JADEX Framework Exercise II

Integrated Master's in Informatics Engineering

Intelligent Agents

2017/2018

<u>Synthetic Intelligence Lab</u>

Filipe Gonçalves

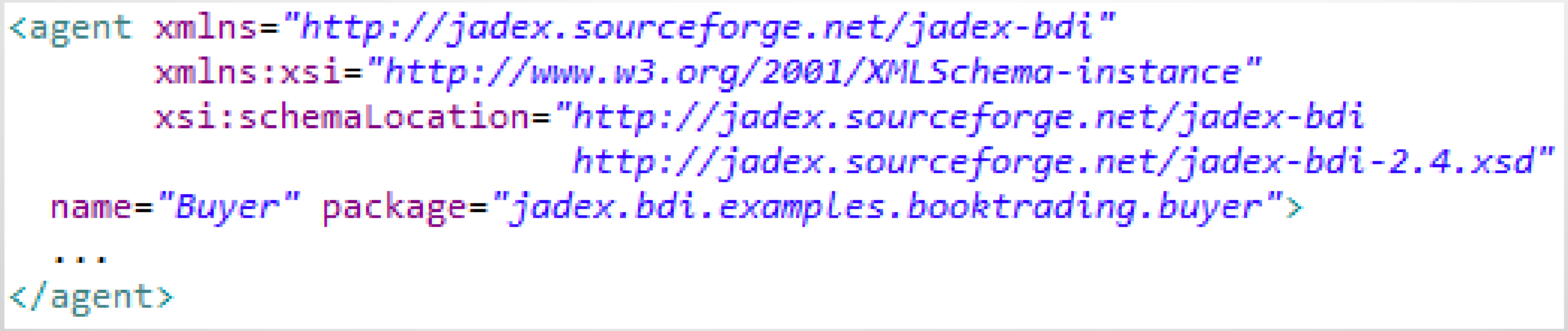
Paulo Novais

Useful Links

- https://sourceforge.net/projects/jadex/files/jadex/2.4
- http://www.agilemethod.csie.ncu.edu.tw/download/agent/tutorial.pdf
- https://download.actoron.com/docs/releases/jadex-2.4/

ADF Structure

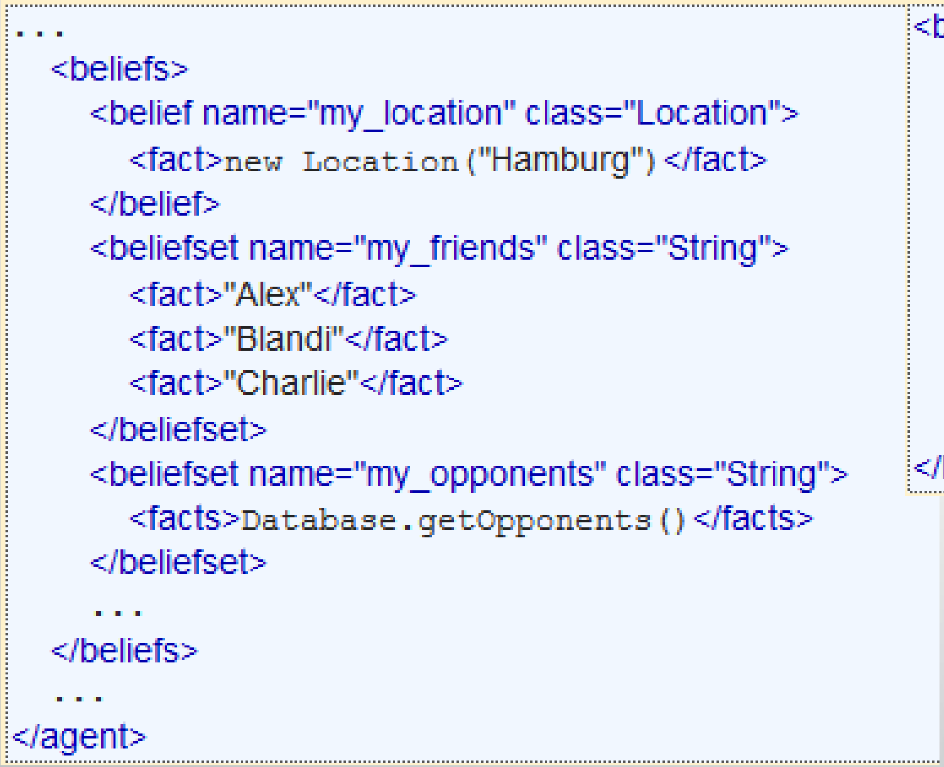
- XML Agent Definition File
- <agent></agent> defines XML file type (Agent head);
- Requires the definition of Agent's name and package (found in Agent body);



```
<agent xmlns="http://jadex.sourceforge.net/jadex-bdi"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://jadex.sourceforge.net/jadex-bdi
                           http://jadex.sourceforge.net/jadex-bdi-2.4.xsd"
  name="Buyer" package="jadex.bdi.examples.booktrading.buyer">
  ...
</agent>
```

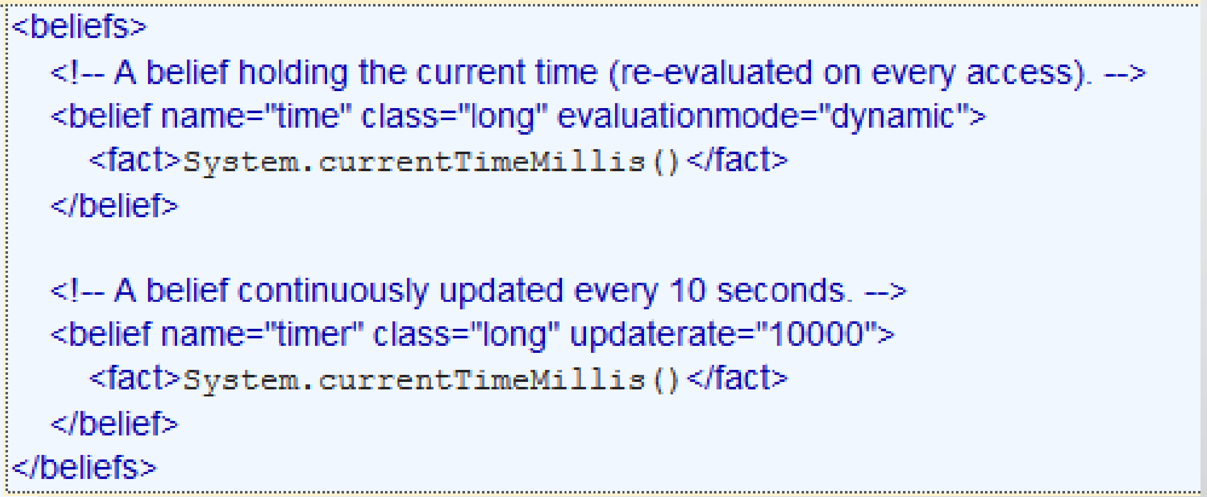
Creating Beliefs

<belief> and **<beliefset>**



```
...
  <beliefs>
    <belief name="my_location" class="Location">
      <fact>new Location("Hamburg")</fact>
    </belief>
    <beliefset name="my_friends" class="String">
      <fact>"Alex"</fact>
      <fact>"Blandi"</fact>
      <fact>"Charlie"</fact>
    </beliefset>
    <beliefset name="my_opponents" class="String">
      <facts>Database.getOpponents()</facts>
    </beliefset>
    ...
  </beliefs>
  ...
</agent>
```

