

Podstwy JADE

sprawozdanie z ćwiczeń laboratoryjnych 6:

Agnieszka Majkut

nr indeksu 286116

Wprowadzenie:

Celem laboratoriów było zapoznanie się z podstawami JADE.

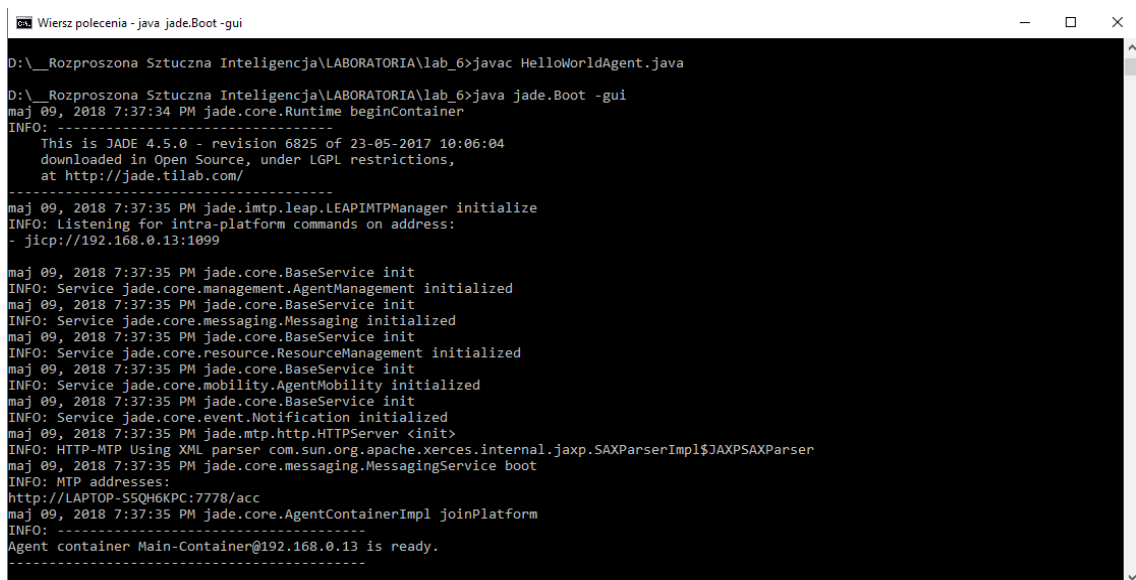
Należało wykonać:

- skompilować przykład **HelloWorldAgent.java**
- zmodyfikować powyższy przykład
- dodać kontener i uruchomić agenta klasy **HelloWorldAgent** w nowym kontenerze
- zmodyfikować przykład **TimeAgent.java** tak, aby agent usuwał się po 2 minutach
- dodać modyfikację pliku **HelloWorldAgent.java** tak, aby agent wypisywał dotychczasowy komunikat tyle razy, ile zostanie podane w parametrach agenta

Po ściągnięciu i rozpakowaniu JADE ustawiłam zmienną środowiskową CLASSPATH. Jej wartość to ścieżka do folderu jade.jar (tj. C:\Program Files\Java\JADE\JADE-bin_4.5.0\lib\jade.jar).

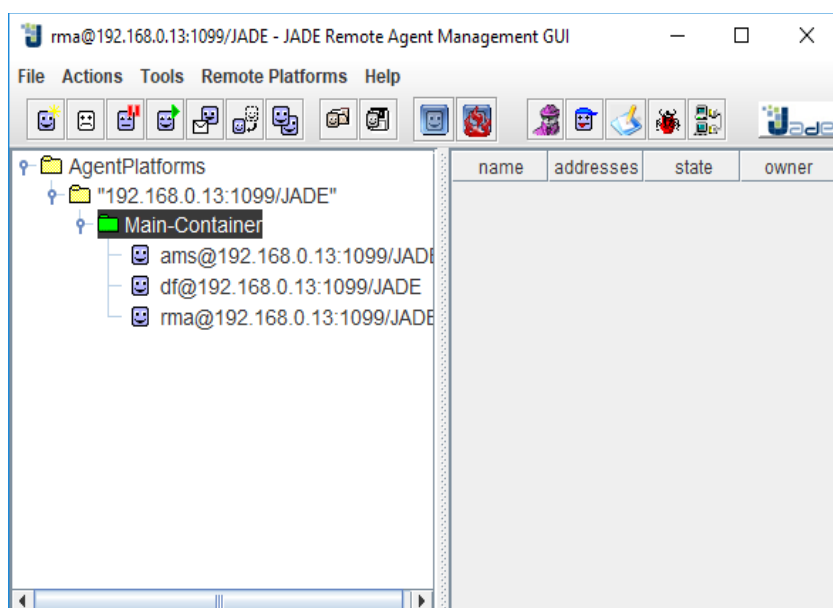
Uruchomiłam platformę JADE”

