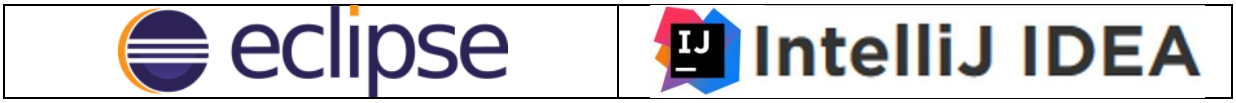
1 Preparation of environment

Before everything else please open our project in your Java Integrated Development Environment:

1.1 Available instructions:



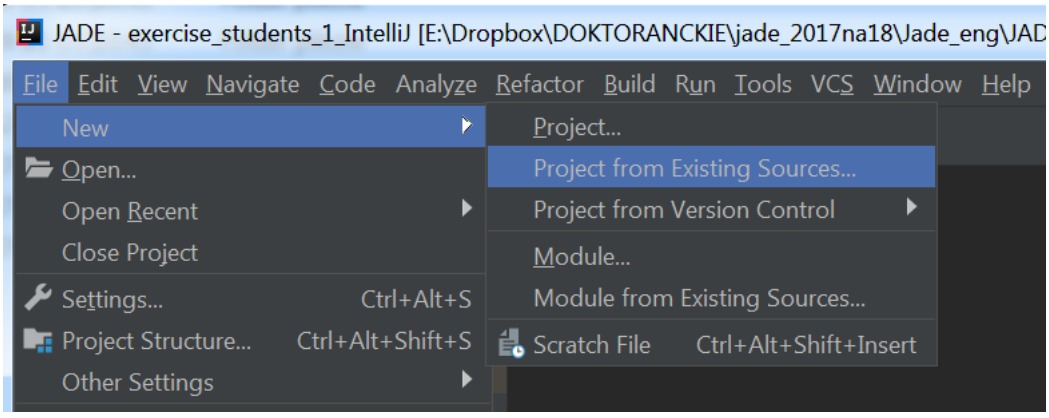
1.1.1 Configuration in IntelliJ:

This tutorial step by step is how to import existing eclipse's project to IntelliJ, so if You know how to import, skip until section Exercise 0.

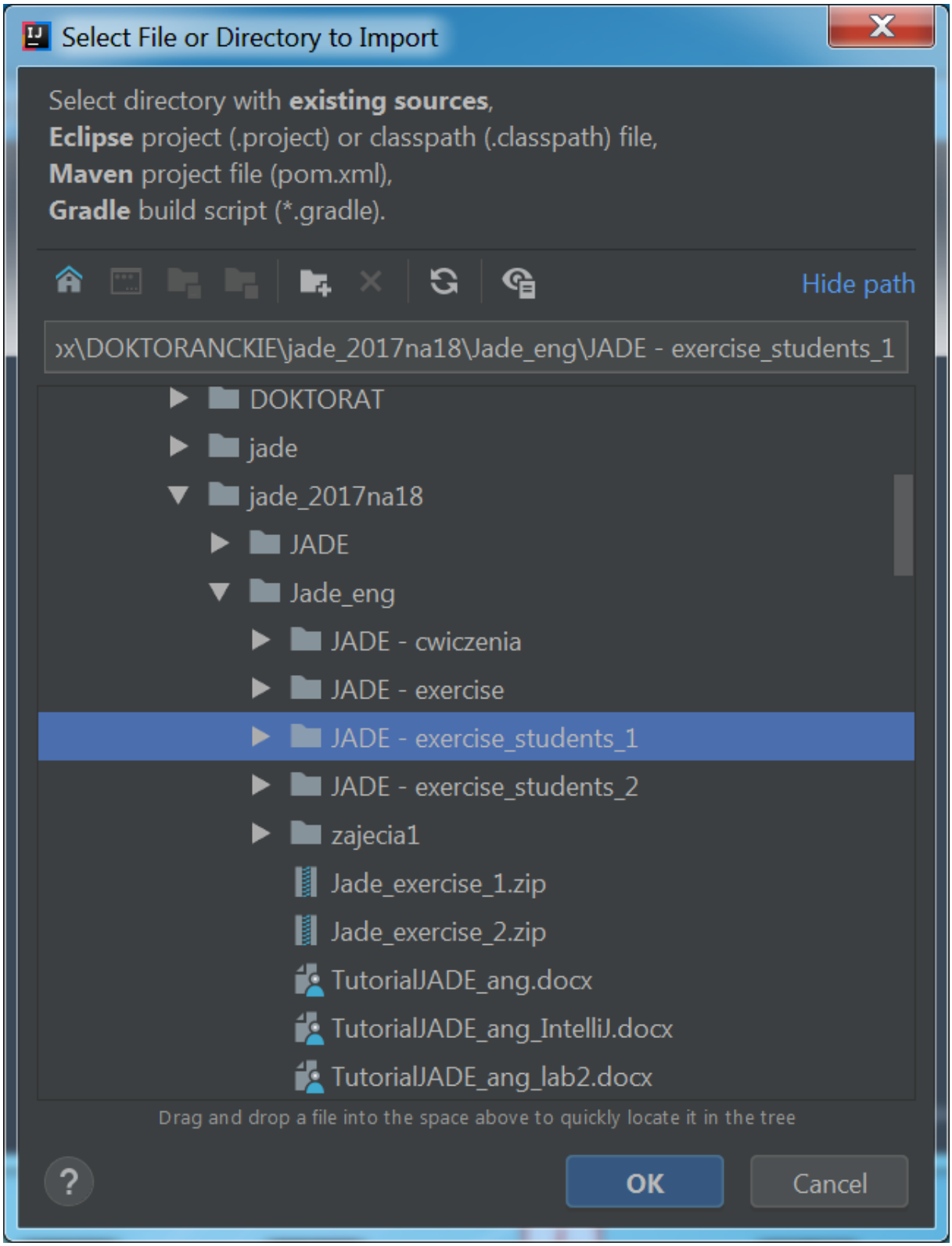
1. Let's run IntelliJ:

2. Let's unpack our package
3. Import Project

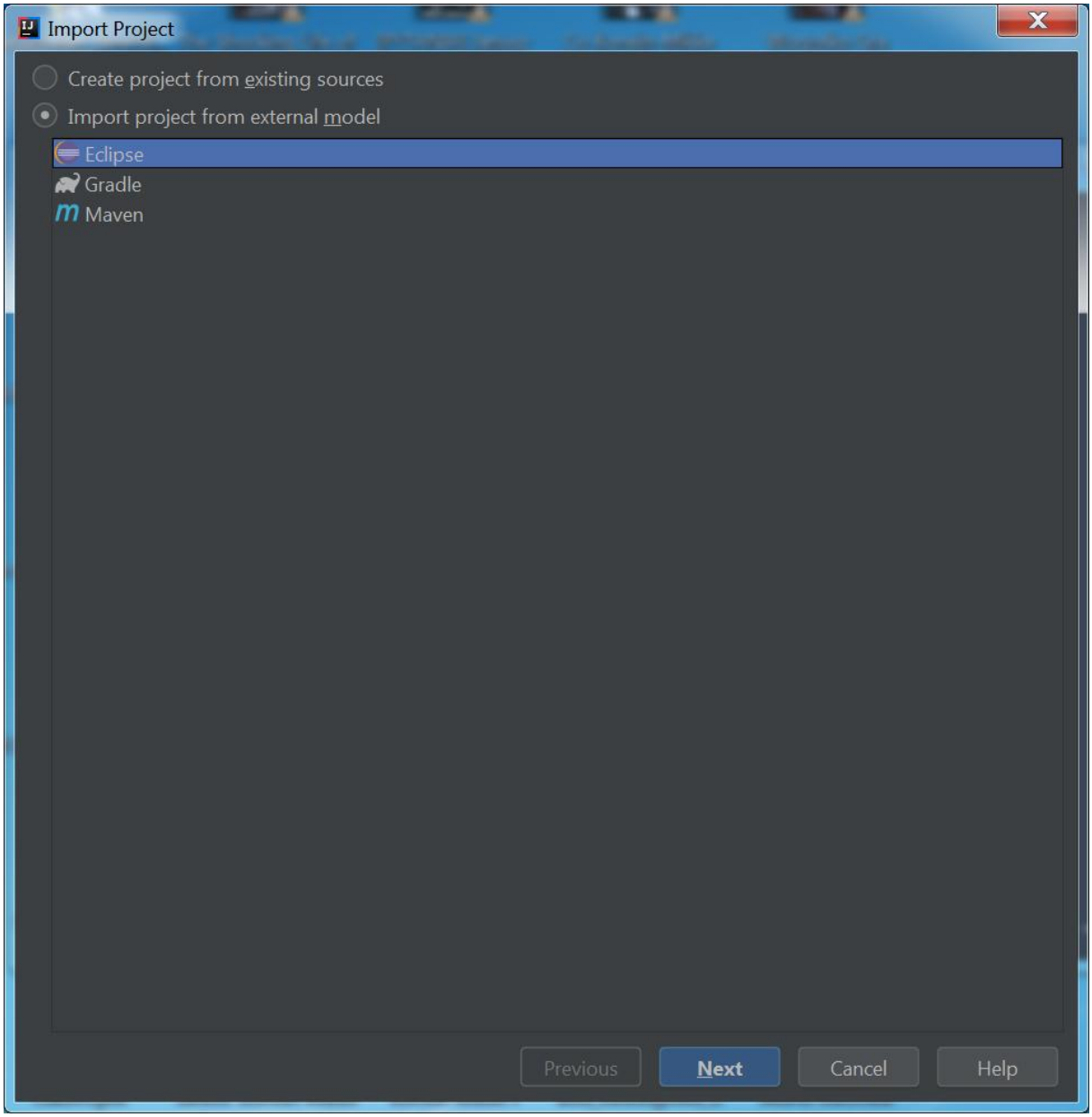
   a. If somebody does not see that window importing can be selected: File->New->Project from Existing Sources:



4. Choose path to unpacked project (it is possible to paste path to project), then OK:

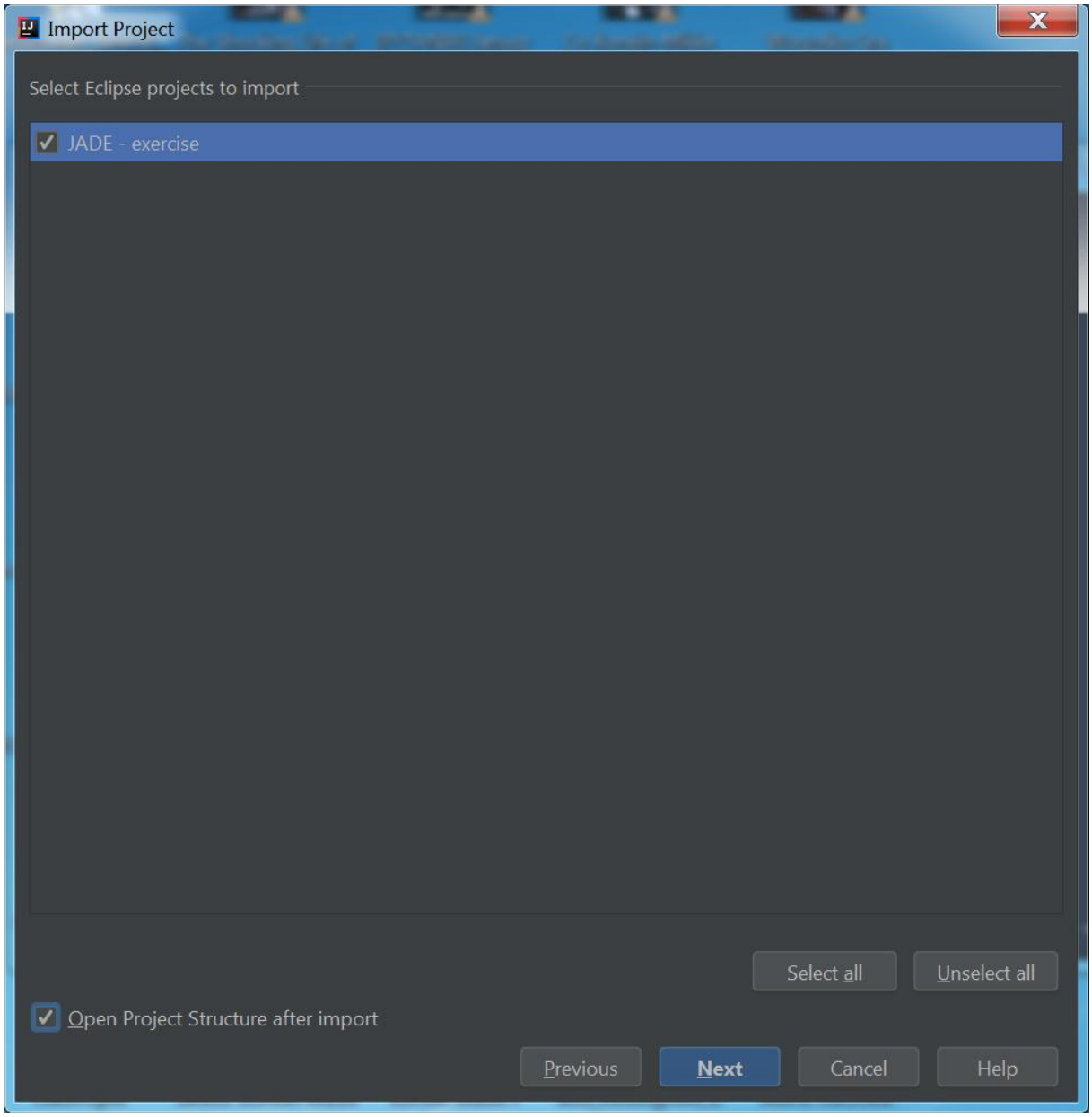


5. Choose "Import project from external model" and choose Eclipse, then Next:



6. In window "Select Eclipse projects directory" You can press just Next, or change something.

7. You should see project:

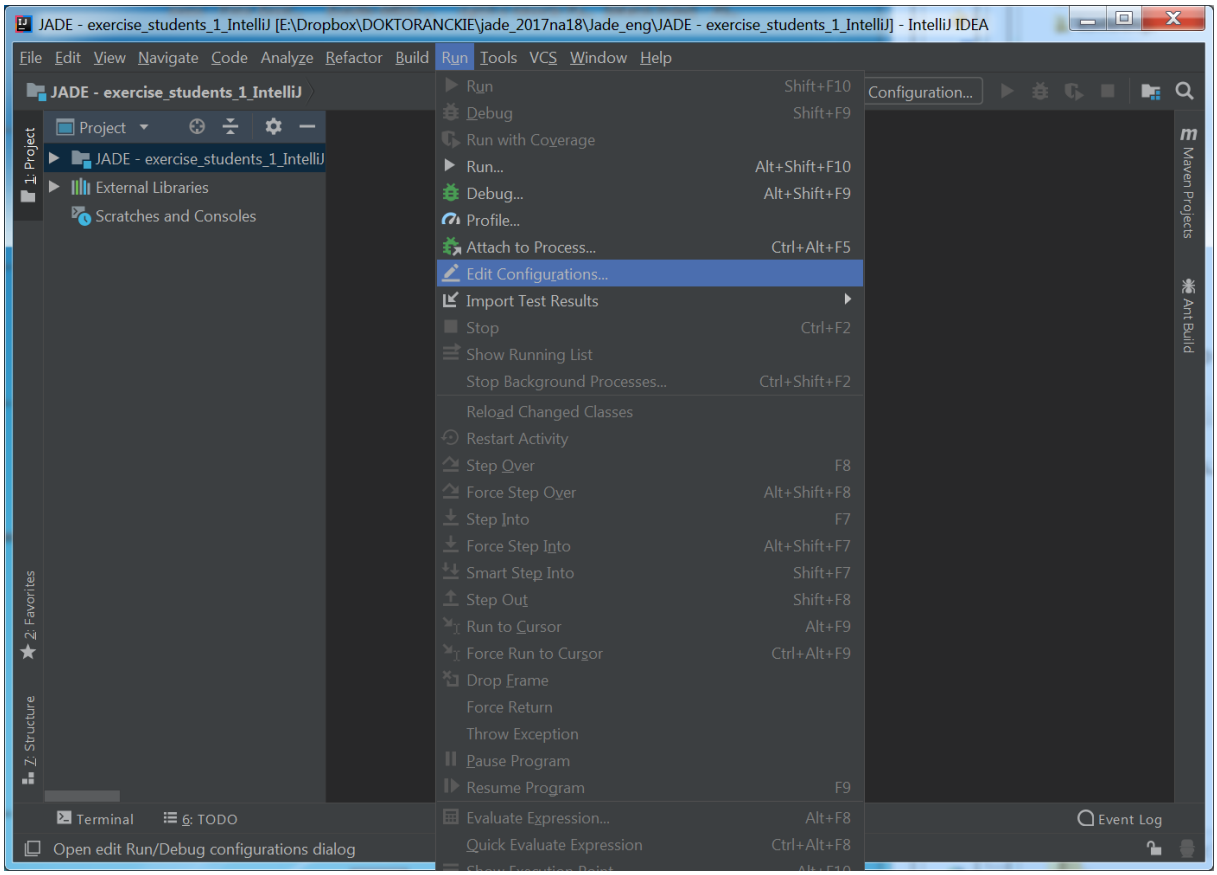


select "Open Project Structure after import", then Next

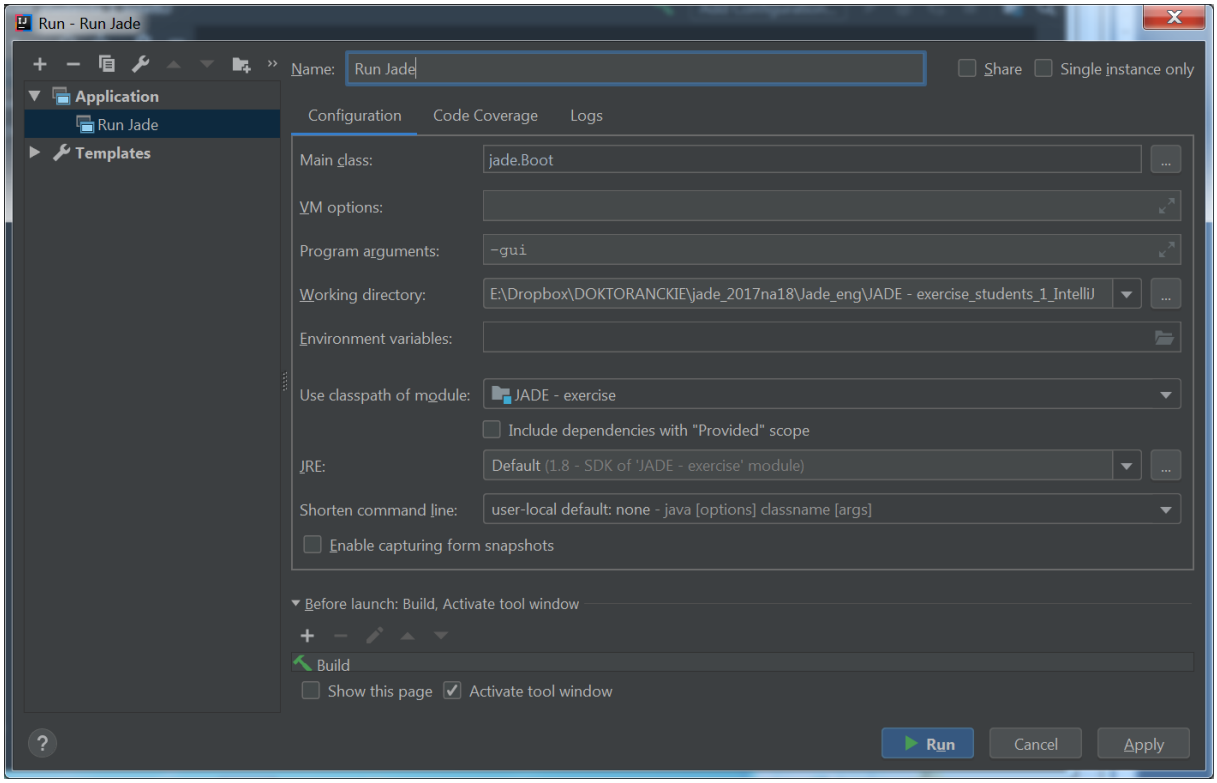
8. In window "Choose project code style" You can press just Next, or change something.
9. In next window You probably will see possibility with selecting JDK, just press Finish.