Dynamically evaluated beliefs



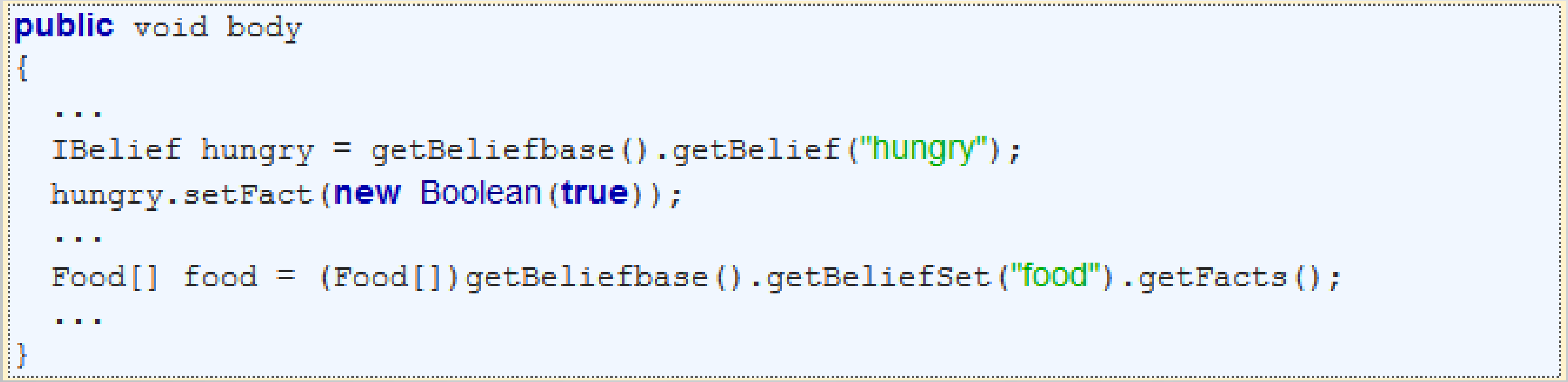
```
<beliefs>
  <!-- A belief holding the current time (re-evaluated on every access). -->
  <belief name="time" class="long" evaluationmode="dynamic">
    <fact>System.currentTimeMillis()</fact>
  </belief>

  <!-- A belief continuously updated every 10 seconds. -->
  <belief name="timer" class="long" updaterate="10000">
    <fact>System.currentTimeMillis()</fact>
  </belief>
</beliefs>
```

Beliefs Access from Plans

IBeliefbase getBeliefbase()

– IBelief getBelief()
  - Object getFact()
  - setFact(Object)

– IBeliefSet getBeliefSet()
  - Object getFact()
  - Object[] getFacts()
  - addFact(Object)
  - addFacts(Object[])
  - removeFact(Object)
  - removeFacts()



```
public void body
{
  ...
  IBelief hungry = getBeliefbase().getBelief("hungry");
  hungry.setFact(new Boolean(true));
  ...
  Food[] food = (Food[])getBeliefbase().getBeliefSet("food").getFacts();
  ...
}
```

Creating Plans Head

- The plan head (in ADF) defines the circumstances under which the plan body is instantiated and executed. The plan body is declared using **<body>**

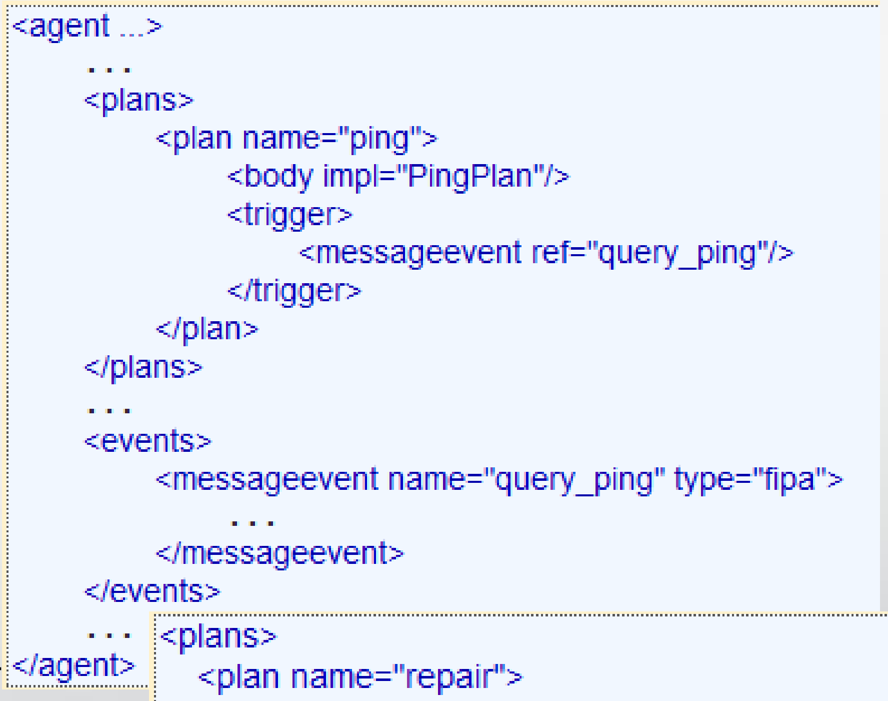
<trigger>

Triggered by goals, internal events, message events:

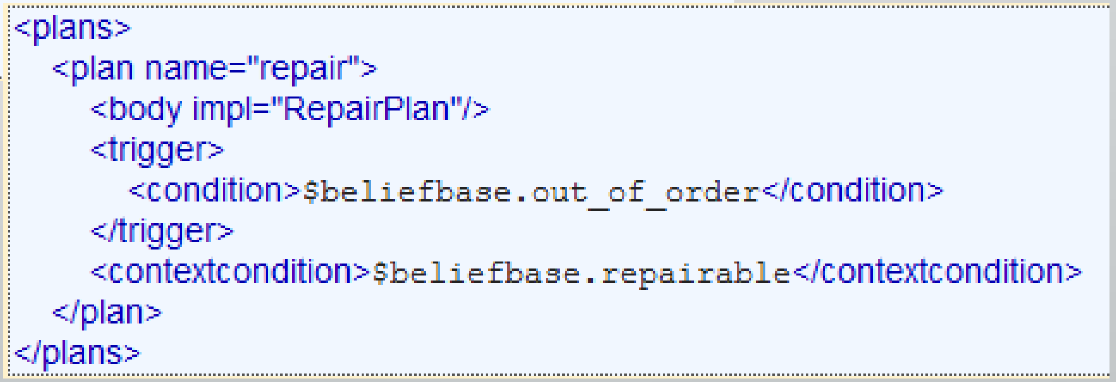
- **<goal>**
- **<goalfinished>**
- **<internalevent>**
- **<messageevent>**

Triggered by the alteration of a fact:

