo.setAmount(amount);

ACLMessage msg = new ACLMessage( ACLMessage.REQUEST );
msg.addReceiver(server);
try {
   msg.setContentObject( mo );
}
catch (Exception ex) { ex.printStackTrace(); }
send(msg);
```

- Example: Receiving Messages

```
class ReceiveMessages extends CyclicBehaviour
{
      public ReceiveMessages(Agent a) {

           super(a);
      }

      public void action()
      {
         ACLMessage msg = receive();
         if (msg == null) { block(); return; }

         try {
           Object content = msg.getContentObject();

           switch (msg.getPerformative()) {

           case (ACLMessage.REQUEST):
               if (action instanceof CreateAccount)
                   addBehaviour(new HandleCreateAccount(myAgent, msg));
               else if (content instanceof MakeOperation)
                   addBehaviour(new HandleOperation(myAgent, msg));
               ...
      }
}
```

```
class HandleOperation extends OneShotBehaviour
{
      ACLMessage request;

      public HandleOperation(Agent a, ACLMessage request) {

           super(a);
           this.request = request
      }

      public void action() {

         try {
           Operation op = (Operation)
           request.getContentObject();
           ACLMessage reply = request.createReply();
           // Process the operation
           Object result = processOperation(op);
           ...
         }
         catch (Exception ex) { ex.printStackTrace(); }
      }
}
```

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Mestrado Integrado em Engenharia Informática
Mestrado em Engenharia Informática
Agentes Inteligentes
2020/2021

Filipe Gonçalves, Paulo Novais, César Analide