Fabiola Dąbroś

Inżynieria Obliczeniowa gr.1

Ćwiczenia 6 – Podstawy JADE

Wykonanie ćwiczenia rozpoczęłam od zapoznania się z plikami programmersguide.pdf oraz administratorguide.pdf. Utworzyłam pliki Compiljade.bat, który umożliwia kompilację pliku oraz runjade.bat, który umożliwia uruchomienie platformy jade z Gui. Innym sposobem na uruchomienie platformy Jade jest uruchomienie jej za pomocą konsoli.

Skopiowałam przykład hello znajdujący się w folderze Jade. Skompilowałam klasę HelloWorldAgent1 w wyniku czego otrzymałam następujący komunikat:

```
Agent container Main-Container@192.168.21.1 is ready.
---------------------------------------------
Hello World! My name is HelloWorldAgent
```

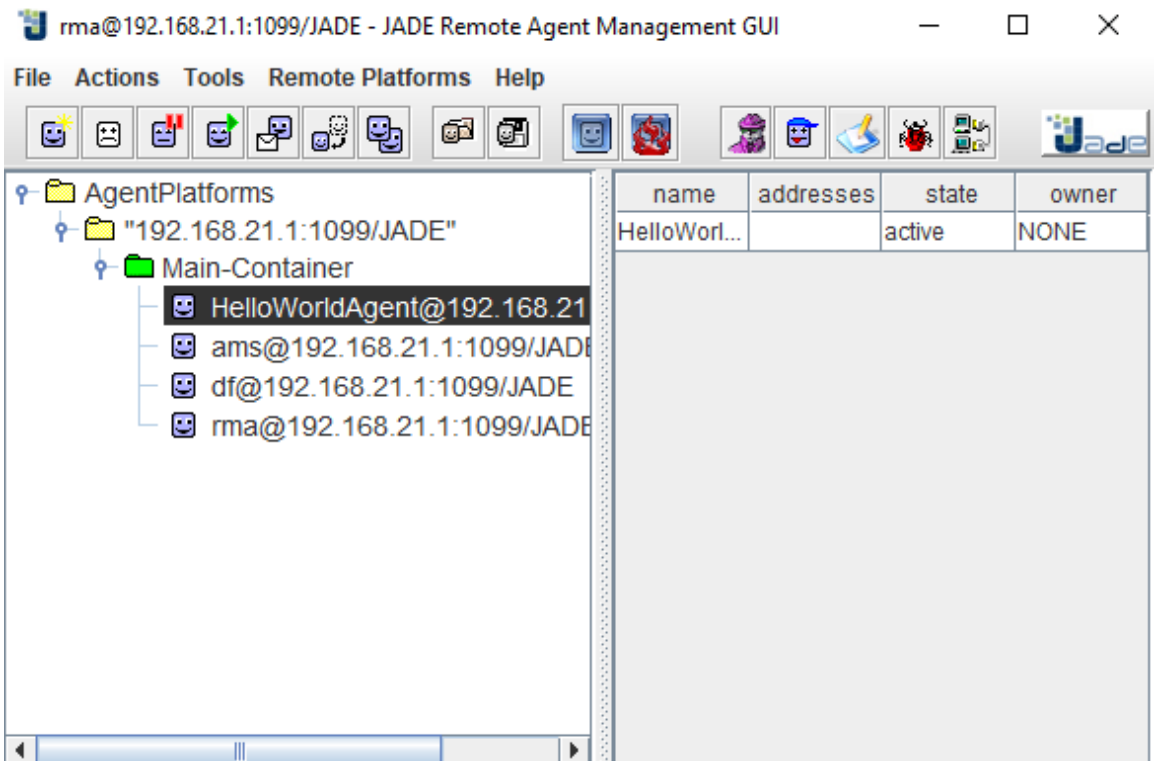
Następnie dokonałam modyfikacji kodu w taki sposób , aby agent nie usuwał się po wypisaniu tekstu na ekranie. Teraz kod wygląda następująco:

```
import jade.core.Agent;

public class HelloWorldAgent extends Agent{

    @Override
    protected void setup() {
        System.out.println("Hello World! My name is "+getLocalName());
        // Make this agent terminate
        //doDelete();
    }
}
```

