

# Antras laboratorinis darbas Nr. 2

---

## Darbo tikslas:

- Sudarykite agentų platformą, kurioje pagrindinis agentas sukurtų kitus sistemos agentus pagrindiniame konteineryje (Main-Container). Agentai tarpusavyje bendrauja ACL žinutėmis.

## 1. Remiantis aprašu pirmajame laboratoriniame darbe susikurkite Pagrindinį agentą.

1.1. Agentų pakete sukurkite naują klasę: **Class Name: Pagrindinis**.

Class Name:

1.2. Importuokite **jade.core.Agent** klasę.

```
import jade.core.Agent;
```

1.3. Sukurkite **Pagrindinį** agentą.

```
public class Pagrindinis extends Agent
{
    protected void setup()
    {
    }
}
```

## 2. Susikurkite papildomą agentą.

2.1. Agentų pakete sukurkite naują klasę: **Class Name: Papildomas**.

Class Name:

2.2. Importuokite **jade.core.Agent** klasę: `import jade.core.Agent;`

2.3. Sukurkite agentą “**Papildomas**”.

```
public class Papildomas extends Agent
{
    protected void setup()
    {
        addBehaviour(new OneShotBehaviour(this)
        {
            @Override
            public void action()
            {
                System.out.println("-----");
                System.out.println("Papildomas Agentas");
                System.out.println("-----");
            }
        });
    }
}
```

### 3. Papildykite Pagrindinį agentą naujais funkcionalumais

3.1. Papildome programos kodą žemiau nurodytomis eilutėmis:

*AC* – agentų valdiklis, reikalingas naujų agentų kūrimui.

*Pack* – paketo, kuriame saugomi agentai, pavadinimas.

```
public class Pagrindinis extends Agent
{
    //Globalūs kintamieji
    private AgentController AC = null;
    private String pack = "Agentai";

    protected void setup()
    {
```

3.2. Papildykite programos kodą metodu, kuris sukuria agentą nurodytu pavadinimu **Pav**:

```
private void SukurtiAgentą(String Pav)
{
    String Pavadinimas = Pav;
}
```

3.3. Importuokite tokias bibliotekas:


```
import jade.wrapper.AgentContainer;
import jade.wrapper.AgentController;
```

3.4. Papildykite programos kodą tokiomis eilutėmis:

- (*AgentContainer*).getContainerController – grąžinama konteinerio valdiklio reikšmė, kurioje yra **Pagrindinis** agentas.
- *AgentContainer.createNewAgent*(*String name*, *String path*, *args*), kur  
**name** – sukuriama agento vardas  
**path** – kelias iki agento klasės  
**args** – papildomi argumentai

```
private void SukurtiAgentą(String Pav)
{
    String Pavadinimas = Pav;
    try
    {
        AgentContainer Konteineris = (AgentContainer) getContainerController();
        AC = Konteineris.createNewAgent( Pavadinimas, pack + "." + Pavadinimas ,null );
        AC.start();
    }
    catch(Exception any)
    {
        any.printStackTrace();
    }
}
```

Agentas sukuriamas pagrindiniame konteineryje (Main-Container).

 Main-Container

3.5. Papildykite programos kodą, kad ekrane būtų pranešama apie sukurtą naują agentą, t.y., atspausdintų sukurto agento vardą ir konteinerį, kuriame jis sukurtas.

```
private void SukurtiAgentą(String Pav)
{
    String Pavadinimas = Pav;
    try
    {
        AgentContainer Konteineris = (AgentContainer) getContainerController();
        AC = Konteineris.createNewAgent( Pavadinimas, pack + "." + Pavadinimas , null );
        AC.start();
        //Programos kodas
    }
    catch (Exception any)
    {
        any.printStackTrace();
    }
}
```




3.6. Importuokite: `import jade.core.behaviours.OneShotBehaviour;`

3.7. **setup()** viduje pridėkite: **addBehaviour(new OneShotBehaviour(this) {**  
Iškvieskite sukurtą metodą **SukurtiAgentą()**.

```
addBehaviour(new OneShotBehaviour(this)
{
    @Override
    public void action()
    {
        System.out.println("-----");
        SukurtiAgentą("Papildomas");
        System.out.println("-----");
    }
});
```

3.8. Paleiskite agentą. **Jade Remote Agent Management GUI**

Atsivėrusiame lange turėtume matyti pagrindiniame konteineryje sukurtus agentus.