1.1.2 Running application in IntelliJ:

1. Choose Run -> Edit Configurations...



2. In a new window press +, then choose Application, and type in Main class and Program arguments what You see below:

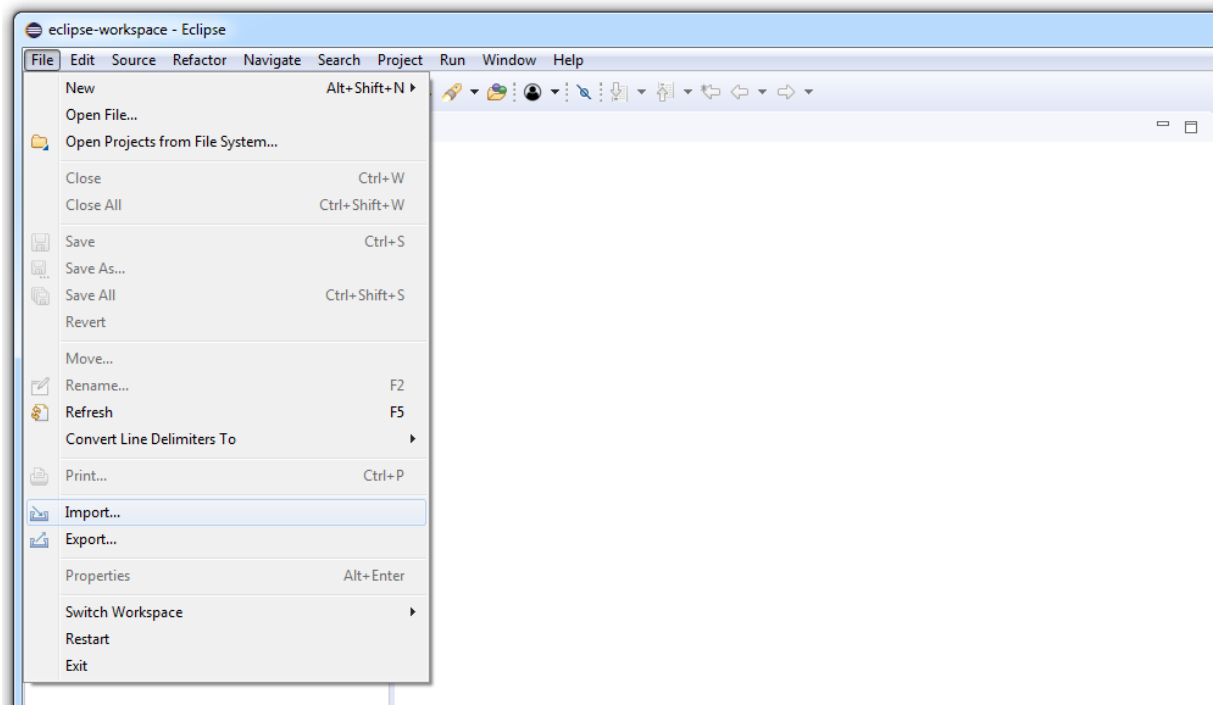


3. Press Run.

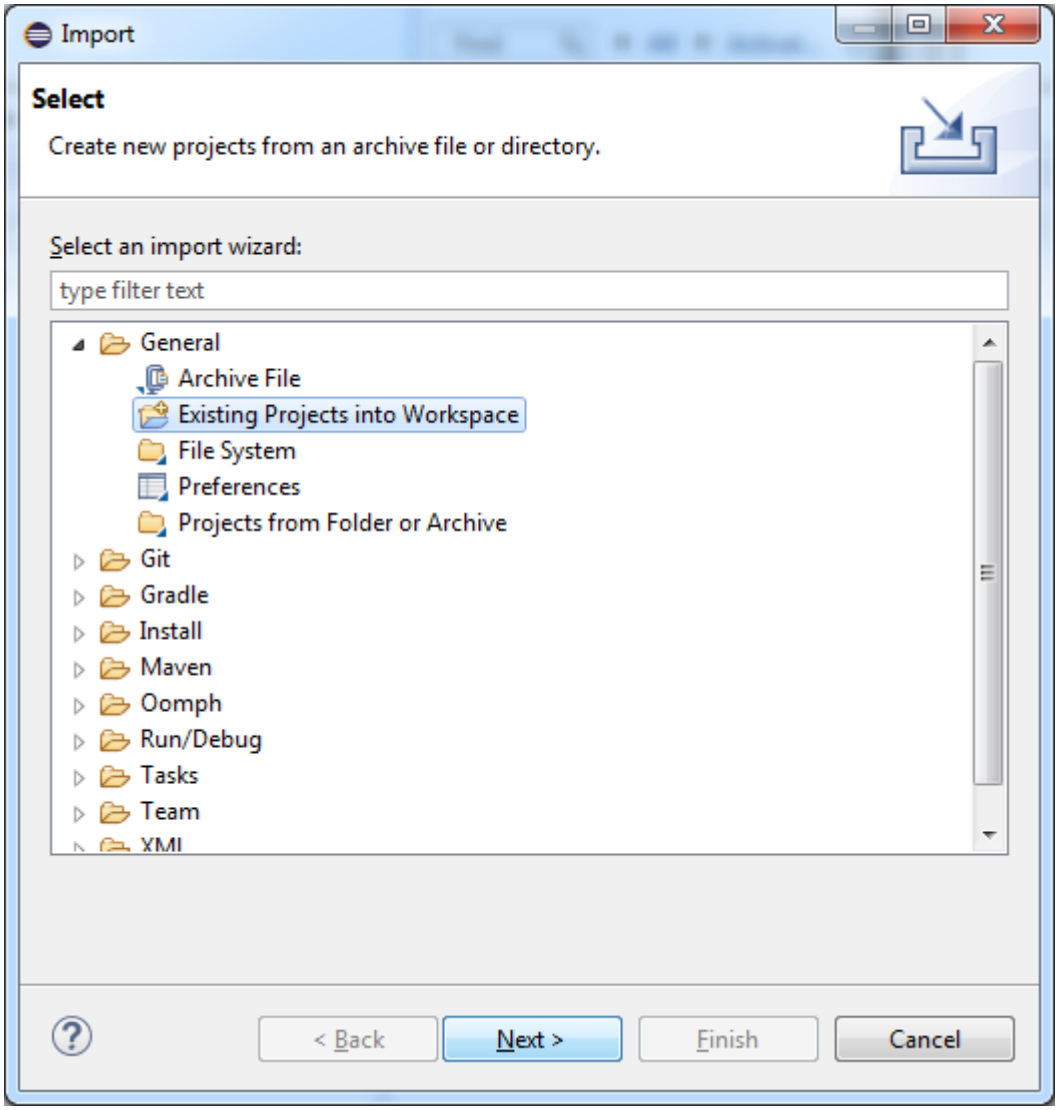
4. When we will have problems with port we would have to use another port with argument: **-host 192.168.2.9 -port 12344**
5. In other problems it can help if we download newest version of JADE framework: http://jade.tilab.com/download/jade/

1.1.3 Configuration in Eclipse:

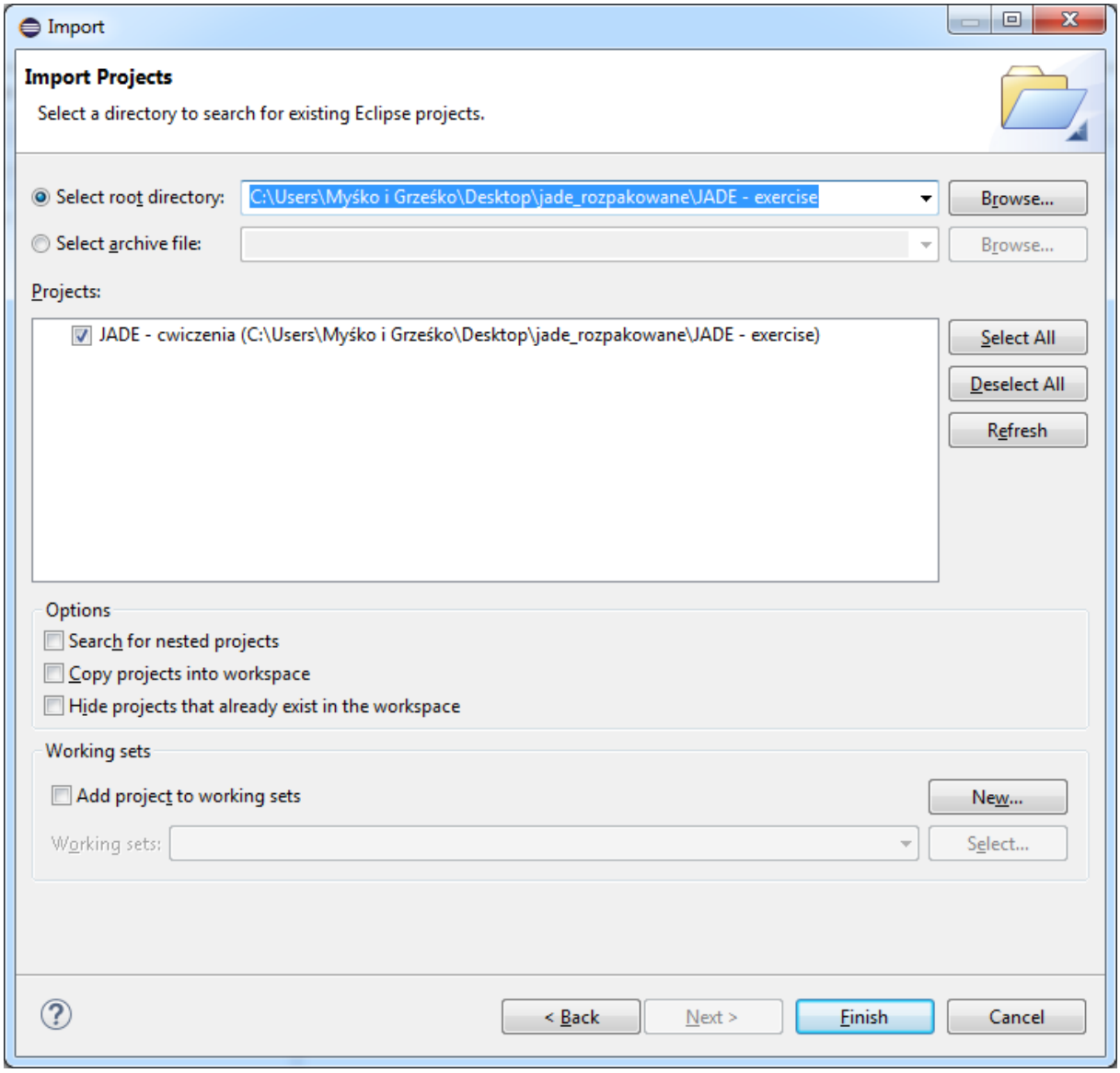
1. Let's run Eclipse:
2. Let's unpack our package
3. *File ->Import*