(HelloWorldAgent2)

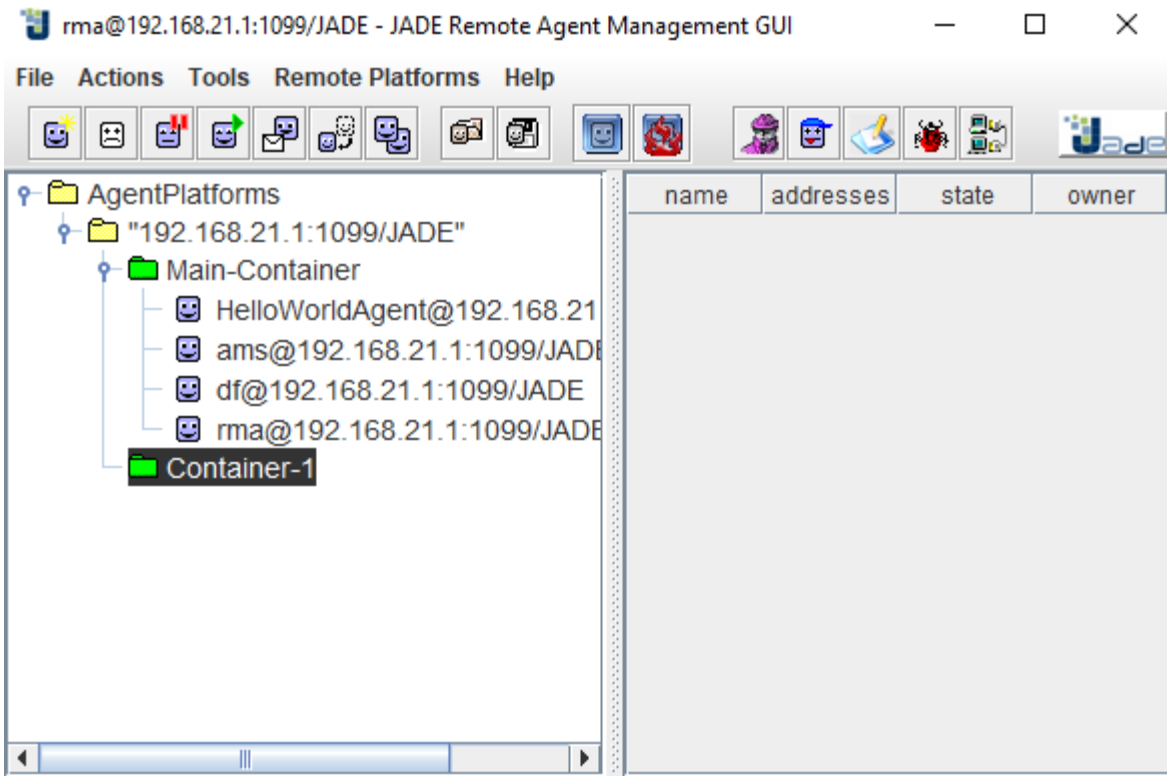
Należało usunąć metodę doDelete(), która jest odpowiedzialna za usunięcie agenta.