 Main-Container  
     PAG\_Agentas@Platforma  
     Papildomas@Platforma

## 4. Agentų bendravimas ACL žinutėmis.

4.1. Kad agentai tarpusavyje bendrautų ACL žinutėmis, **pagrindiniam** agentui pridėkite ciklišką elgseną *CyclicBehaviour* , kuri apdorotų gaunamas žinutes.

**myAgent.receive()** – nuskaityta žinutė iš agento žinučių eilės.

**block()** – blokuojamos agento elgsenos. Ši funkcija žinučių apdorojimui reikalinga tam, kad agentas nenaudotų procesoriaus resursų laukdamas žinutės (nes žinutės gavimo laikas nėra žinomas).

```
addBehaviour( new CyclicBehaviour(this)
{
    public void action()
    {
        String vardas;
        String turinys;
        jade.lang.acl.ACLMessage zinute = myAgent.receive();
        if(zinute != null)
        {
        }
        else
        {
            block();
        }
    }
});
```

#### 4.2. Papildykite programos kodą žemiau pateiktomis eilutėmis :

ACLMessage.getSender – grąžinamas siuntėjas

ACLMessage getContent – grąžinamas žinutės turinys

Agentas.substring – nukreipiamas agento konteinerio vardas

```
public void action()
{
    String vardas;
    String turinys;
    jade.lang.acl.ACLMessage zinute = myAgent.receive();
    if(zinute != null)
    {
        vardas = zinute.getSender().getName();
        vardas = vardas.substring(0, vardas.indexOf("@"));
        if (vardas.equals("Papildomas"))
        {
        }
    }
}
```

#### 4.3. Papildykite programos kodą eilutėmis:

Gavus žinutę iš **“papildomas”** agento, **pagrindinis** agentas ekrane atspausdina: (1) iš ko gavo žinutę, (2) gautos žinutės turinį; (3) atsakymą siuntėjui

```

if (vardas.equals("Papildomas"))
{
    turinys = zinute.getContent();
    ACLMessage atsakymas;
    atsakymas = new ACLMessage(ACLMessage.INFORM);
    atsakymas.addReceiver(new AID(vardas, AID.ISLOCALNAME));
    //programos kodas
}

```

4.4. Papildomui agentui pridėkite elgseną (*TickerBehaviour*)

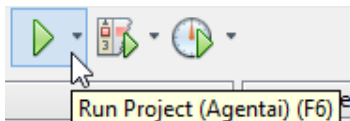
4.5. Papildykite programos kodą taip, kad **papildomas** agentas kas 5 sekundes išsiųstų ACL žinutę **pagrindiniam** agentui.

```

addBehaviour(new TickerBehaviour(this, 1234)
{
    protected void onTick()
    {
        // Programos kodas
    }
});

```

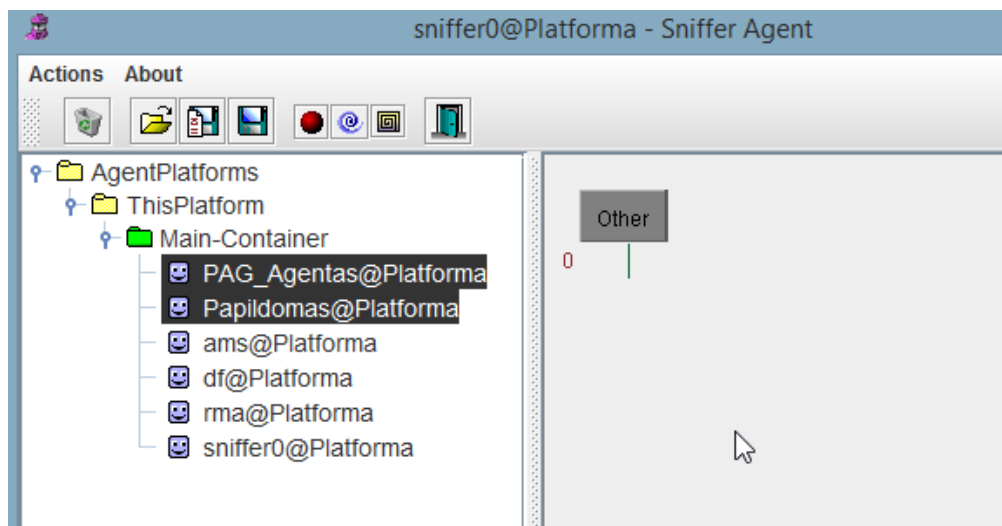
4.6. Paleiskite **pagrindinį** agentą. Spauskite **run main project** mygtuką.