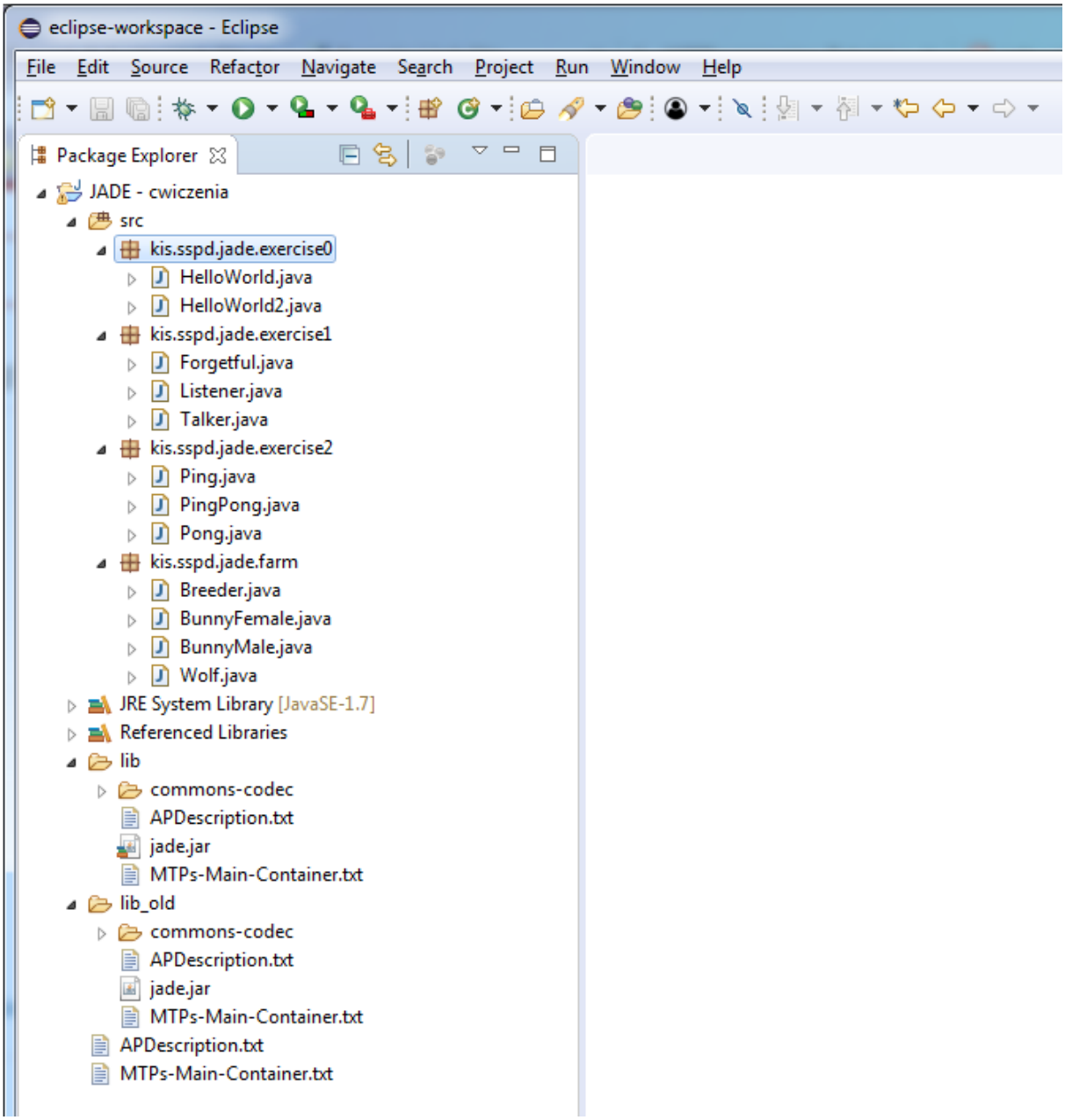
4. General -> Existing Project into Workspace -> Next:



5. Browse button, in which we have to choose path of our unpacked archive and project, after that press Finish:



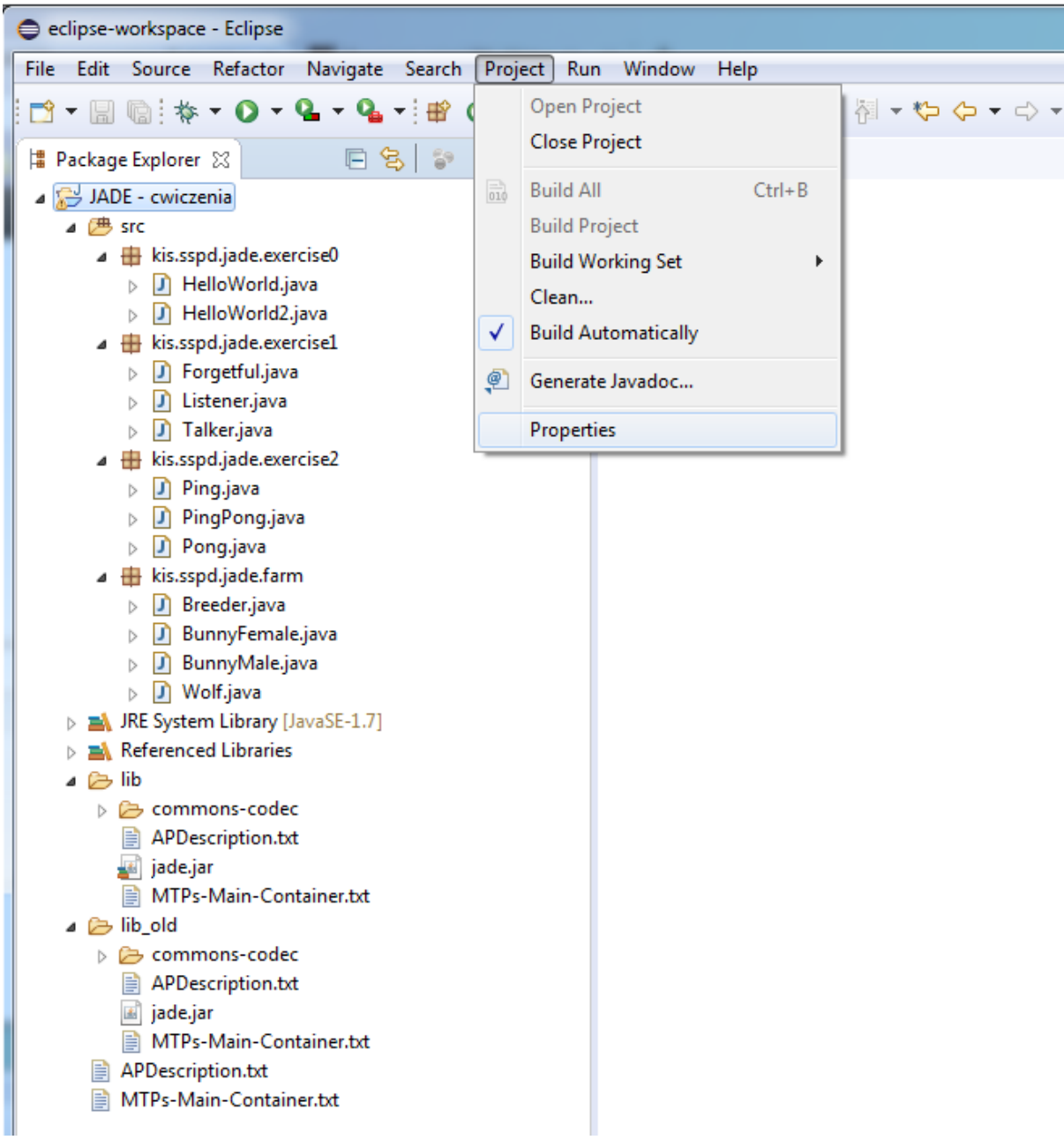
6. After that we should see project with four packages:



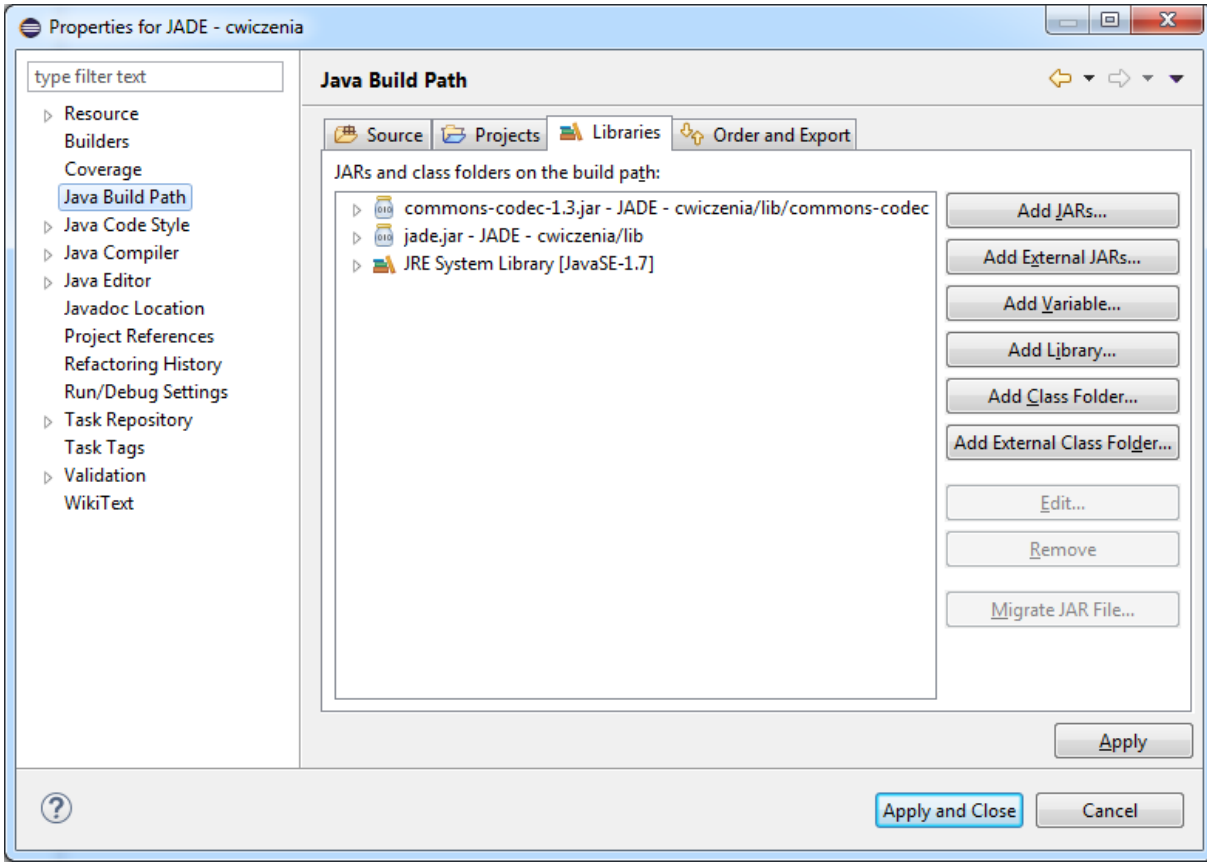
2.1.1 Running application in Eclipse

Let's try to run project, if you see any problems please check:

1. if all libraries are properly added:
    a. *Project ->Properties*



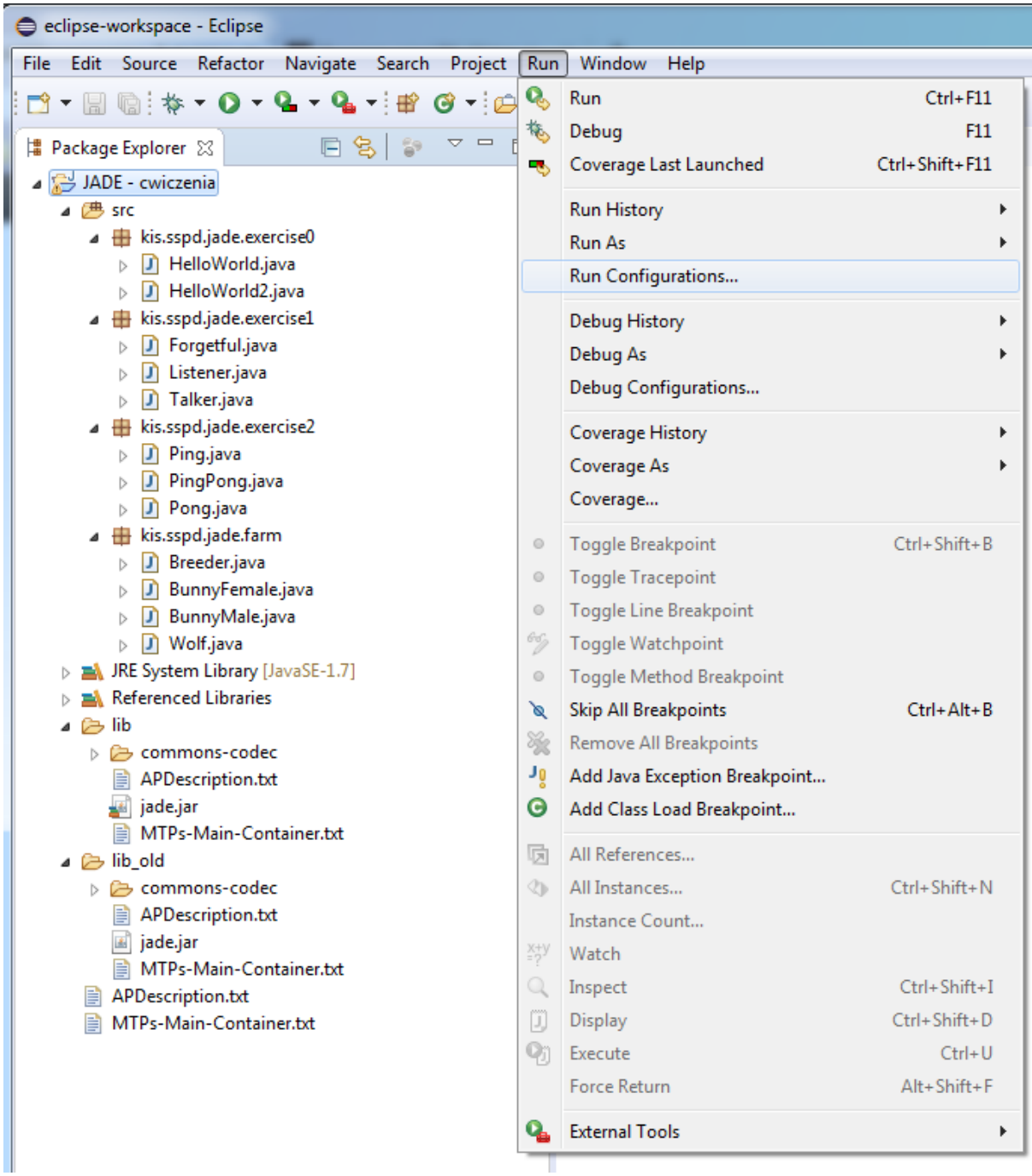
b. *Java Build Path ->Libraries*:



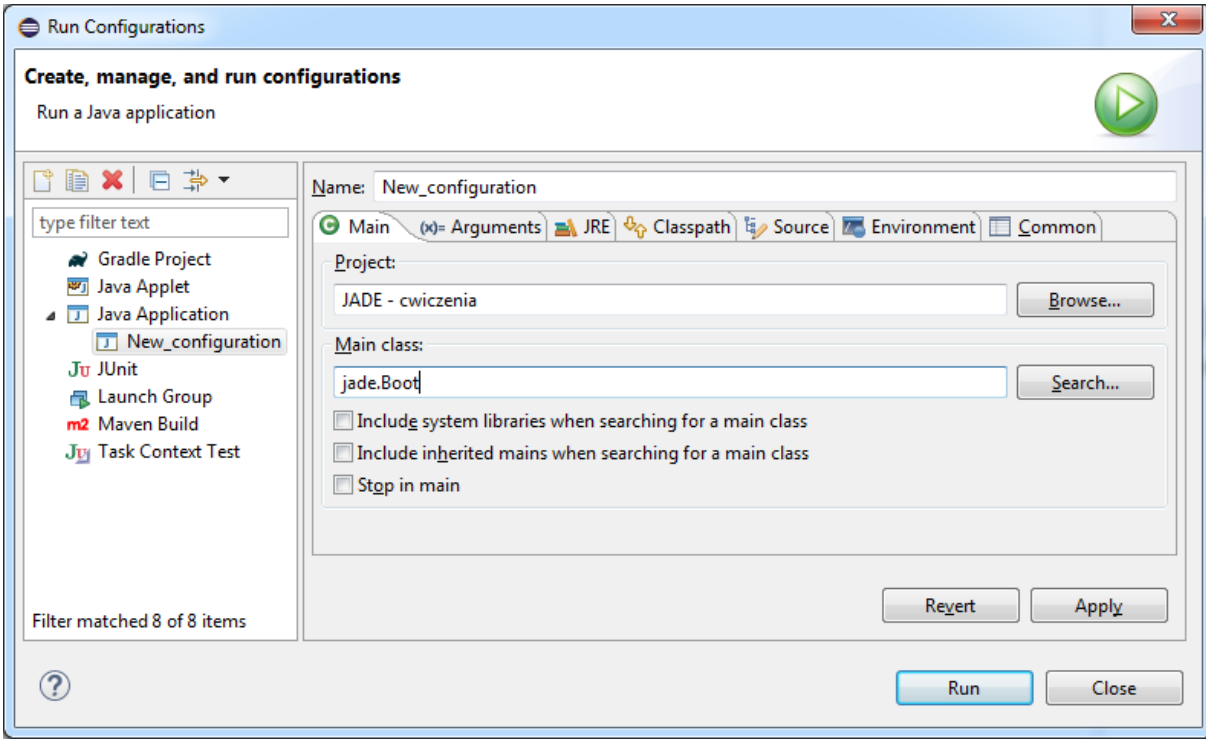
c. If we don't have those libraries we have to add them manually

2. Our running configuration and program arguments are correct:

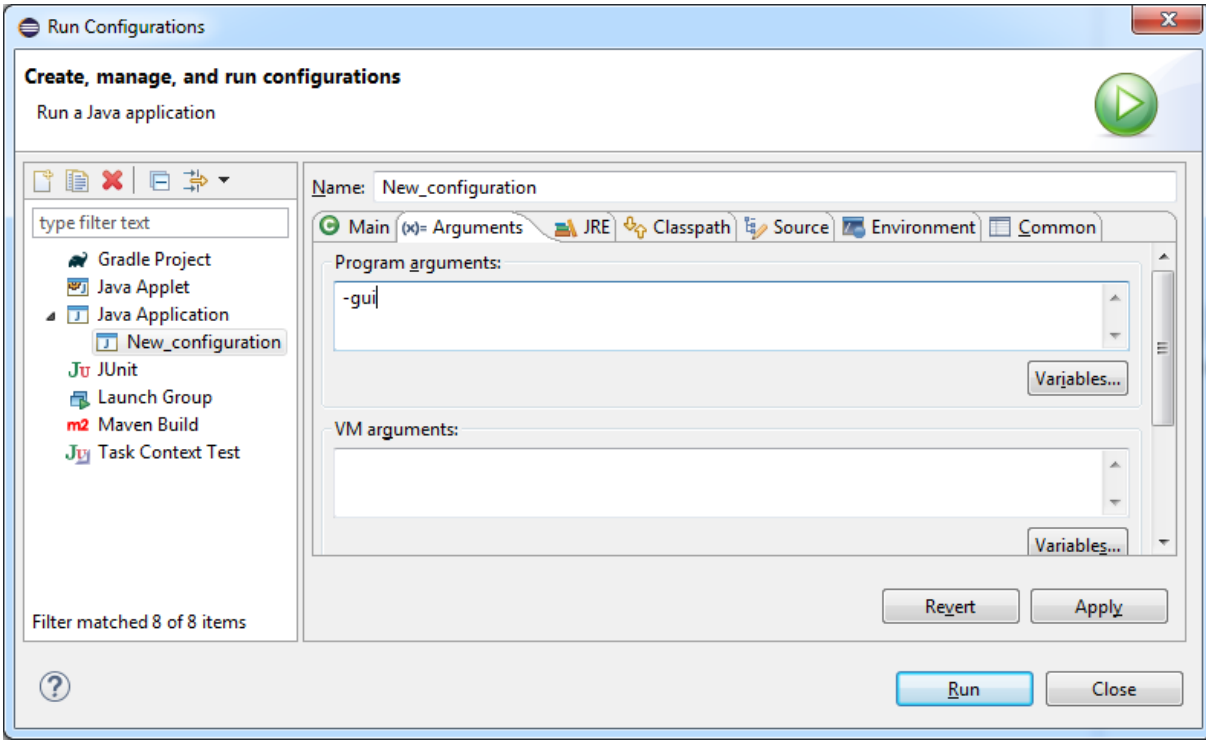
a. *Run ->Run Configurations ...*



b. *Java Application ->New*, main class should be **jade.Boot**



c. *Arguments ->Program arguments*, we need to add **-gui**:



d. We can also set some name different than New_configuration

e. When we will have problems with port we would have to use another port with argument: **-host 192.168.2.9 -port 12344**