Potwierdzeniem działania jest obecność agenta w kontenerze:

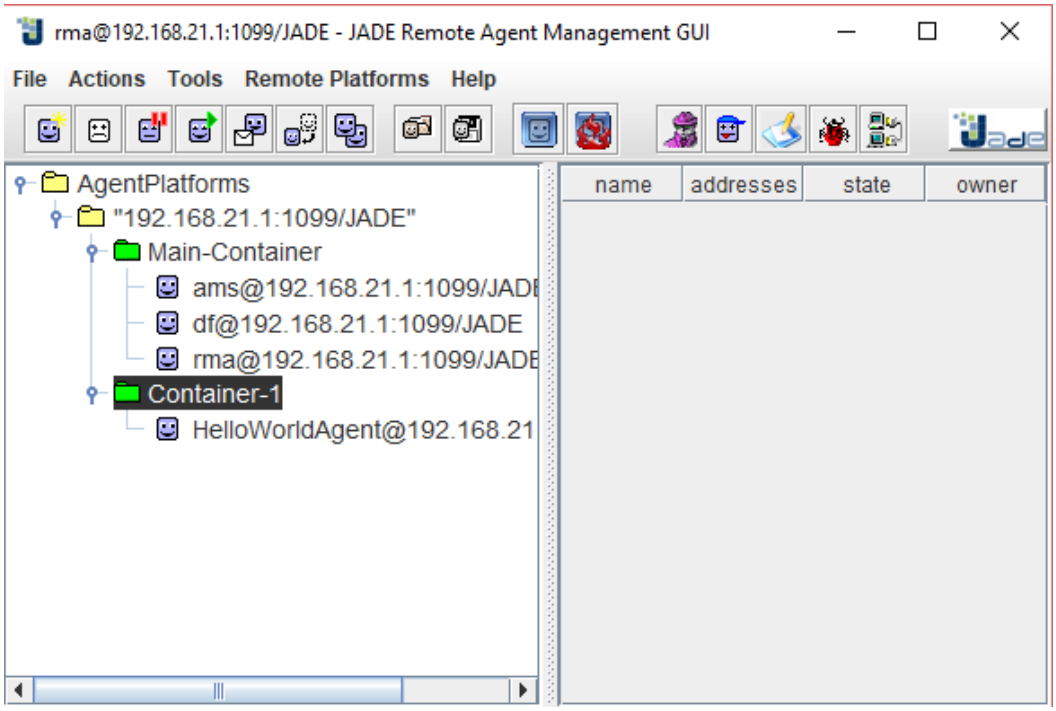


Następnie do istniejącej platformy dodałam kolejny kontener. W tym celu uruchomiłam konsolę jeszcze raz z poleceniem:

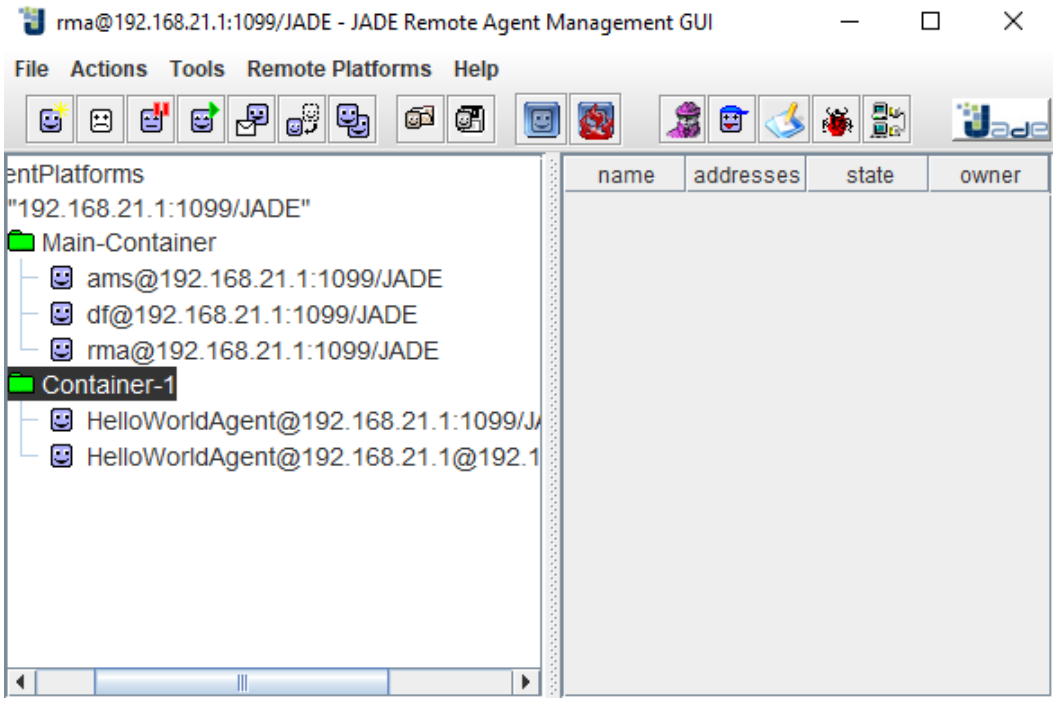
–container



Kolejno przeniosłam agenta do innego kontenera przy pomocy opcji migrate agent.



Sklonowałam agenta dzięki clone agent.



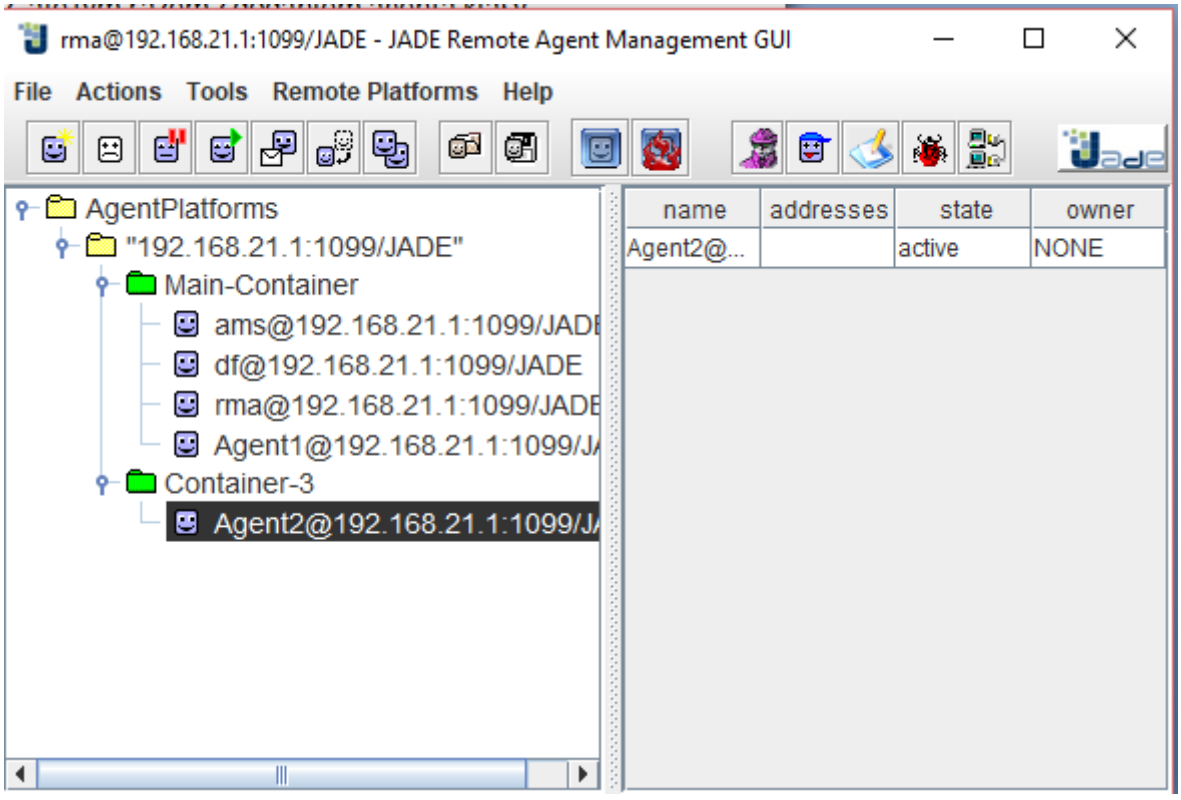
Oraz na sam koniec usunęłam agenta, korzystając z opcji Kill.