- w trybie tekstowym (**java jade.Boot -container**)
- w trybie graficznym (**java jade.Boot -gui**)



```
O:\_Rozproszona Sztuczna Inteligencja\LABORATORIA\lab_6>javac HelloWorldAgent.java
O:\_Rozproszona Sztuczna Inteligencja\LABORATORIA\lab_6>java jade.Boot -gui
maj 09, 2018 7:37:34 PM jade.core.Runtime beginContainer
INFO: -----
This is JADE 4.5.0 - revision 6825 of 23-05-2017 10:06:04
downloaded in Open Source, under LGPL restrictions,
at http://jade.tilab.com/
-----
maj 09, 2018 7:37:35 PM jade.imtp.leap.LEAPIMTPManager initialize
INFO: Listening for intra-platform commands on address:
- jicp://192.168.0.13:1099
maj 09, 2018 7:37:35 PM jade.core.BaseService init
INFO: Service jade.core.management.AgentManagement initialized
maj 09, 2018 7:37:35 PM jade.core.BaseService init
INFO: Service jade.core.messaging.Messaging initialized
maj 09, 2018 7:37:35 PM jade.core.BaseService init
INFO: Service jade.core.resource.ResourceManagement initialized
maj 09, 2018 7:37:35 PM jade.core.BaseService init
INFO: Service jade.core.mobility.AgentMobility initialized
maj 09, 2018 7:37:35 PM jade.core.BaseService init
INFO: Service jade.core.event.Notification initialized
maj 09, 2018 7:37:35 PM jade.mtp.http.HTTPServer <init>
INFO: HTTP-MTP Using XML parser com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl$JAXPSAXParser
maj 09, 2018 7:37:35 PM jade.core.messaging.MessagingService boot
INFO: MTP addresses:
http://LAPTOP-S5QH6KPC:7778/acc
maj 09, 2018 7:37:35 PM jade.core.AgentContainerImpl joinPlatform
INFO: -----
Agent container Main-Container@192.168.0.13 is ready.
```

rys1. Uruchomienie platformy JADE

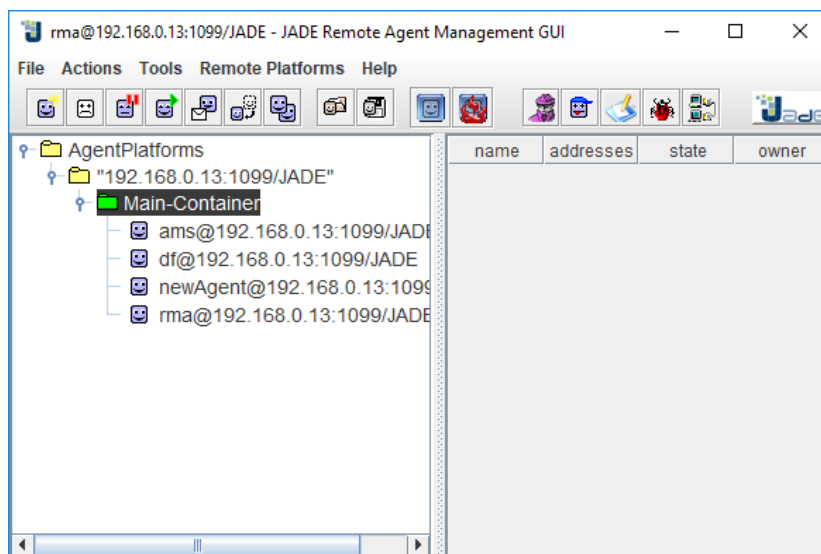
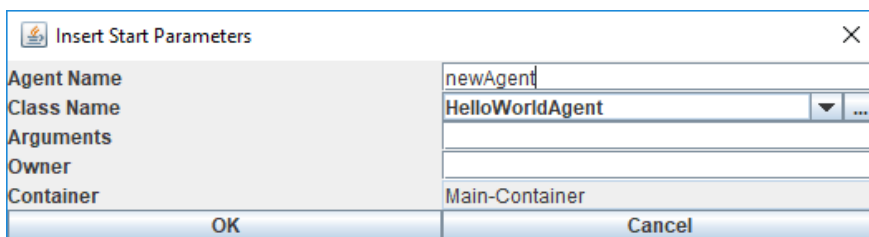


rys2. GUI platformy JADE

Skopiowałam przykład hello oraz skompilowałam klasę HelloWorldAgent. Uruchomiłam agenta z linii poleceń (**java jade.Boot -gui newAgent:HelloAgent**) oraz za pomocą GUI.

```
Hello World! My name is newAgent
```

rys3. Agent wyświetlił powyższy tekst w CMD



rys4, 5. Uruchomienie agenta przy pomocy GUI

Dokonałam modyfikacji kodu **HelloWorldAgent**, tak aby agent nie usuwał się po wypisaniu tekstu na ekran. Następnie skompilowałam kod.

```
3 import jade.core.Agent;
4
5 public class HelloWorldAgent extends Agent {
6
7     protected void setup() {
8         System.out.println("Hello World! My name is "+getLocalName());
9
10        //doDelete();
11    }
12 }
```

rys6. Należy wycommentować lub usunąć podświetloną linię 10.