- **<factchanged>**
- **<factadded>**
- **<factremoved>**



```
<agent ...>
    . . .
    <plans>
        <plan name="ping">
            <body impl="PingPlan"/>
            <trigger>
                <messageevent ref="query_ping"/>
            </trigger>
        </plan>
    </plans>
    . . .
    <events>
        <messageevent name="query_ping" type="fipa">
            . . .
        </messageevent>
    </events>
    . . .
</agent>
```



```
<plans>
  <plan name="repair">
    <body impl="RepairPlan"/>
    <trigger>
      <condition>$beliefbase.out_of_order</condition>
    </trigger>
    <contextcondition>$beliefbase.repairable</contextcondition>
  </plan>
</plans>
```

Creating Plans Body

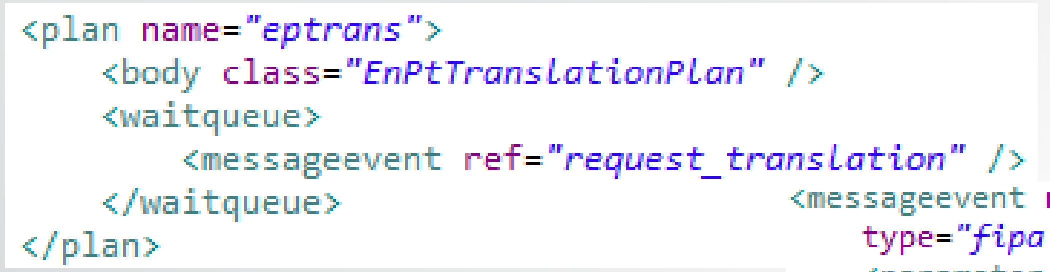
Standard plans inherit from jadex.bdi.runtime.Plan

- **body()**

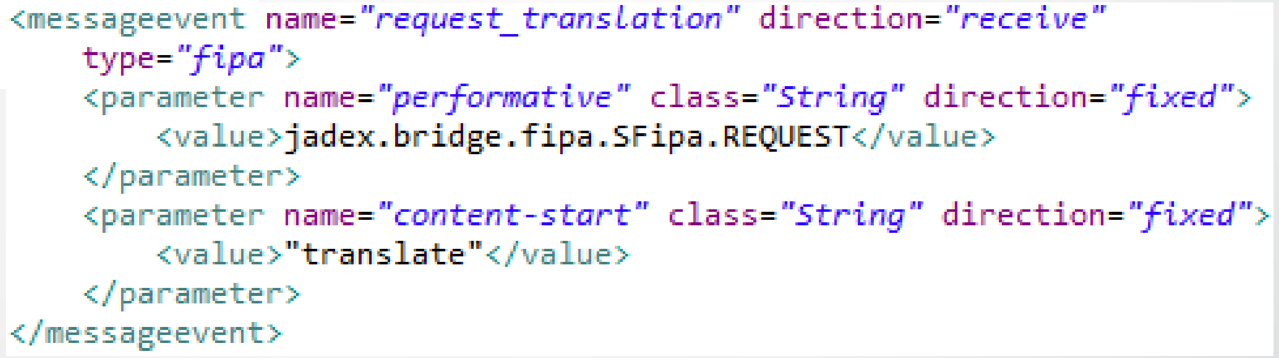
```
public class MyPlan extends Plan
{
  public void body()
  {
    // Application code goes here.
    ...
  }

  public void passed()
  {
    // Clean-up code for plan success.
    ...
  }

  public void failed()
  {
    // Clean-up code for plan failure.
    ...
    getException().printStackTrace();
  }

  public void aborted()
  {
    // Clean-up code for an aborted plan.
    ...
    System.out.println("Goal achieved? "+isAbortedOnSuccess());
  }
}
```

```
public void body()
{
  // Send request.
  ...

  // Wait for agree/refuse.
  IMessageEvent e1 = waitForMessageEvent(...);
  boolean agreed = ...;
  ...

  // Wait for inform/failure.
  if(agreed)
  {
    IMessageEvent e2 = waitForReply(...);
    boolean informed = ...;

    ...
    if(informed)
    {
      ...
    }
    else
    {
      ...
    }
  }
  else
  {
    ...
  }
}
```

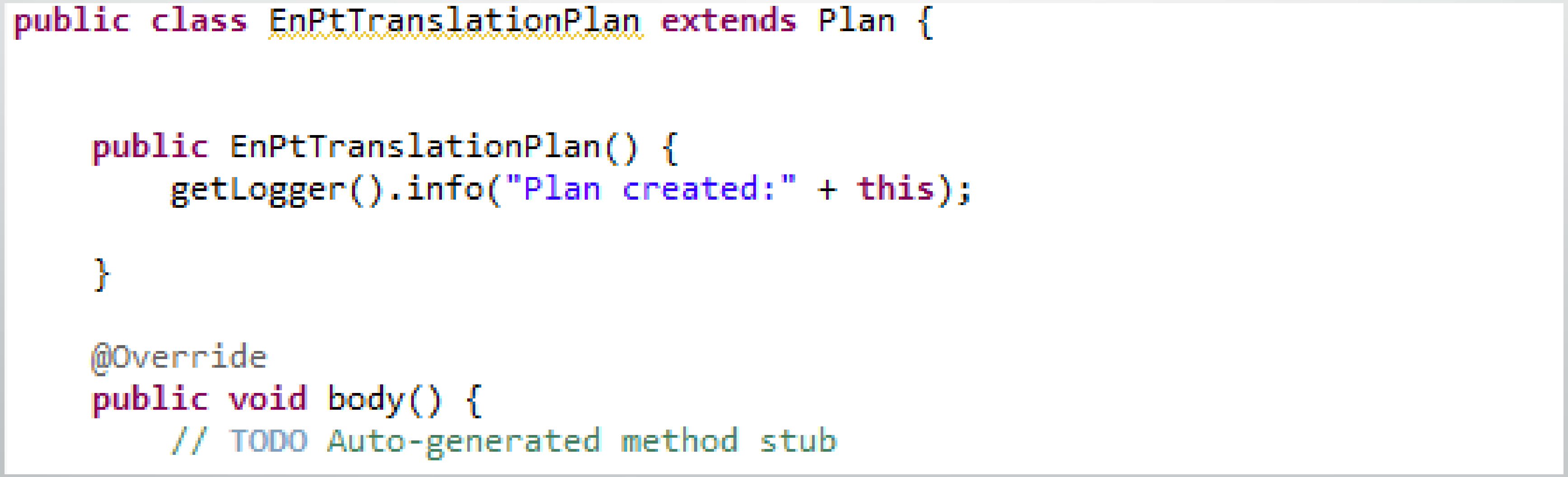
Plan Example



```xml
<plan name="eptrans">
    <body class="EnPtTranslationPlan" />
    <waitqueue>
        <messageevent ref="request_translation" />
    </waitqueue>
</plan>
```



```xml
<messageevent name="request_translation" direction="receive"
    type="fipa">
    <parameter name="performative" class="String" direction="fixed">
        <value>jadex.bridge.fipa.SFipa.REQUEST</value>
    </parameter>
    <parameter name="content-start" class="String" direction="fixed">
        <value>"translate"</value>
    </parameter>
</messageevent>
```



```java
public class EnPtTranslationPlan extends Plan {


    public EnPtTranslationPlan() {
        getLogger().info("Plan created:" + this);

    }

    @Override
    public void body() {
        // TODO Auto-generated method stub
```