Kolejno zmodyfikowałam program TimeAgent w taki sposób aby agent usuwał się dopiero po 2 minutach. W tym celu zmodyfikowałam warunek metody dodającej zachowanie agenta, w tym wypadku WakerBehaviour, aby wykonała się po 120s.

addBehaviour(new WakerBehaviour(this, 120000)

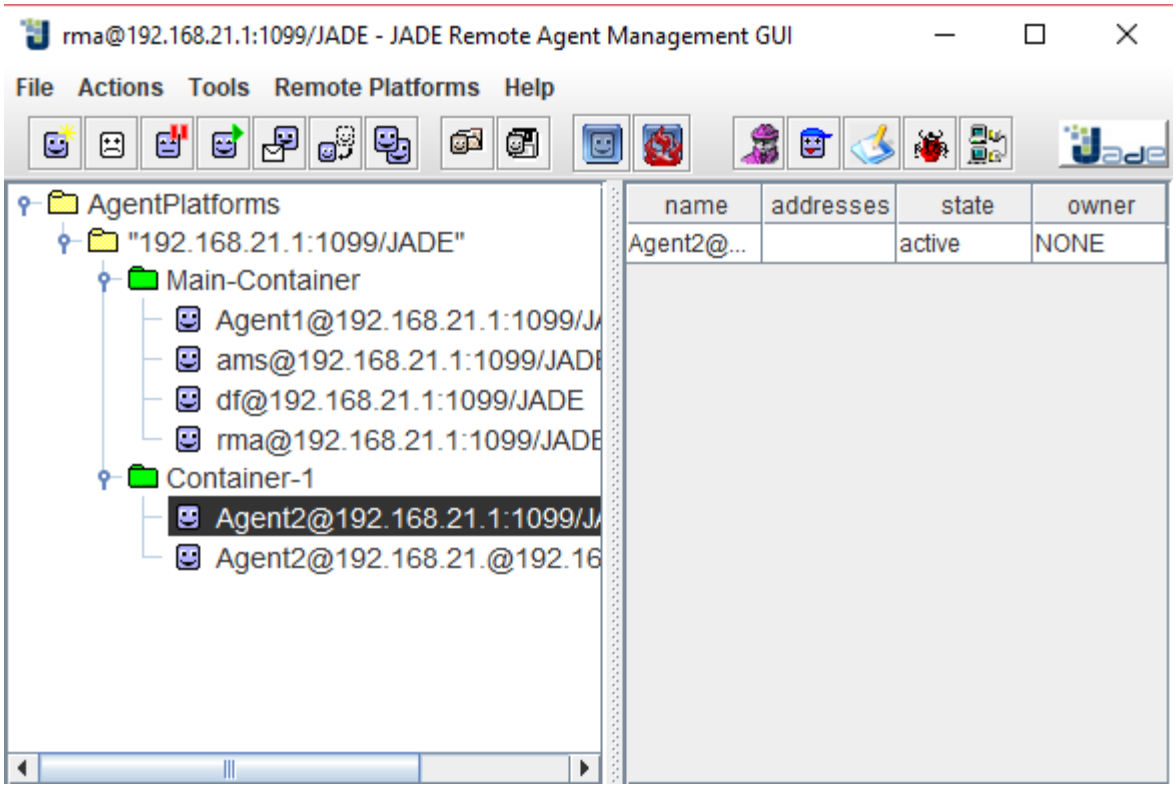
Przyłączyłam kontener do zdalnego hosta poprzez opcję
–container –name C –host<adres IP hosta>
a następnie usunęłam go używając kill.