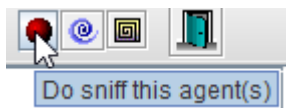
4.7. Automatiškai pasileidžia **Jade Remote Agent Management GUI**. Pasirenkame **AgentPlatforms** ir spaudžiame **Start Sniffer**.



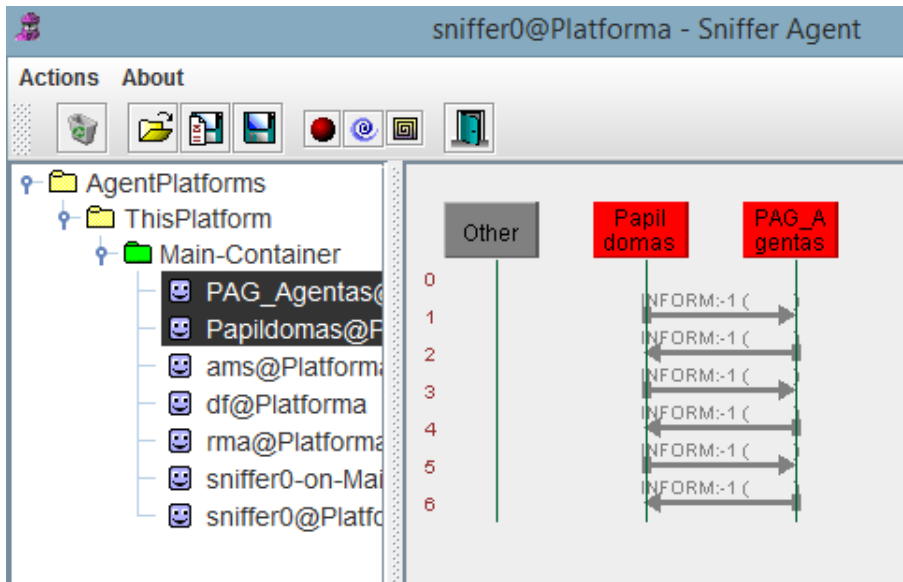
4.8. Atsidariusiame *Sniffer* lange pasirenkame abu agentus: **PAG\_Agentas** ir **Papildomas**.



4.9. Pasirinkus agentus spaudžiame **Do sniff this agent(s)** mygtuką.



4.10. Matome kaip **papildomas** agentas siunčia **INFORM** tipo žinute **PAG\_Agentui(Pagrindiniam)**, o jis siunčia atgal atsakymą.



## ACLMessage tipai

<a href="#">ACCEPT_PROPOSAL</a> constant identifying the FIPA performative
<a href="#">AGREE</a> constant identifying the FIPA performative
<a href="#">AMS_FAILURE_AGENT_NOT_FOUND</a> AMS failure reasons
<a href="#">AMS_FAILURE_AGENT_UNREACHABLE</a>
<a href="#">AMS_FAILURE_FOREIGN_AGENT_NO_ADDRESS</a>
<a href="#">AMS_FAILURE_FOREIGN_AGENT_UNREACHABLE</a>
<a href="#">AMS_FAILURE_SERVICE_ERROR</a>
<a href="#">AMS_FAILURE_UNAUTHORIZED</a>
<a href="#">AMS_FAILURE_UNEXPECTED_ERROR</a>
<a href="#">CANCEL</a> constant identifying the FIPA performative
<a href="#">CFP</a> constant identifying the FIPA performative
<a href="#">CONFIRM</a> constant identifying the FIPA performative
<a href="#">DISCONFIRM</a> constant identifying the FIPA performative