Handling Events

Events are usually handled by plans

Internal events: **IInternalEvent**

- occurrence inside the agent

```
...
<events>
   <internalevent name="gui_update">
      <parameter name="content" class="String"/>
   </internalevent>
</events>
...
```

```
...
public void body()
{
  String update_info;
  ...
  // "gui_update" internal event type must be defined in the ADF
  IInternalEvent event = createInternalEvent("gui_update");
  // Setting the content parameter to the update info
  event.getParameter("content").setValue(update_info);
  dispatchInternalEvent(event);
  ...
}
```

Handling Events

Message events: **IMessageEvent**

- All message types an agent wants to send/receive are specified in the ADF

```
<imports>
  <import>jadex.base.fipa.SFipa</import>
</imports>
. . .
<events>
  <!-- A query-ref message with content "ping" -->
  <messageevent name="query_ping" type="fipa" direction="receive">
    <parameter name="performative" class="String" direction="fixed">
      <value>SFipa.QUERY_REF</value>
    </parameter>
    <parameter name="content" class="String" direction="fixed">
      <value>"ping"</value>
    </parameter>
  </messageevent>

  <!-- An inform message where content contains the word "hello" -->
  <messageevent name="inform_hello" type="fipa" direction="receive">
    <parameter name="performative" class="String" direction="fixed">
      <value>SFipa.INFORM</value>
    </parameter>
    <match>((String)$content).indexOf("hello")!=-1</match>
  </messageevent>
</events>
```

- Only incoming messages are handled by the event dispatching mechanism

```
<imports>
  <import>jadex.base.fipa.SFipa</import>
</imports>
. . .
<events>
  <!-- A query-ref message with content "ping" -->
  <messageevent name="query_ping" type="fipa" direction="send">
    <parameter name="performative" class="String">
      <value>SFipa.QUERY_REF</value>
    </parameter>
    <parameter name="content" class="String">
      <value>"ping"</value>
    </parameter>
  </messageevent>
</events>
```

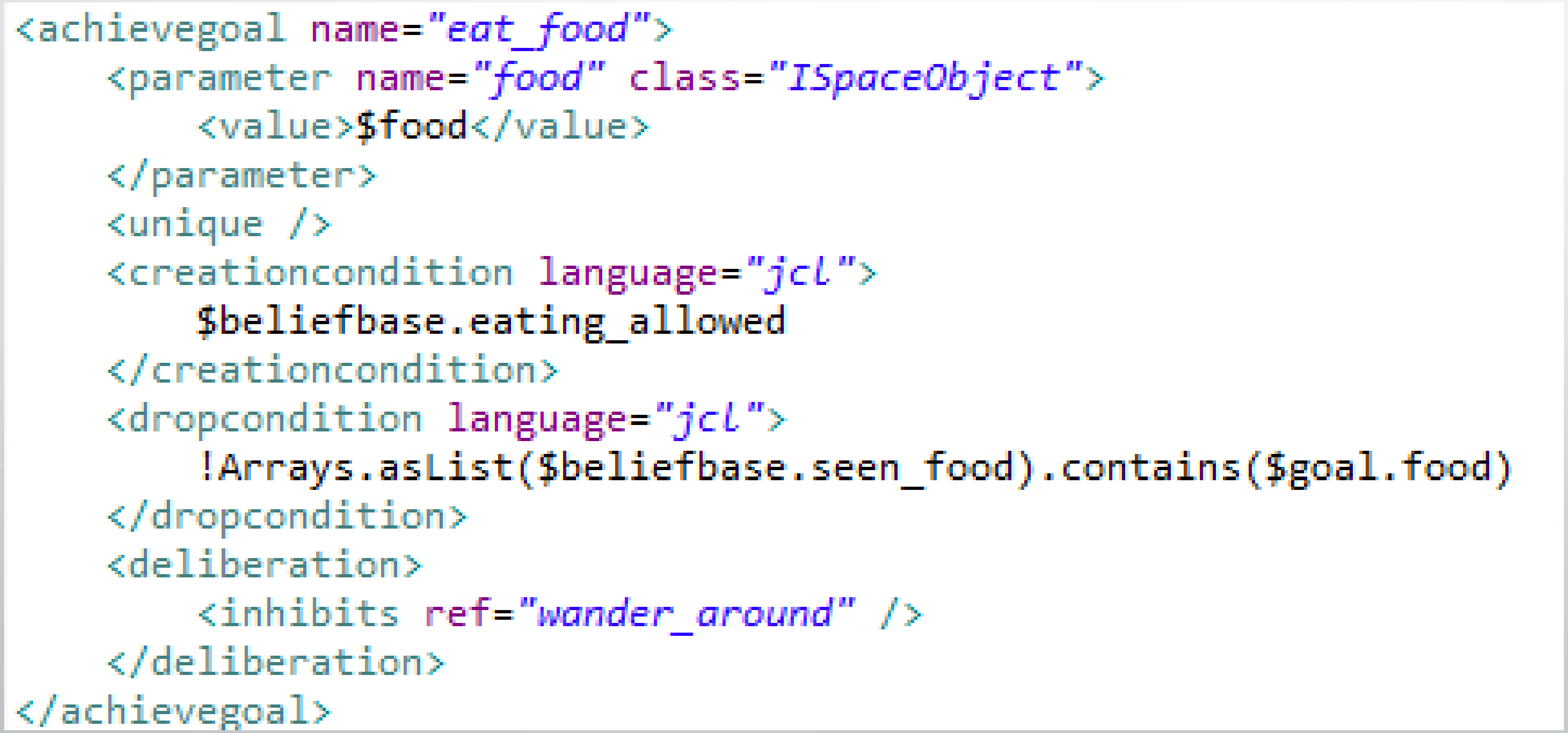
```
public void body()
{
  IMessageEvent me = createMessageEvent("query_ref");
  me.getParameterSet(SFipa.RECEIVERS).addValue(cid);
  //me.getParameter(SFipa.CONTENT).setValue("ping 2");
  sendMessage(me);
}
```

Creating Goals

Goals normally are associated with conditions

<unique/> will not pursue two goals to perform this goal;