3. In other problems it can help if we download newest version of JADE framework: http://jade.tilab.com/download/jade/

2 Exercise 0

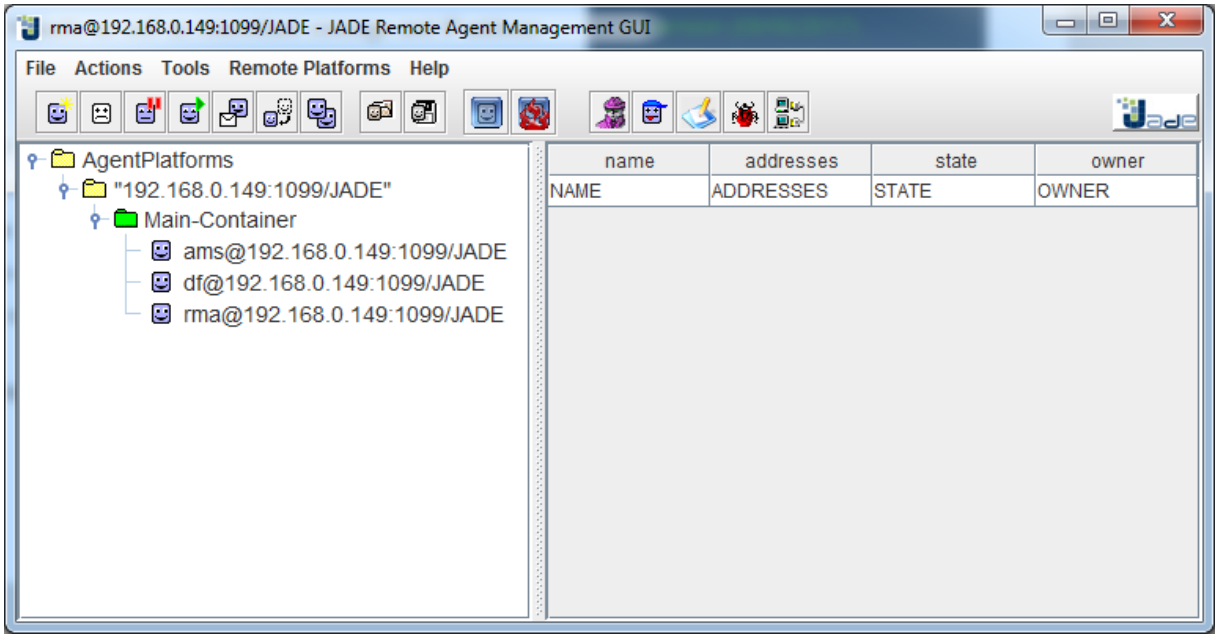
2.1 The JADE Environment

What is **JADE**? JADE (Java Agent DEvelopment Framework) is a software Framework fully implemented in the Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of graphical tools that support the debugging and deployment phases. It includes:

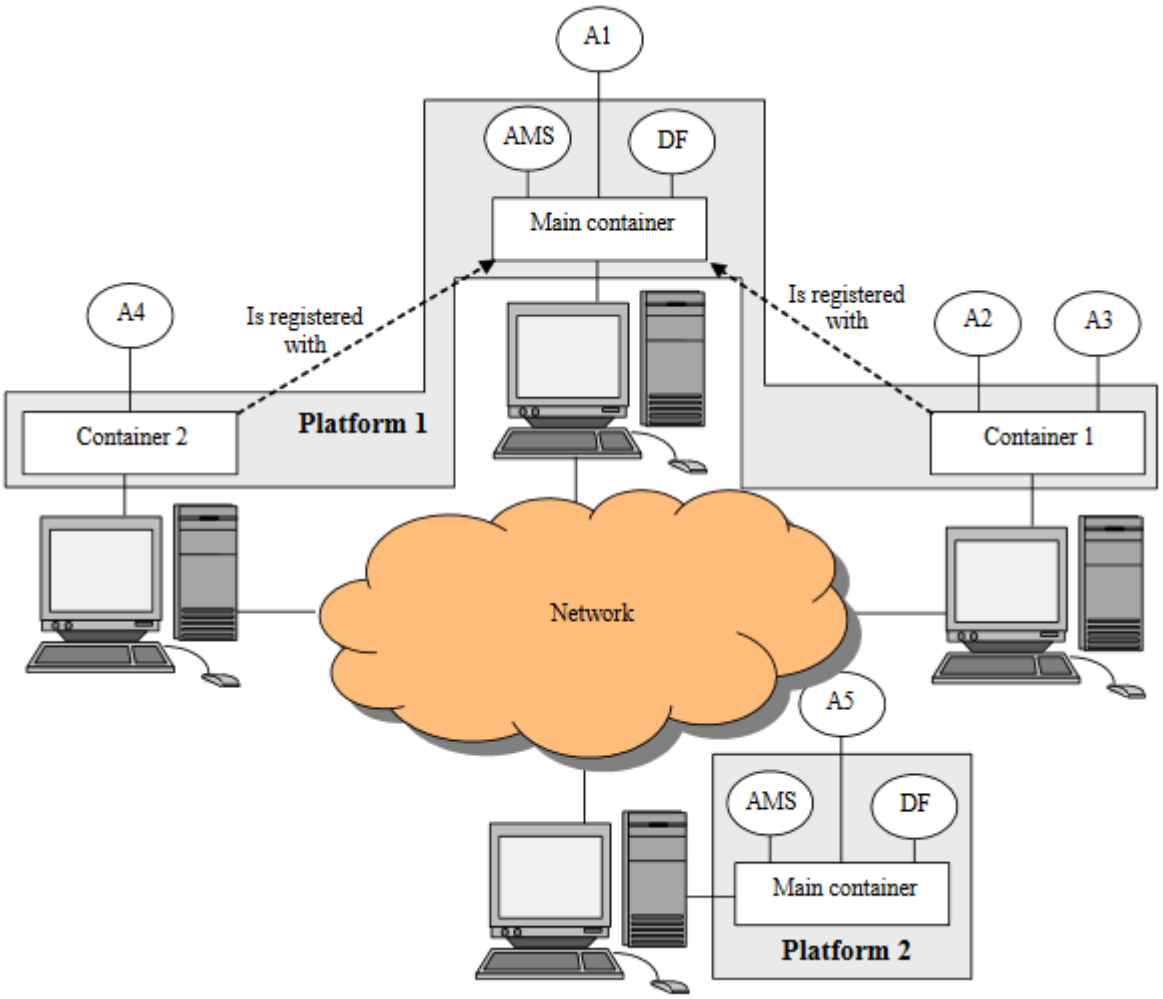
1. A runtime environment where JADE agents can “live” and that must be active on a given host before one or more agents can be executed on that host.
2. A library of classes that programmers have to/can use (directly or by specializing them) to develop their agents.
3. A suite of graphical tools that allows administrating and monitoring the activity of running agents.

2.1.1 JADE GUI:

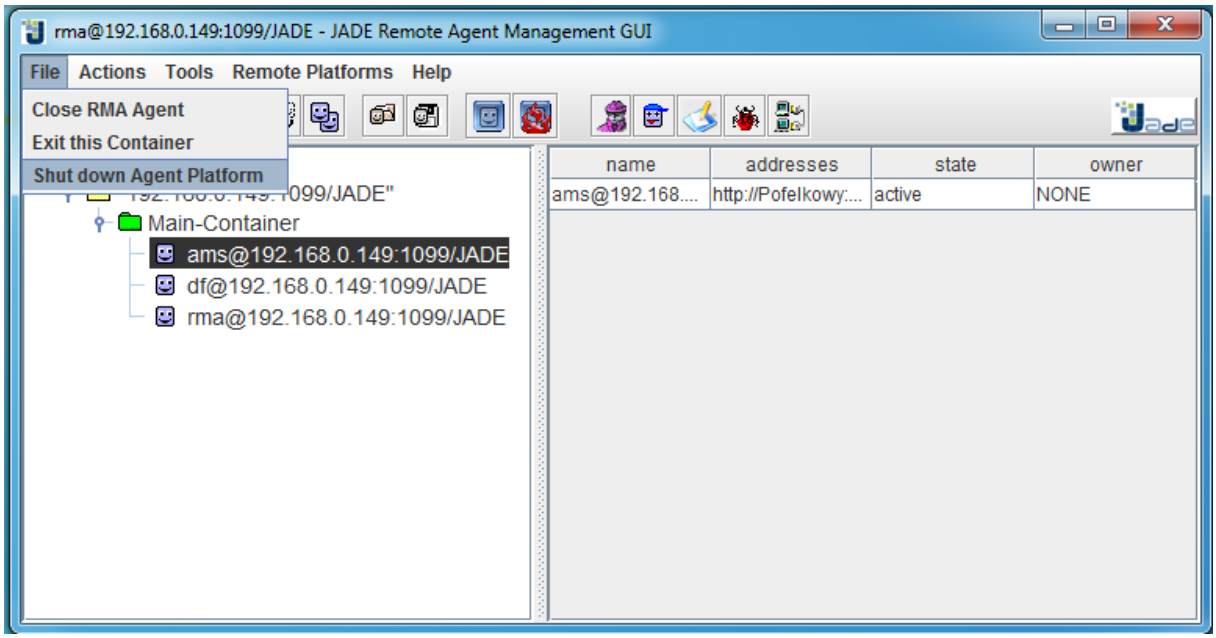
If everything is OK program's named RMA window should appear, after full expanding of *AgentPlatforms* we will see "*Main-Container*" directory:



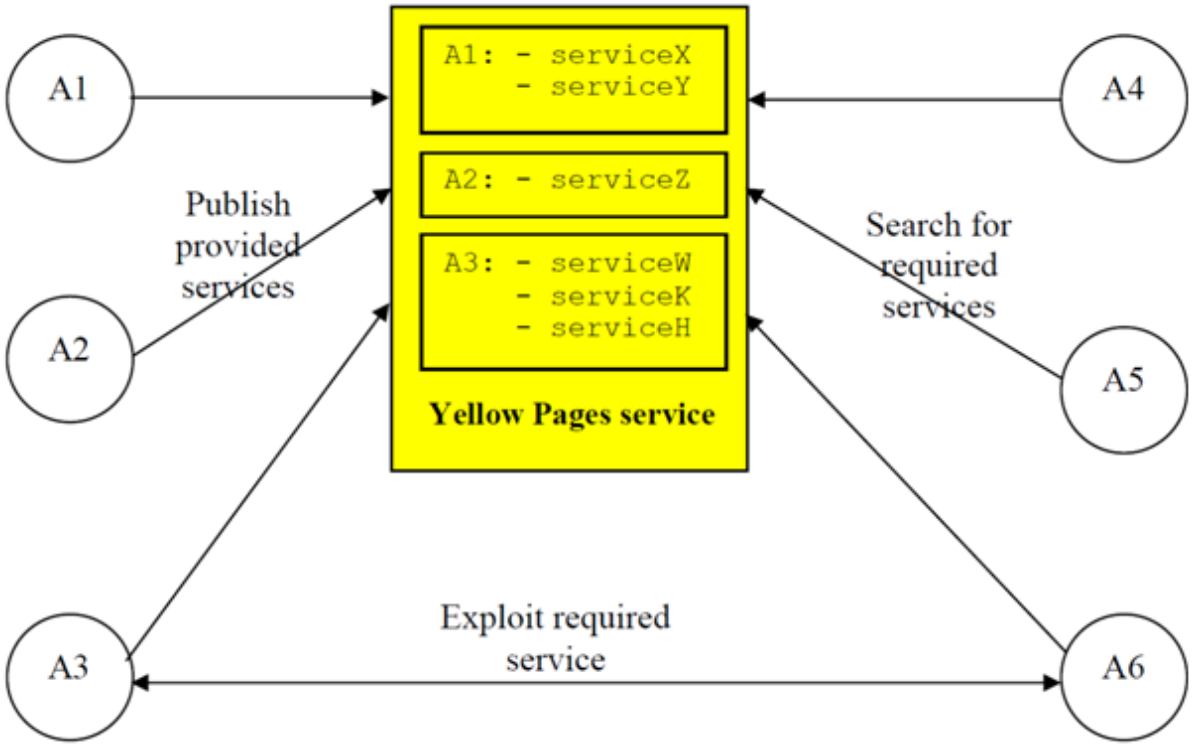
- For single JVM is connected single agent platform by default. Newly added agents are added to already existing platform. At the picture below we can see Containers and Platforms:



- RMA is not agent platform. In order to close JADE (e.g. for recompiling purpose) in RMA we have to click *File ->Shut down Agent Platform*. If we close RMA without closing platform we have to manually kill processes from Eclipse.



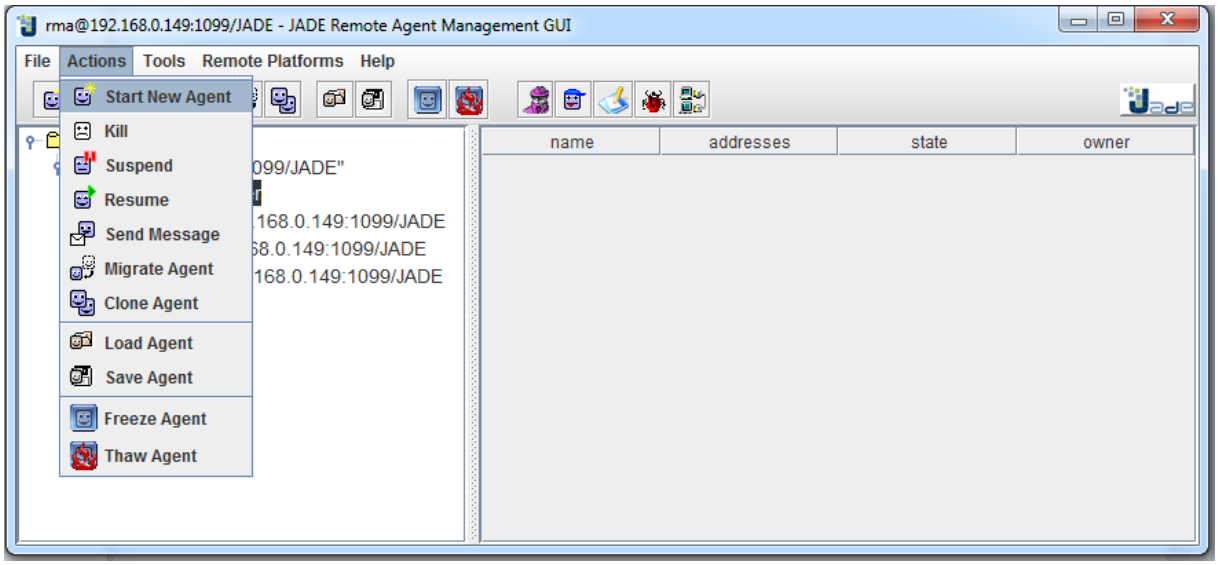
- Each running instance of the JADE runtime environment is called a Container as it can contain several agents. The set of active containers is called a Platform. A single special *Main container* must always be active in a platform and all other containers register with it as soon as they start. It follows that the first container to start in a platform must be a main container while all other containers must be "normal" (i.e. non-main) containers and must "be told" where to find (host and port) their main container (i.e. the main container to register with).
- After running the platform we can see 3 already running agents:
    - The **RMA** (Remote Monitoring Agent) allows controlling the life cycle of the agent platform and of all the registered agents. The distributed architecture of JADE allows also remote controlling, where the GUI is used to control the execution of agents and their life cycle from a remote host.
    - The **AMS** (Agent Management System) that provides the naming service (i.e. ensures that each agent in the platform has a unique name) and represents the authority in the platform (for instance it is possible to create/kill agents on remote containers by requesting that to the AMS).
    - The **DF** (Directory Facilitator) that provides a Yellow Pages service by means of which an agent can find other agents providing the services he requires in order to achieve his goals.



Please spend few minutes to explore RMA, because we are going to use its during rest of the day.

2.2 Hello word

After selecting "Main-Container" please clik *Actions ->Start New Agents* in order to create new agent:



Then we have to type Agent name and choose agent's class name, in order to choose class we should click **...** then select *kis.sspd.jade.exercise0.HelloWorld*, after that we can create the agent by clicking *OK*.

- What happened?
- What the agent is doing?
- What can we see in console?
- Let's kill our running agent (*Actions ->Kill*), what can we observe?
- Is it possible to kill one of 3 default agents?
- Is it possible to kill main container?
- Can we create multiple agents with the same name?
- Let's look to the HelloWorld agent's code.

2.3 Hello word2

Please run agent *HelloWorld2*

- What do we see in console just after creating the agent? What do we see after few seconds?
- Let's try to kill the agent manually
- Let's look to the HelloWorld agent's code.

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*,	`void`

	suspended or *waiting* to *deleted* state within Agent Platform Life Cycle, thereby destroying the agent.	
addBehaviour(Behaviour b)	This method adds a new behaviour to the agent.	`void`

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

3. Exercise 1

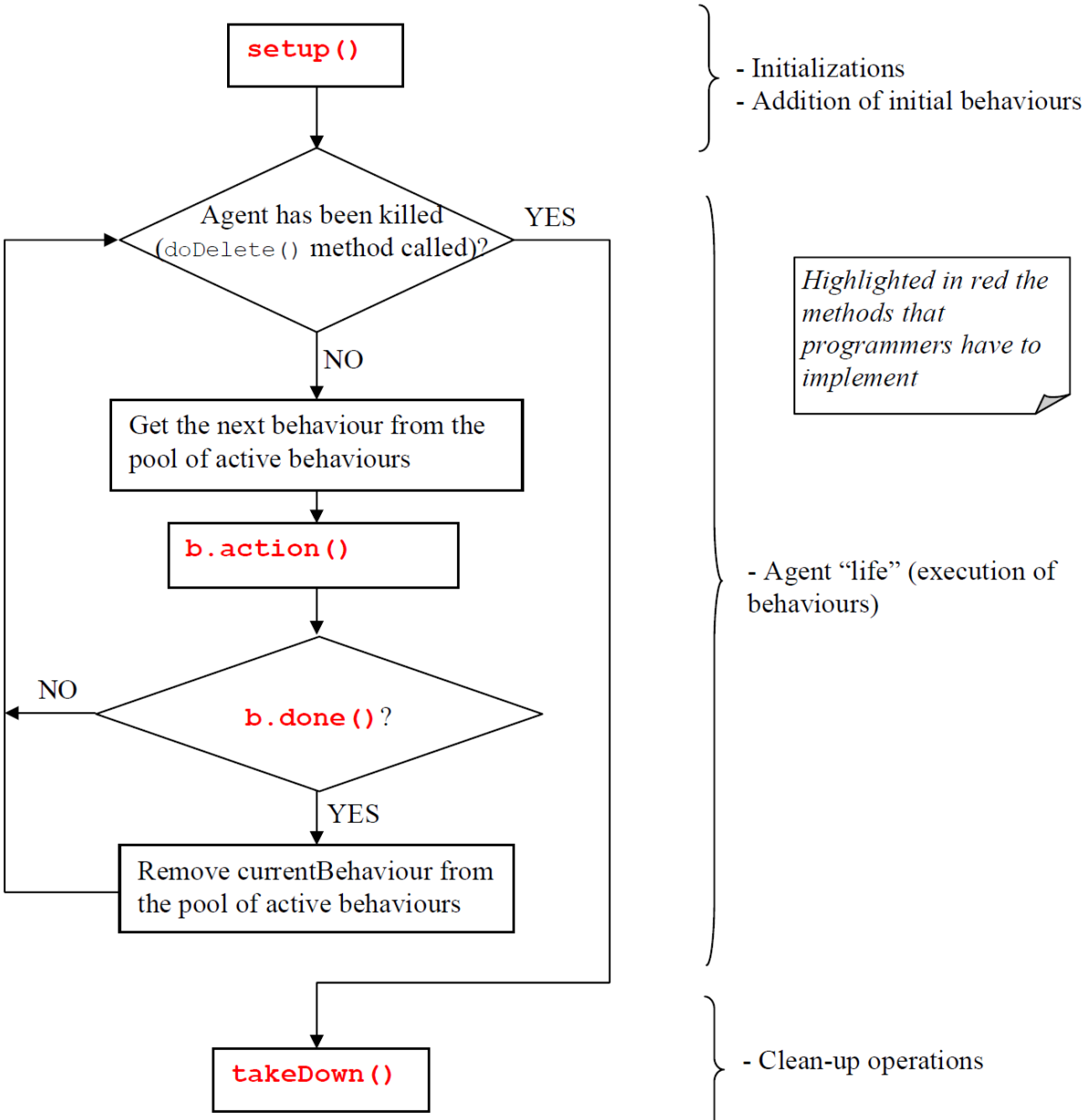
3.1 Agent behaviour and lifecycle - Forgetful