<a href="#"><u>DONT_NOTIFY_FAILURE</u></a>	User defined parameter key specifying, when set to "true", that if the delivery of a message fails, no FAILURE notification has to be sent back to the sender.
<a href="#"><u>FAILURE</u></a>	constant identifying the FIPA performative
<a href="#"><u>IGNORE_FAILURE</u></a>	User defined parameter key specifying, when set to "true", that if the delivery of a message fails, no failure handling action must be performed.
<a href="#"><u>INFORM</u></a>	constant identifying the FIPA performative
<a href="#"><u>INFORM_IF</u></a>	constant identifying the FIPA performative
<a href="#"><u>INFORM_REF</u></a>	constant identifying the FIPA performative
<a href="#"><u>NO_CLONE</u></a>	User defined parameter key specifying that this message does not need to be cloned by the message delivery service.
<a href="#"><u>NOT_UNDERSTOOD</u></a>	constant identifying the FIPA performative
<a href="#"><u>PROPAGATE</u></a>	constant identifying the FIPA performative
<a href="#"><u>PROPOSE</u></a>	constant identifying the FIPA performative
<a href="#"><u>PROXY</u></a>	constant identifying the FIPA performative
<a href="#"><u>QUERY_IF</u></a>	constant identifying the FIPA performative
<a href="#"><u>QUERY_REF</u></a>	constant identifying the FIPA performative
<a href="#"><u>REAL_SENDER</u></a>	User defined parameter key specifying the AID of the real sender of a message.
<a href="#"><u>REFUSE</u></a>	constant identifying the FIPA performative
<a href="#"><u>REJECT_PROPOSAL</u></a>	constant identifying the FIPA performative
<a href="#"><u>REQUEST</u></a>	constant identifying the FIPA performative
<a href="#"><u>REQUEST_WHEN</u></a>	constant identifying the FIPA performative
<a href="#"><u>REQUEST_WHENEVER</u></a>	constant identifying the FIPA performative
<a href="#"><u>SF_TIMEOUT</u></a>	User defined parameter key specifying that this message must be stored for a given timeout (in ms) in case it is sent to/from a temporarily disconnected split container.
<a href="#"><u>SUBSCRIBE</u></a>	constant identifying the FIPA performative
<a href="#"><u>SYNCH_DELIVERY</u></a>	User defined parameter key specifying that this message must be delivered synchronously.
<a href="#"><u>TRACE</u></a>	User defined parameter key specifying that the JADE tracing mechanism should be activated for this message.
<a href="#"><u>UNKNOWN</u></a>	constant identifying an unknown performative

## Punktų realizacijos

### 3.5 Punkto realizacija:

```
private void SukurtiAgentą(String Pav)
{
    String Pavadinimas = Pav;
    try
    {
        AgentContainer Konteineris = (AgentContainer)getContainerController();
        AC = Konteineris.createNewAgent( Pavadinimas, pack + "." + Pavadinimas ,null );
        AC.start();
        System.out.println("-----");
        System.out.println("Pagrindinis agentas : " + getLocalName());
        System.out.println("Pagrindinio agento sukurtas agentas : " + Pavadinimas);
        System.out.println("Konteineris : " + Konteineris.getContainerName());
        System.out.println("-----");
    }
    catch(Exception any)
    {
        any.printStackTrace();
    }
}
```

### 4.3 Punkto realizacija:

```
if (vardas.equals("Papildomas"))
{
    turinys = zinate.getContent();
    ACLMessage atsakymas;
    atsakymas = new ACLMessage(ACLMessage.INFORM);
    atsakymas.addReceiver(new AID(vardas, AID.ISLOCALNAME));

    System.out.println("Is agento " + vardas + " gauta žinutė: ");
    System.out.println(turinys);
    atsakymas.setContent("Pagrindinis agentas patvirtina gaves žinute");
    send(atsakymas);
}
```