<deliberation> goal is more important then the one defined in <inhibits>

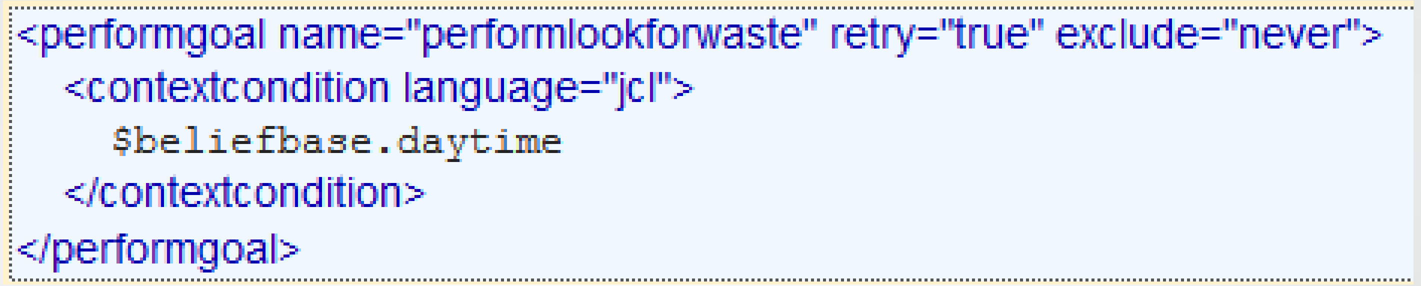


```
<achievegoal name="eat_food">
    <parameter name="food" class="ISpaceObject">
        <value>$food</value>
    </parameter>
    <unique />
    <creationcondition language="jcl">
        $beliefbase.eating_allowed
    </creationcondition>
    <dropcondition language="jcl">
        !Arrays.asList($beliefbase.seen_food).contains($goal.food)
    </dropcondition>
    <deliberation>
        <inhibits ref="wander_around" />
    </deliberation>
</achievegoal>
```

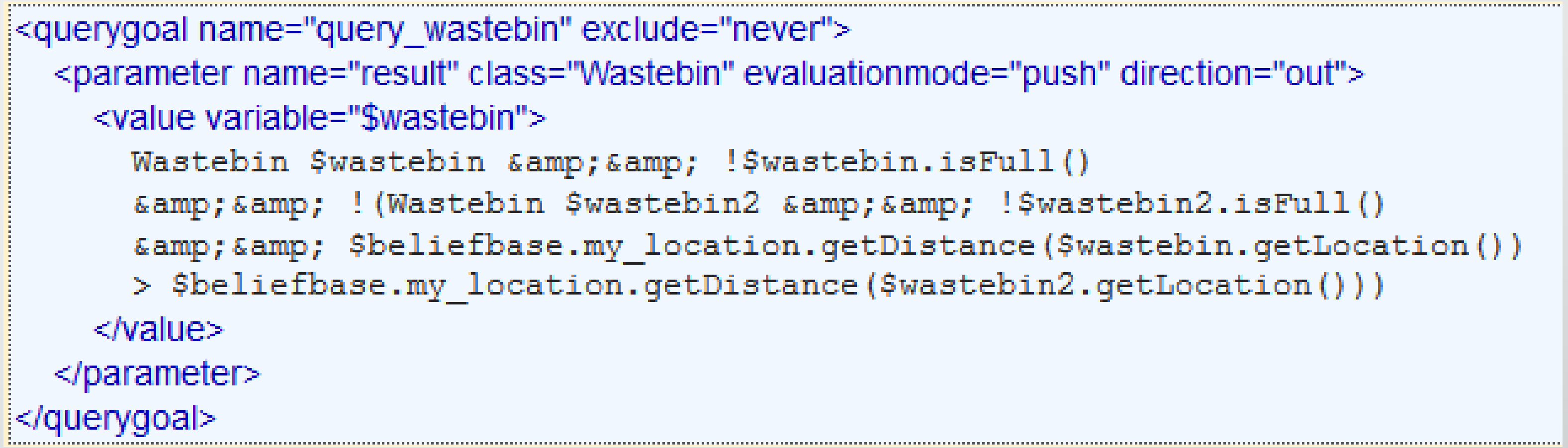
Creating Goals

Goals Applicability:

- **<unique/>**
- **<creationcondition>**
- **<contextcondition>**
- **<dropcondition>**
- **<deliberation>**



```
<performgoal name="performlookforwaste" retry="true" exclude="never">
  <contextcondition language="jcl">
    $beliefbase.daytime
  </contextcondition>
</performgoal>
```

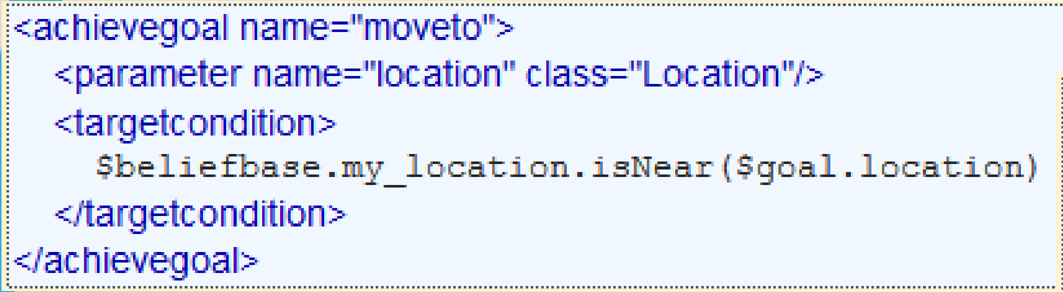


```
<querygoal name="query_wastebin" exclude="never">
  <parameter name="result" class="Wastebin" evaluationmode="push" direction="out">
    <value variable="$wastebin">
      Wastebin $wastebin &amp;&amp; !$wastebin.isFull()
      &amp;&amp; !(Wastebin $wastebin2 &amp;&amp; !$wastebin2.isFull()
      &amp;&amp; $beliefbase.my_location.getDistance($wastebin.getLocation())
      > $beliefbase.my_location.getDistance($wastebin2.getLocation()))
    </value>
  </parameter>
</querygoal>
```

Achieve and Maintain Goals

Achieve goal:

- **<targetcondition>**

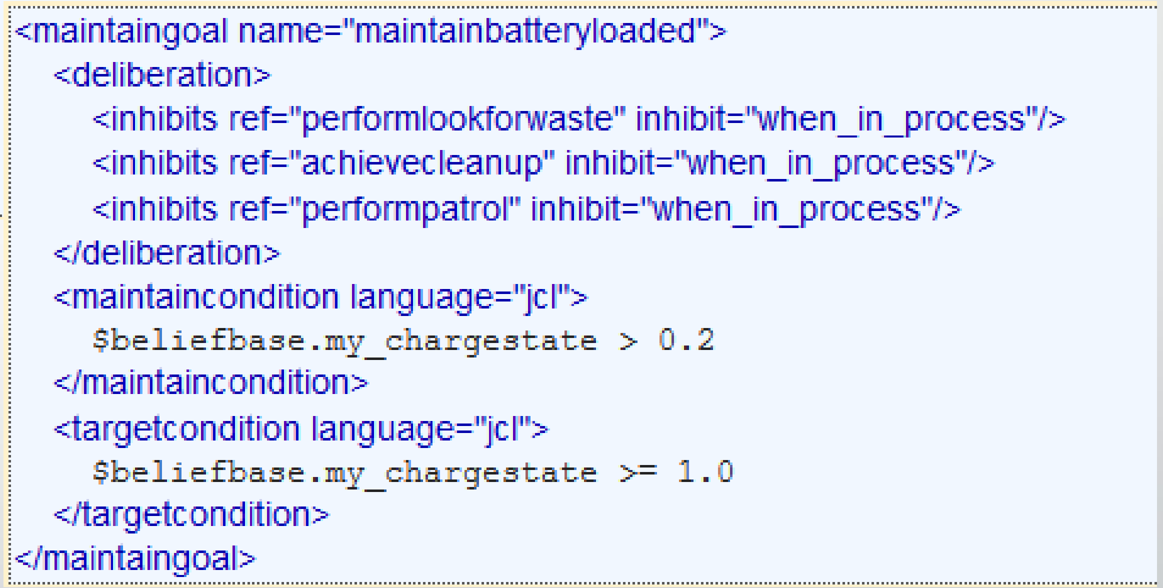


```
<achievegoal name="moveto">
  <parameter name="location" class="Location"/>
  <targetcondition>
    $beliefbase.my_location.isNear($goal.location)
  </targetcondition>
</achievegoal>
```