Następnie przyłączyłam go jeszcze raz, ale tym razem z dodaniem agenta klasy TimeAgent dzięki poleceniu:
-container Agent2: TimeAgent



W tym momencie osobno wykonuje się tick Agenta1 oraz Agenta2, przy czym Agent1 kontynuuje swoje działanie a nie zaczyna od początku.

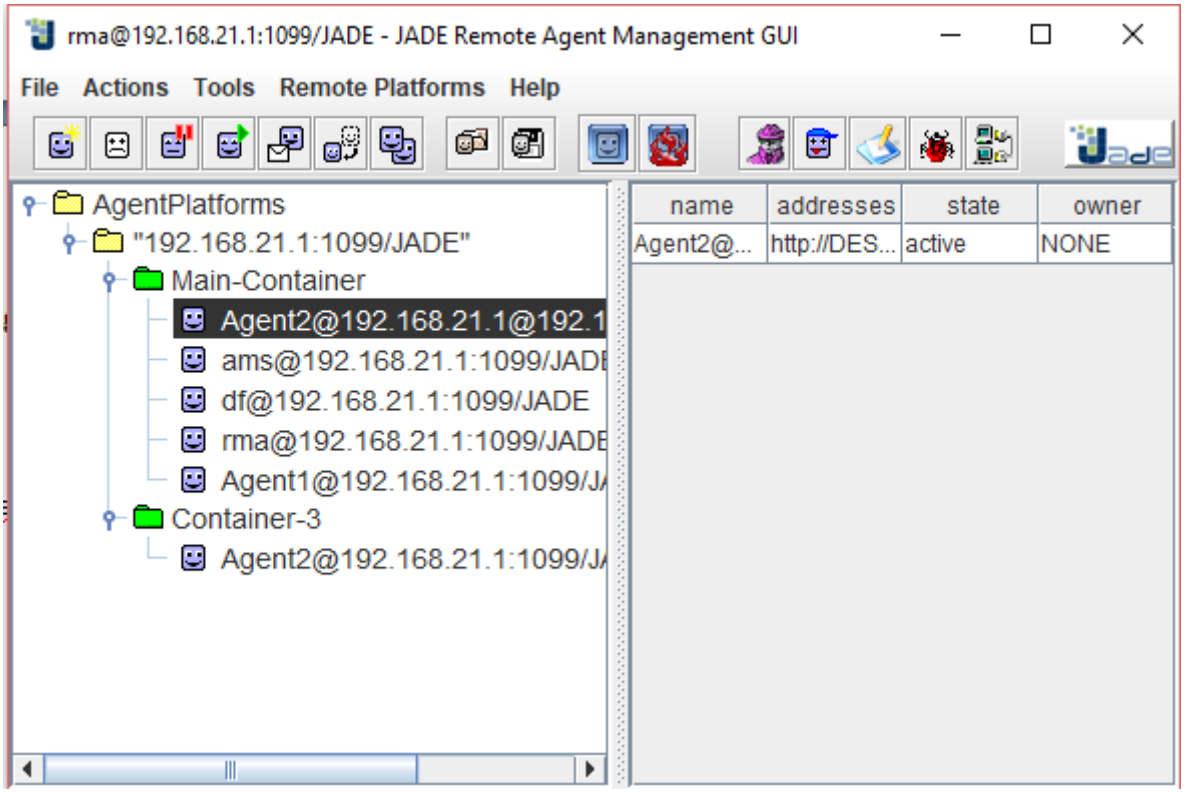
Kolejno wykonałam sklonowanie Agenta2.