The actual job an agent has to do is typically carried out within "behaviours". A behaviour represents a task that an agent can carry out and is implemented as an object of a class that extends *`jade.core.behaviours.Behaviour`*

Each class extending Behaviour must implement the **`action()`** method, that actually defines the operations to be performed when the behaviour is in execution and the **`done()`** method (returns a boolean value), that specifies whether or not a behaviour has completed and have to be removed from the pool of behaviours an agent is carrying out.

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

3.1.1 Agent lifecycle



Agent Thread path of execution

- Agent lives until the method **`doDelete()`** is not called, even if pool of behaviours is empty.
- The only behaviour, which would return **`true`** by function **`done()`** are removed from the pool

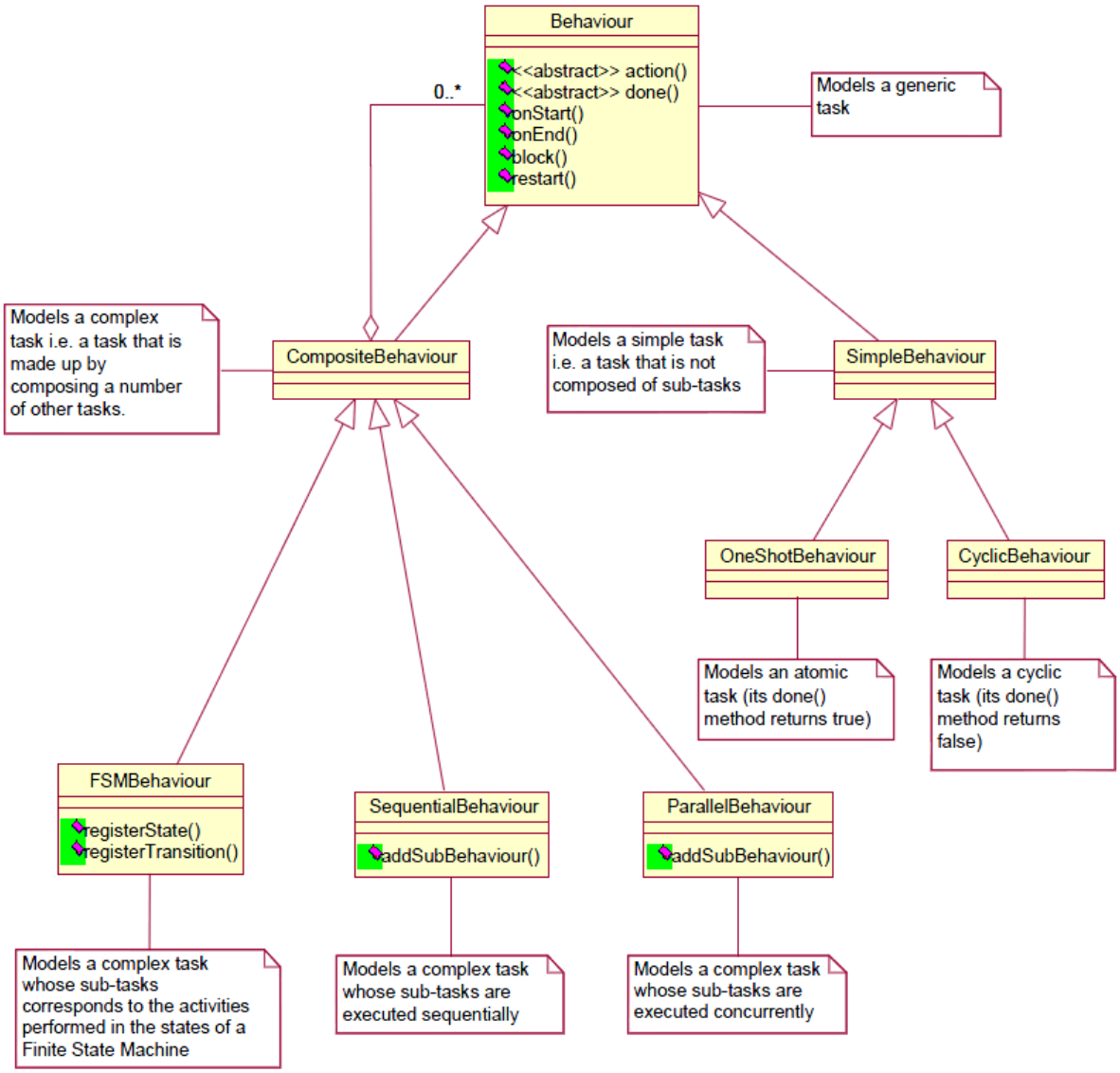
3.1.2 The Forgetful agent

Let's look to the agent's code.

- What can be returned by function **`done()`** in **`OneShotBehaviour`**? When it is going to be removed from behaviour pool?
- What can be returned by function **`done()`** in **`CyclicBehaviour`**? When it is going to be removed from behaviour pool?

- What does it means for those behaviours?
- What would be the output of the agent in your opinion?
- Let's run the agent and check
- Uncomment the line **`myAgent.addBehaviour(new SingleBehaviour());`** in class **`CyclicBehaviour2`**
- Comment **`myAgent.addBehaviour(new SingleBehaviour());`** in class **`CyclicBehaviour1`**
- What is the output?

UML diagram of types of defined behaviours in JADE:



3.2 The Listener agent

- Run the Listener Agent
- Send do the running agent message: *Actions ->Send Message*
- Check the processor usage
- Add one more listener agent and check usage now

- Check usage after killing them both softly
- Add the code below after the if-statement:

```
else {
    block();
}
```

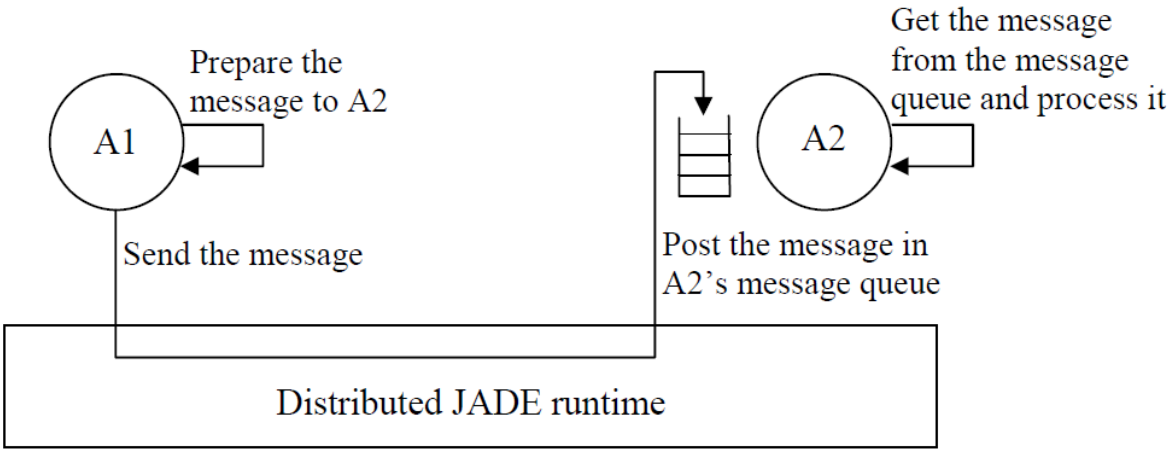
- How does the CPU usage looks now?
- Let's add to the else-statement at the end line:
  `System.out.println("Blocked.");`
- When exactly the agent is being blocked?

3.3 The Talker agent

- Run the Listener agent
- Run the Talker agent adding in field *Arguments* name of the running Listener agent.
- Let's take a look into code of both agents. What types of arguments are being sent to the Listener agent

3.4 Way of communication

One of the most important features that JADE agents provide is the ability to communicate and interactions protocols. The communication paradigm adopted is the asynchronous message passing. Each agent has a sort of mailbox (the agent message queue) where the JADE runtime posts messages sent by other agents. Whenever a message is posted in the message queue the receiving agent is notified. If and when the agent actually picks up the message from the message queue to process it is completely up to the programmer however. All received messages are being hold in FIFO queue, but it is available mechanism to search in the queue with patterns. Messages are objects containing lots of information and having many methods to help communicate (more details soon).



The JADE asynchronous message passing paradigm

4 Exercise 2

- Run the PingPong agent
- Take a look to the source code of all tree agents: Ping, Pong, PingPong
- How they communicate?
- How new agents are being created? Who is their creator?
- What would happen (or rather not) if we don't call the **`start()`** method

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
- Another JADE tutorial, which added some information to the document: http://jade.tilab.com/doc/administratorsguide.pdf
- Robert's tutorial in polish