Maintain goal:

- **<maintaincondition>**

  <targetcondition>



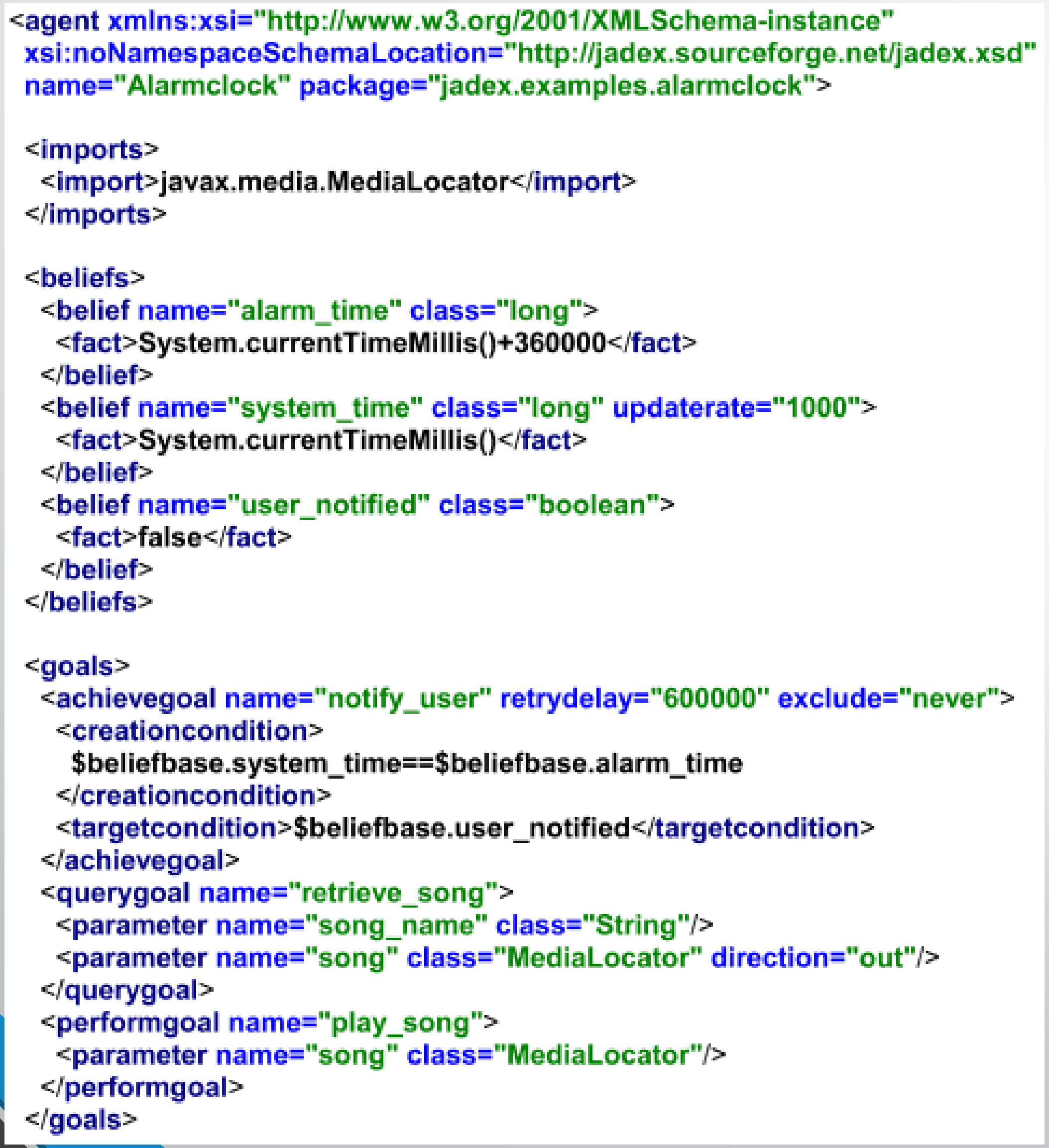
```
<maintaingoal name="maintainbatteryloaded">
  <deliberation>
    <inhibits ref="performlookforwaste" inhibit="when_in_process"/>
    <inhibits ref="achievecleanup" inhibit="when_in_process"/>
    <inhibits ref="performpatrol" inhibit="when_in_process"/>
  </deliberation>
  <maintaincondition language="jcl">
    $beliefbase.my_chargestate > 0.2
  </maintaincondition>
  <targetcondition language="jcl">
    $beliefbase.my_chargestate >= 1.0
  </targetcondition>
</maintaingoal>
```

Goals execution from Plans

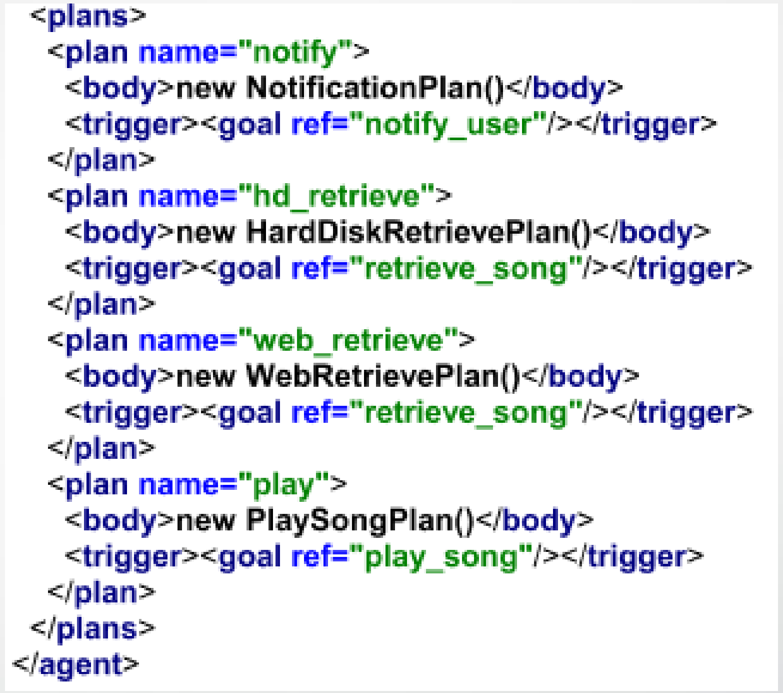
- createGoal()
- dispatchSubgoal()
- dispatchSubgoalAndWait()
- dispatchTopLevelGoal()
- drop()

```
public void body()
{
  // Create new top-level goal.
  IGoal goal1 = createGoal("mygoal");
  dispatchTopLevelGoal(goal1);
  ...
  // Create subgoal and wait for result.
  IGoal goal2 = createGoal("mygoal");
  dispatchSubgoalAndWait(goal2);
  Object val = goal2.getParameter("someoutparam").getValue();
  ...
  // Drop top-level goal.
  goal1.drop();
}
```