### 4.5 Punkto realizacija:

```
addBehaviour(new TickerBehaviour(this, 5000)
{
    protected void onTick()
    {
        ACLMessage zinate;
        zinate = new ACLMessage(ACLMessage.INFORM);
        zinate.addReceiver(new AID("PAG_Agentas", AID.ISLOCALNAME));
        zinate.setContent("Papildomo agento žinutės turinys");
        send(zinate);
    }
});
```



## Savarankiško darbo užduotys

Nr.	Sudėtingumas	Užduotis
1		<ul style="list-style-type: none"> <li>Sukurti agentą <b>Pagrindinis</b> ir agentą <b>Papildomas</b>.</li> <li><b>Pagrindinis</b> agentas kas 2 sekundes parašo pranešimą, kurį reikia nurodyti Argumentų eilutėje.</li> <li><b>Papildomas</b> agentas gavęs žinutę patvirtina, kad gavo žinutę perspausdindamas žinutės turinį.</li> <li><b>Pagrindinio</b> agento žinutės tipas : <b>INFORM</b></li> <li><b>Papildomas</b> agento žinutės tipas: <b>CONFIRM</b></li> <li>Argumentų eilutė turi atrodyti taip: „&lt;<b>Pagrindinis</b> agentas&gt;(&lt;pranešimas&gt;)“</li> </ul>
2		<ul style="list-style-type: none"> <li>Sukurti agentą <b>Pagrindinis</b> ir agentą <b>Papildomas</b>.</li> <li><b>Pagrindinis</b> agentas kas 5 sekundes parašo pranešimą su nurodytu išsijungimo laiku po kurio <b>Papildomas</b> agentas turi baigti darbą</li> <li><b>Papildomas</b> agentas, gavęs žinutę, patvirtina, kad gavo žinutę perspausdindamas žinutės turinį ir po nurodyto laiko parašo paskutinį pranešimą, kad baigia darbą su žinutės tipu <b>CONFIRM</b>.</li> <li><b>Pagrindinio</b> agento žinutės tipas : <b>INFORM</b></li> <li><b>Papildomas</b> agento žinutės tipas: <b>INFORM ;CONFIRM</b>.</li> </ul>
3		<ul style="list-style-type: none"> <li>Sukurti agentus <b>Pagrindinis</b>, <b>Papildomas</b> ir <b>Papildomas2</b>.</li> <li><b>Pagrindinis</b> agentas kas 5 sekundes parašo pranešimą agentui <b>Papildomas</b> su nurodytu išsijungimo laiku po kurio <b>Papildomas</b> agentas turi baigti darbą ir išspausdina tekstą nurodytą Argumentų eilutėje. Išsijungus <b>Papildomas</b> agentui, įsijungia agentas <b>Papildomas2</b>, o išsijungus <b>Papildomas2</b> agentui vėl įsijungia <b>Papildomas</b>.</li> <li><b>Papildomas</b> agentas, gavęs žinutę, patvirtina, kad gavo žinutę perspausdindamas žinutės turinį ir parašo savo vardą. Po nurodyto laiko parašo paskutinį pranešimą, kad baigia darbą su žinutės tipu <b>CONFIRM</b>.</li> <li><b>Papildomas2</b> agentas, gavęs žinutę, patvirtina, kad gavo žinutę perspausdindamas žinutės turinį ir parašo savo vardą. Po nurodyto laiko parašo paskutinį pranešimą, kad baigia darbą su žinutės tipu <b>CONFIRM</b>.</li> <li><b>Pagrindinio</b> agento žinutės tipas : <b>INFORM</b></li> <li><b>Papildomas</b> agento žinutės tipas: <b>INFORM ;CONFIRM</b></li> <li><b>Papildomas2</b> agento žinutės tipas: <b>INFORM ;CONFIRM</b></li> <li>Argumentų eilutė turi atrodyti taip:</li> <li>„&lt;<b>Pagrindinis</b> agentas&gt;(&lt;pranešimas&gt;&lt;intervalas&gt;)“</li> </ul>