1 Official JADE reference

1.1 Useful methods

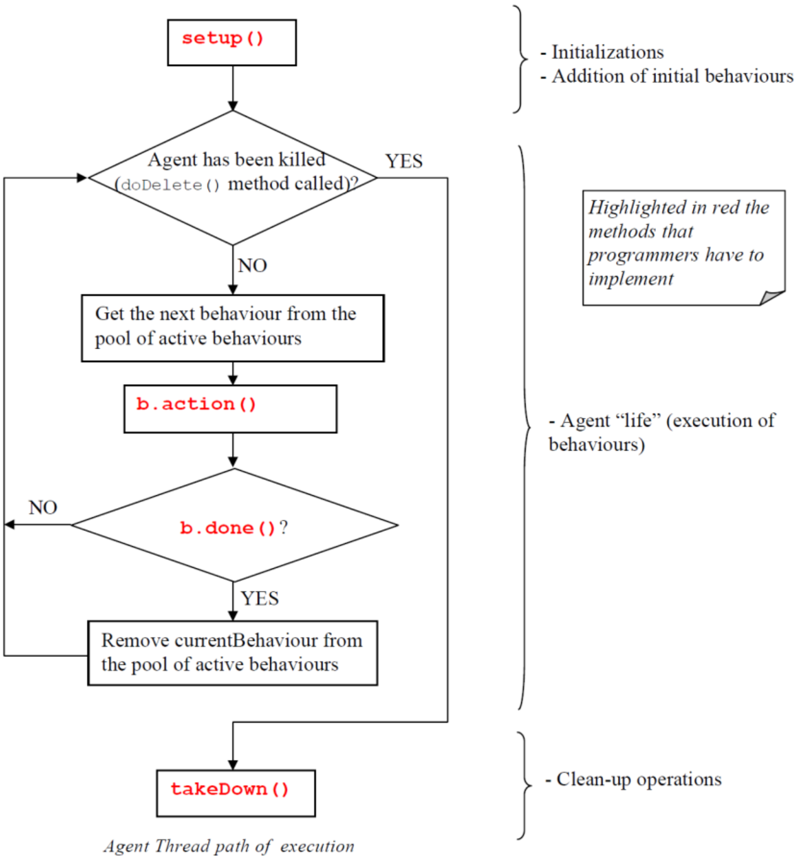
Class **Agent** and all derived classes inherit few methods, e.g.:

Method	Description	Return value
`setup()`	This protected method is an empty placeholder for application specific startup code.	`void`
`takeDown()`	This protected method is an empty placeholder for application specific cleanup code.	`void`
`doDelete()`	Make a state transition from *active*, *suspended* or *waiting* to *deleted* state within Agent Platform Life Cycle, thereby destroying the agent.	`void`
`addBehaviour(Behaviour b)`	This method adds a new behaviour to the agent.	`void`

While extending **`Behaviour`** class you have to implement:

Method	Description	Return value
`action()`	Runs the behaviour.	`abstract void`
`done()`	Check if this behaviour is done.	`abstract boolean`

Instead of class **`Behaviour`** you can extend: **`CyclicBehaviour`** or **`OneShotBehaviour`.**


setup()
Agent has been killed
(doDelete() method called)?
YES
NO
Get the next behaviour from the pool of active behaviours
b.action()
NO
b.done()?
YES
Remove currentBehaviour from the pool of active behaviours
takeDown()
- Initializations
- Addition of initial behaviours
Highlighted in red the methods that programmers have to implement
- Agent "life" (execution of behaviours)
- Clean-up operations

Agent Thread path of execution

More information in reference: http://jade.tilab.com/doc/api/index.html

1.2 Problems

In case of problem like:

```
WARNING: Error adding ICP jade.imtp.leap.JICP.JICPPeer@6576fe71[Error: Not
possible to launch JADE on a remote host (192.168.2.9). Check the -host and -
local-host options.].
```

Use additional args (details):

```
-gui -host 192.168.2.9 -port 12344
```

2 Task to do - Rabbits' farm

Students' task is to code simple rabbit farm. The farm consists four types of agents:

- Breeder - owner of the farm
- RabbitMale
- RabbitFemale
- Wolf - he eats rabbits, so nobody wanted him here

A simulation skeleton is ready -especially the Breeder class is almost fully coded. Your task is to implement those class of behaviors in rest of agents whose consist line TODO. Implementation is divided into three parts, which completion is base for your credit from laboratory. Before coding please look into the Breeder section below.

2.0 Your signature

Please hardcode your mail, name and student's ID in file Breeder.java.

2.1 Breeder

- The Breeder class is superior agent in our simulation.
- Only he can create new "agents".
- Take a look carefully to the method `ListenBehaviour`. Especially look to which types of messages Breeder reacts. How to run a simulation?
- The class is implemented, but probably some changes are necessary.
    - Tip: How agents can contact with breeder agent?

2.2 Natural end of life [1 point]

- Your first task is to implement time of life of rabbit and wolf.
- Agents birth with age = 0
- [Rabbit only]: After agent's age pass through lifetime the Breeder should be informed about that and after agent receive response from the superior is should remove itself. Wolfs lives forever.
- After agent's age pass through adultAge the proper message should appear on the console
- Single age unit is 1 second.

2.3 Wolf eating rabbits [1 points]

- Your second task is to implement full life cycle of wolf
- Every time unit wolf should eat rabbit
- Eating rabbits by the wolf should have consecutive flow:

1. Wolf is asking the Breeder about name of random rabbit
2. Wolf is informing the random rabbit about dinner plans (that Wolf wants to eat the rabbit)
3. If the rabbit lives and received the information about Wolf's plan, the rabbit should response to the Wolf positively and remove itself
4. If the Wolf does not receive response in noticeable time (please set proper time), he will return to first point of the flow.
5. If the Wolf receive positive response, the rabbit is eaten, and the Wolf should inform about this in the console

2.4 Breading of rabbits [1 point]

- Third task is to implement breeding of rabbits
- Every time unit rabbit male should try to make small rabbits, using algorithm:
    1. A RabbitMale asks the Breeder about random rabbit's name
    2. If the random rabbit is a RabbitFemale in proper age, she accepts the RabbitMale
    3. If male's response time was exceeded (how long the time should be? Please choose proper timeout) or he received negative answer the male is finishing courtship
    4. If the male received positive response the rabbit finishes courtship and starts making new rabbits (he is notifying the female about this). Additionally the male should not start new courtship during next two time units.
    5. After making of new rabbits the female should tell the Breeder that now is time to add new males to the pool of rabbits
    6. Please take a notice that female should not wait forever for response

2.5 Scoring system

In the end of lesson please send me your solution, the subject of message should start with:

```
[JADE1]Rabbits' farm 1
```

Points as above.

You have 3 weeks to send me fully implemented exercise with subject:

```
[JADE1]Rabbits' farm 2
```

Points as above.

Your final points after two laboratories are sum of:

- Points from first jade laboratory
- Points from implemented task during a second jade laboratory
- Points from implementing task at home

Please, send me only *.java files, not entire project.

If you have any suggestions how to improve these two exercises please include your suggestions in a mail.

Bibliography:

- JADE home page: http://jade.tilab.com

- JADE official tutorial from which many sentences are copied:
  http://jade.tilab.com/doc/tutorials/JADEProgramming-Tutorial-for-beginners.pdf
  http://jade.tilab.com/papers/JADETutorialIEEE/JADETutorial_Programming.pdf
- Another JADE tutorial, which added some information to the document:
  http://jade.tilab.com/doc/administratorsguide.pdf
- Robert's tutorial in polish