Od tego momentu Agent2 wykonuje podwojnie swoje działanie.

```
Agent Agent2: tick=41
Agent Agent2@192.168.21.: tick=41
Agent Agent2: tick=42
Agent Agent2@192.168.21.: tick=42
Agent Agent2: tick=43
Agent Agent2@192.168.21.: tick=43
Agent Agent2: tick=44
Agent Agent2@192.168.21.: tick=44
Agent Agent2: tick=45
Agent Agent2@192.168.21.: tick=45
Agent Agent2: tick=46
Agent Agent2@192.168.21.: tick=46
Agent Agent2: tick=47
Agent Agent2@192.168.21.: tick=47
Agent Agent2: tick=48
Agent Agent2@192.168.21.: tick=48
Agent Agent2: tick=49
Agent Agent2@192.168.21.: tick=49
Agent Agent2: tick=50
Agent Agent2@192.168.21.: tick=50
Agent Agent2: tick=51
Agent Agent2@192.168.21.: tick=51
```

Następnie wykonowałam migracje Agenta2 do Main-Container.



Od tego momentu widzimy przeplatane działanie Agenta1 z Agentem2. Sklonowany Agent2 wciąż był aktywny ale nie pokazywał swojego działania.

```
Agent Agent1: tick=76
Agent Agent1: tick=77
Agent Agent1: tick=78
Agent Agent1: tick=79
Agent Agent2@192.168.21.1: tick=56
Agent Agent1: tick=80
Agent Agent2@192.168.21.1: tick=57
Agent Agent1: tick=81
Agent Agent2@192.168.21.1: tick=58
Agent Agent1: tick=82
Agent Agent2@192.168.21.1: tick=59
Agent Agent1: tick=83
Agent Agent2@192.168.21.1: tick=60
Agent Agent1: tick=84
Agent Agent2@192.168.21.1: tick=61
Agent Agent1: tick=85
Agent Agent2@192.168.21.1: tick=62
Agent Agent1: tick=86
Agent Agent2@192.168.21.1: tick=63
Agent Agent1: tick=87
Agent Agent2@192.168.21.1: tick=64
Agent Agent1: tick=88
Agent Agent2@192.168.21.1: tick=65
```

Jeżeli jest brak klasy tworzonego agenta to kompilator zwróci błąd. Tak samo stanie się w sytuacji gdy będziemy chcieli stworzyć agenta w kontenerze, który nie istnieje. Oprócz tego nie można też sklonować agenta do kontenera, który nie istnieje.

Następnie dokonałam modyfikacja HelloWorldAgent w taki sposób, aby agent wypisywał dotychczasowy komunikat tyle razy, ile zostanie podane w parametrach agenta. Kod prezentuje się następująco:

```
import jade.core.Agent;

public class HelloWorldAgent extends Agent{

    int added;

    protected void setup() {

        System.out.println("Agent "+getLocalName()+" started.");
        Object[] args = getArguments();
        added = Integer.parseInt(args[0].toString());
        System.out.println("Argument = "+added);
        for(int i=0;i<added;i++)
            System.out.println("Hello World! My name is " +getLocalName() +"argument: "+i);
      }
}
```

(HelloWorldAgent3)

Po uruchomieniu poleceniem:

-gui HelloWorldAgent:HelloWorldAgent(5)

otrzymujemy:

```
------------------------------------------
Agent HelloWorldAgent started.
Argument = 5
Hello World! My name is HelloWorldAgent.  argument: 0
Hello World! My name is HelloWorldAgent.  argument: 1
Hello World! My name is HelloWorldAgent.  argument: 2
Hello World! My name is HelloWorldAgent.  argument: 3
Hello World! My name is HelloWorldAgent.  argument: 4
```

Następnie zmodyfikowałam mój dotychczasowy program tak, aby wszytsko co było w metodzie main() zostało wywołane w klasie agenta. Prezentuje on się następująco:

```
import jade.core.Agent;

public class HelloWorldAgent extends Agent{

     protected void setup()
        {
            System.out.println("Hello World.My name is "+ getLocalName());
            this.main(null);
        }

        public static void main(String[] args) {
            System.out.println("Print from Main");
        }
}
```

(HelloWorldAgent4)

Po uruchomieniu otrzymujemy:

```
-------------------------------------------
Hello World.My name is HelloWorldAgent
Print from Main
```