Zmodyfikowałam **TimeAgent.java** tak, aby agent usuwał się dopiero po 2 minutach.

```
3 import jade.core.Agent;
4 import jade.core.behaviours.WakerBehaviour;
5 import jade.core.behaviours.TickerBehaviour;
6
7 public class TimeAgent extends Agent {
8
9     protected void setup() {
10         System.out.println("Agent "+getLocalName()+" started.");
11
12         // Add the TickerBehaviour (period 1 sec)
13         addBehaviour(new TickerBehaviour(this, 1000) {
14             protected void onTick() {
15                 System.out.println("Agent "+myAgent.getLocalName()+" tick="+getTickCount());
16             }
17         });
18
19         // Add the WakerBehaviour (wakeup-time 2 min)
20         addBehaviour(new WakerBehaviour(this, 120000) {
21             protected void handleElapsedTimeout() {
22                 System.out.println("Agent "+myAgent.getLocalName()+" It's wakeup-time. Bye...");
23                 myAgent.doDelete();
24             }
25         });
26     }
27 }
28
```

rys7. Klasa TimeAgent.java

```
Agent newAgent: tick=100
Agent newAgent: tick=101
Agent newAgent: tick=102
Agent newAgent: tick=103
Agent newAgent: tick=104
Agent newAgent: tick=105
Agent newAgent: tick=106
Agent newAgent: tick=107
Agent newAgent: tick=108
Agent newAgent: tick=109
Agent newAgent: tick=110
Agent newAgent: tick=111
Agent newAgent: tick=112
Agent newAgent: tick=113
Agent newAgent: tick=114
Agent newAgent: tick=115
Agent newAgent: tick=116
Agent newAgent: tick=117
Agent newAgent: tick=118
Agent newAgent: tick=119
Agent newAgent: It's wakeup-time. Bye...
```

rys8. Agent wyświetlał tiki przez 2 minuty, po których zakończył działanie

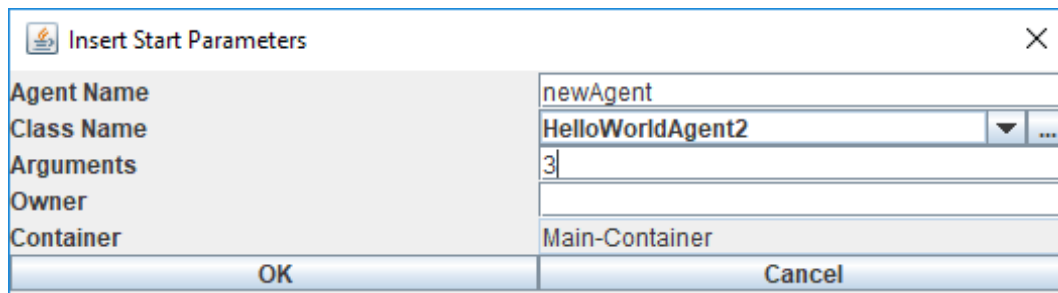
Przyłączyłam kontener na moim komputerze do zdalnego hosta za pomocą komendy

java jade.Boot -container -container-name C – host <adres IP hosta>

Poprosiłam o usunięcie mojego kontenera. Przyłączyłam jeszcze raz z dodaniem agenta klasy TimeAgent. Sklonowałam agenta (sklonowany agent będzie robił to samo co jego pierwowzór – oryginalny agent). Poprosiłam o mirację agenta (po migracji agenci będą wykonywać kod w osobnych terminalach).

Gdy brakuje tworzonego agenta lub kontenera, w którym moglibyśmy go umieścić program zwróci błąd, że dana klasa nie istnieje, agent nie zostanie utworzony. Zostanie wyświetlony błąd, że klasa kontanera nie istnieje jeżeli będziemy chcieli użyć kontenera, który nie istnieje.

Zmodyfikowałam plik **HelloWorldAgent**, tak aby agent wypisywał komunikat tyle razy ile zostanie podane w parametrach agenta.



| Insert Start Parameters | |
|-------------------------|------------------|
| Agent Name | newAgent |
| Class Name | HelloWorldAgent2 |
| Arguments | 3 |
| Owner | |
| Container | Main-Container |
| OK Cancel | |

```
Passed argument = 3
Test Agenta : 0
Test Agenta : 1
Test Agenta : 2
```

rys9,10. Agent wypisał 3 razy komunikat na ekran, tyle razy ile podałam w parametrach

Zmodyfikowałam program, tak aby wszystko co było w metodzie main() zostało wywołane w klasie agenta.

```
3 public class HelloAgent extends Agent
4 {
5     protected void setup()
6     {
7         System.out.println("Hello World. ");
8         System.out.println("My name is "+ getLocalName());
9         this.main(null);
10    }
11
12    public static void main(String[] args) {
13        System.out.println("Main test");
14    }
15
16 }
```

```
Hello World.
My name is newAgent
Main test
```