XML ADF Example

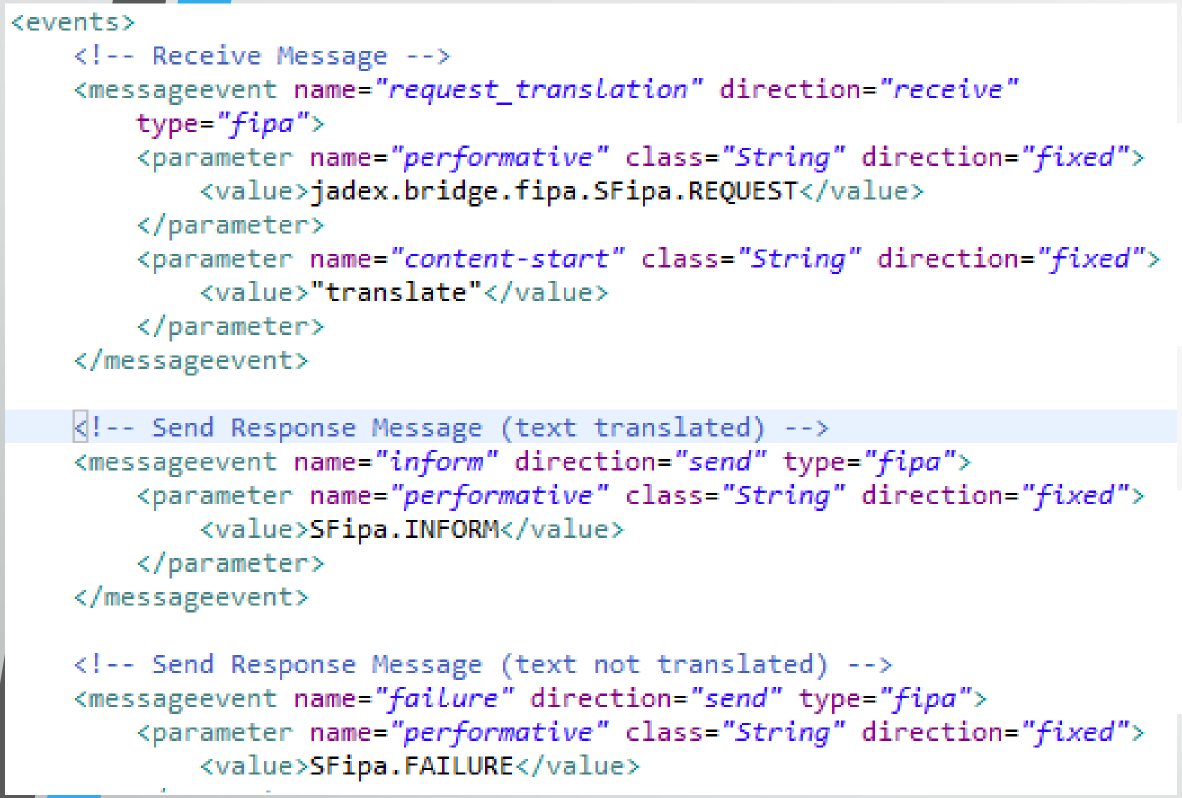


```xml
<agent xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="http://jadex.sourceforge.net/jadex.xsd"
 name="Alarmclock" package="jadex.examples.alarmclock">

 <imports>
  <import>javax.media.MediaLocator</import>
 </imports>

 <beliefs>
  <belief name="alarm_time" class="long">
   <fact>System.currentTimeMillis()+360000</fact>
  </belief>
  <belief name="system_time" class="long" updaterate="1000">
   <fact>System.currentTimeMillis()</fact>
  </belief>
  <belief name="user_notified" class="boolean">
   <fact>false</fact>
  </belief>
 </beliefs>

 <goals>
  <achievegoal name="notify_user" retrydelay="600000" exclude="never">
   <creationcondition>
    $beliefbase.system_time==$beliefbase.alarm_time
   </creationcondition>
   <targetcondition>$beliefbase.user_notified</targetcondition>
  </achievegoal>
  <querygoal name="retrieve_song">
   <parameter name="song_name" class="String"/>
   <parameter name="song" class="MediaLocator" direction="out"/>
  </querygoal>
  <performgoal name="play_song">
   <parameter name="song" class="MediaLocator"/>
  </performgoal>
 </goals>
```

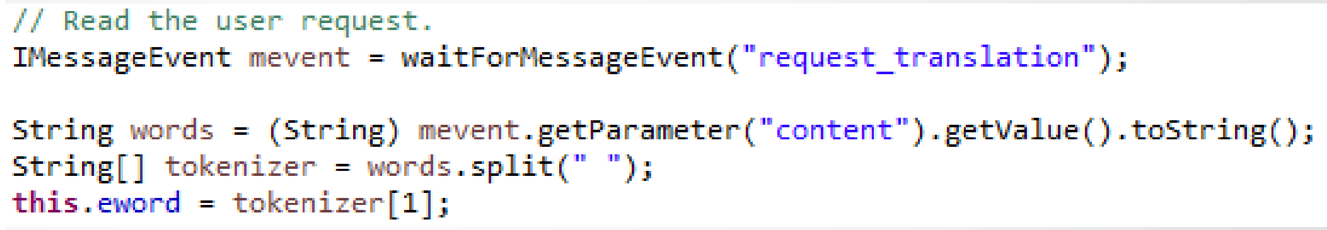


```xml
 <plans>
  <plan name="notify">
   <body>new NotificationPlan()</body>
   <trigger><goal ref="notify_user"/></trigger>
  </plan>
  <plan name="hd_retrieve">
   <body>new HardDiskRetrievePlan()</body>
   <trigger><goal ref="retrieve_song"/></trigger>
  </plan>
  <plan name="web_retrieve">
   <body>new WebRetrievePlan()</body>
   <trigger><goal ref="retrieve_song"/></trigger>
  </plan>
  <plan name="play">
   <body>new PlaySongPlan()</body>
   <trigger><goal ref="play_song"/></trigger>
  </plan>
 </plans>
</agent>
```

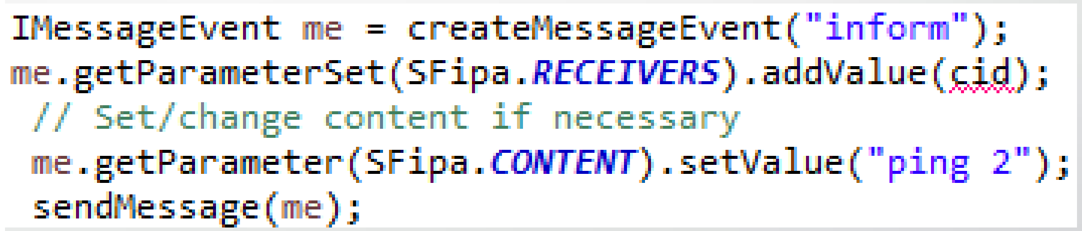
Receive/Send Message



```
<events>
    <!-- Receive Message -->
    <messageevent name="request_translation" direction="receive"
        type="fipa">
        <parameter name="performative" class="String" direction="fixed">
            <value>jadex.bridge.fipa.SFipa.REQUEST</value>
        </parameter>
        <parameter name="content-start" class="String" direction="fixed">
            <value>"translate"</value>
        </parameter>
    </messageevent>

    <!-- Send Response Message (text translated) -->
    <messageevent name="inform" direction="send" type="fipa">
        <parameter name="performative" class="String" direction="fixed">
            <value>SFipa.INFORM</value>
        </parameter>
    </messageevent>

    <!-- Send Response Message (text not translated) -->
    <messageevent name="failure" direction="send" type="fipa">
        <parameter name="performative" class="String" direction="fixed">
            <value>SFipa.FAILURE</value>
```



```
// Read the user request.
IMessageEvent mevent = waitForMessageEvent("request_translation");

String words = (String) mevent.getParameter("content").getValue().toString();
String[] tokenizer = words.split(" ");
this.eword = tokenizer[1];
```



```
IMessageEvent me = createMessageEvent("inform");
me.getParameterSet(SFipa.RECEIVERS).addValue(cid);
 // Set/change content if necessary
 me.getParameter(SFipa.CONTENT).setValue("ping 2");
 sendMessage(me);
```

JADEX Second Exercise

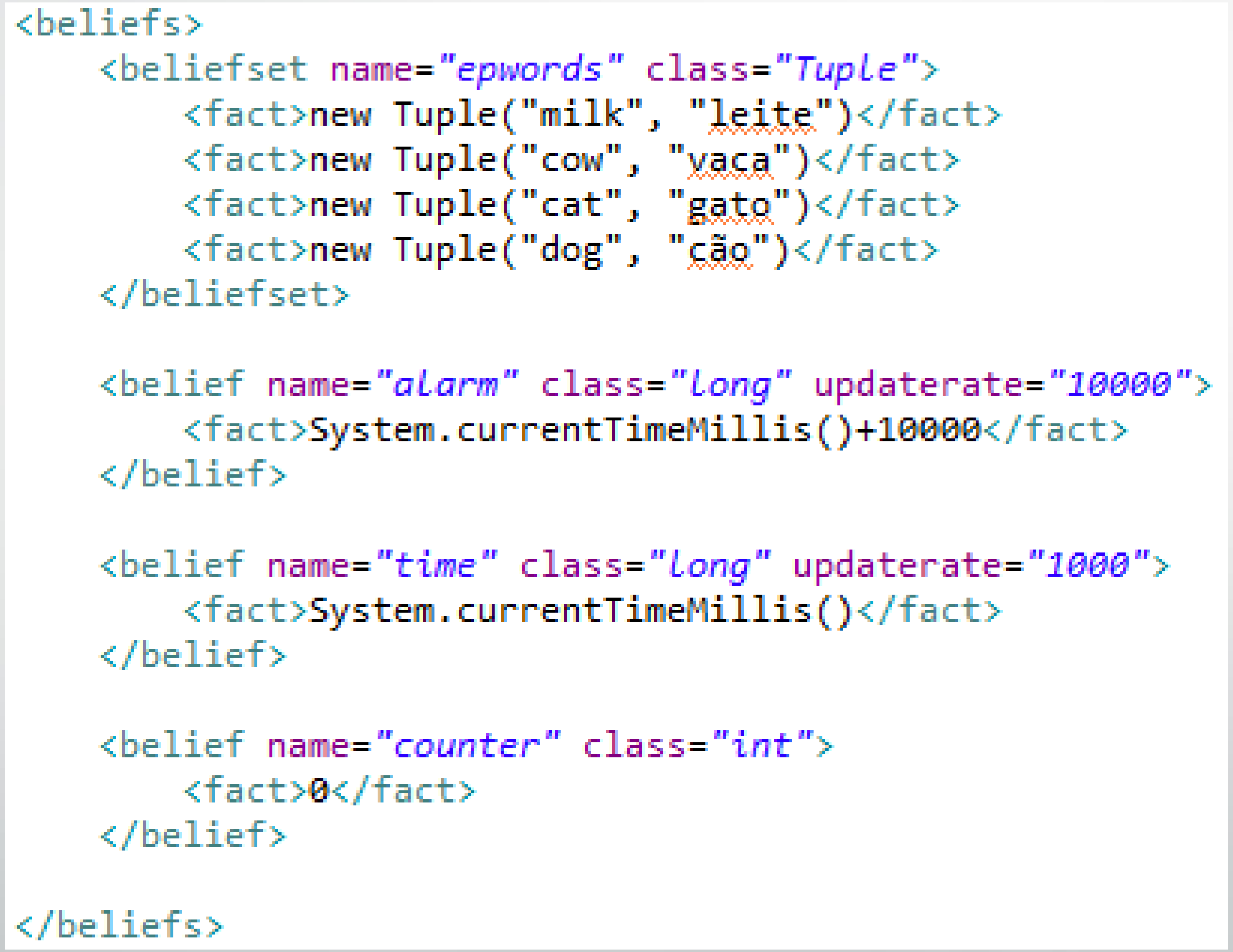
1. Reuse the Java Project available at elearning Platform;
2. Create Translater agent. When receiving one english word, this agent will respond with the word translated (e.g. english -> portuguese) [Based on the information saved in the beliefset];
3. Translator agent presents 3 plans:
   - **Plan translate** – based on a messageEvent, it will verify if the english word exists in the dictionary, translate it, and send the result to the user agent;
   - **Plan addword** – based on a messageEvent, it will add a new Tuple into the dictionary;
   - **Plan notify –** based on a timer, the agent must inform in the console the number of requests received [both translate and addword] every 10 seconds;

JADEX Second Exercise

XML filename: ***.agent.xml

```
<agent xmlns="http://jadex.sourceforge.net/jadex"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://jadex.sourceforge.net/jadex-bdi
                             http://jadex.sourceforge.net/jadex-bdi-2.4.xsd"
        name="***" package="***">
<imports>
<import>jadex.commons.*</import>
<import>jadex.bridge.fipa.*</import>
<import>jadex.bdi.runtime.*</import>
</imports>
```

JADEX Second Exercise



```
<beliefs>
    <beliefset name="epwords" class="Tuple">
        <fact>new Tuple("milk", "leite")</fact>
        <fact>new Tuple("cow", "vaca")</fact>
        <fact>new Tuple("cat", "gato")</fact>
        <fact>new Tuple("dog", "cão")</fact>
    </beliefset>

    <belief name="alarm" class="long" updaterate="10000">
        <fact>System.currentTimeMillis()+10000</fact>
    </belief>

    <belief name="time" class="long" updaterate="1000">
        <fact>System.currentTimeMillis()</fact>
    </belief>

    <belief name="counter" class="int">
        <fact>0</fact>
    </belief>

</beliefs>
```

JADEX Second Exercise

```
<expressions>
    <expression name="query_epword" class="String">
    select one $wordpair.get(1) from Tuple $wordpair in $beliefbase.epwords
    where $wordpair.get(0).equals($eword)
    </expression>
</expressions>
```

JADEX Framework Exercise II

Integrated Master's in Informatics Engineering

Intelligent Agents

2017/2018

<u>Synthetic Intelligence Lab</u>

Filipe Gonçalves

Paulo Novais