Jade program that creates N agents and each agent will print its ID

ABOUT THE CODE

The Imports

```
import jade.core.Agent;
import jade.core.Profile;
import jade.core.ProfileImpl;
import jade.core.Runtime;
import jade.wrapper.ContainerController;
import jade.wrapper.StaleProxyException;
import jade.wrapper.AgentController;
import java.util.Scanner;
```

figure: imports

The profile class

The profile class allows retrieving configuration-dependant classes.

It is abstract, so it cannot be instanciated. We then create an instance of *profileImp* instead.

This class, **ProfileImpl,** extends the Profile class. It allows the JADE core to retrieve configuration-dependent classes and boot parameters.

```
p.setParameter(Profile.MAIN_HOST , "localhost");
p.setParameter(Profile.GUI, "true");
ContainerController cc=rt.createMainContainer(p);
```

The setParameter allows us to assign values to some property names.

We set the main host adress to be the local host, which is our machine in this case.

And the GUI set to true allows us to view the graphical interface.

Container controller is proxy class allowing access to a jade agent container whose instance we also created.

It is possible to request services in process agent containers by invoking methods on instances of this class.

```java
public class Main extends Agent
{
    protected void setup() { System.out.println("I am an Agent  "+getLocalName()); }

    public static void main(String[] args) {
        //Runtime structure so that it knows how to execute our code provided by jade
        Runtime rt=Runtime.instance();

        //profile class so that we can pass arguments, its abstract, so we call its child
        Profile p=new ProfileImpl();

        //main container has address of local host
        p.setParameter(Profile.MAIN_HOST , s1: "localhost");
        p.setParameter(Profile.GUI, s1: "true");
        ContainerController cc=rt.createMainContainer(p);

        Scanner enterUser = new Scanner(System.in);
        System.out.println("Enter a number of agents :");
        int n = enterUser.nextInt();
```

figure: code

The runtime instance

```java
Runtime rt=Runtime.instance();
```

We create an instance of runtime. This is so that it knows how to execute our code provided by jade. Instanciating of this class that should be then used to create agent containers.

The agent Controler

```java
AgentController ac;
```

Agent Controller manages all details of startup and communication with each agent. It can interact with one or more agents simultaneously. A client is not associated with a particular agent until it obtains a handle to it from the Agent Controller.

```java
ac = cc.createNewAgent(" "+i, "TheAgent", null);
ac.start();
```

CreateNewAgent creates a new JADE agent, running within this container. With parameters:

- *Name* – the unique name for the newly created agent (in our program its an integer value).
- *className* – which is name of the class that implements the agent
- *Arguments* – an object array eith initialization parameters

The start method will call the setup method for the agent.

So I created a loop that creates an instance of an agent whose characteristics are defined in a separate file of the className specified in the new agent parameters.

```
for(int i=1; i<=n; i++)
{
    AgentController ac;
    try {
        ac = cc.createNewAgent(" "+i, "TheAgent", null);
        ac.start();
    }
    catch(StaleProxyException e) {
        e.printStackTrace();
    }
```

Upon execution, we can personalize our number of agents through the input, and each agent will then print its name as follows:

```
Main
INFO: Service jade.core.messaging.Messaging initialized
Apr 19, 2022 8:49:37 PM jade.core.BaseService init
INFO: Service jade.core.resource.ResourceManagement initialized
Apr 19, 2022 8:49:37 PM jade.core.BaseService init
INFO: Service jade.core.mobility.AgentMobility initialized
Apr 19, 2022 8:49:37 PM jade.core.BaseService init
INFO: Service jade.core.event.Notification initialized
Apr 19, 2022 8:49:37 PM jade.mtp.http.HTTPServer <init>
INFO: HTTP-MTP Using XML parser com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl$JAXPSAXParser
Apr 19, 2022 8:49:37 PM jade.core.messaging.MessagingService boot
INFO: MTP addresses:
http://DESKTOP-1Q0NA2G:7778/acc
Apr 19, 2022 8:49:37 PM jade.core.AgentContainerImpl joinPlatform
INFO: ------------------------------------
Agent container Main-Container@172.20.10.3 is ready.
--------------------------------------------
Enter a number of agents :
5
I am agent 1
I am agent 2
I am agent 3
I am agent 4
I am agent 5
```

figure execution

Then in the main container, each of the agents will be added as follows:

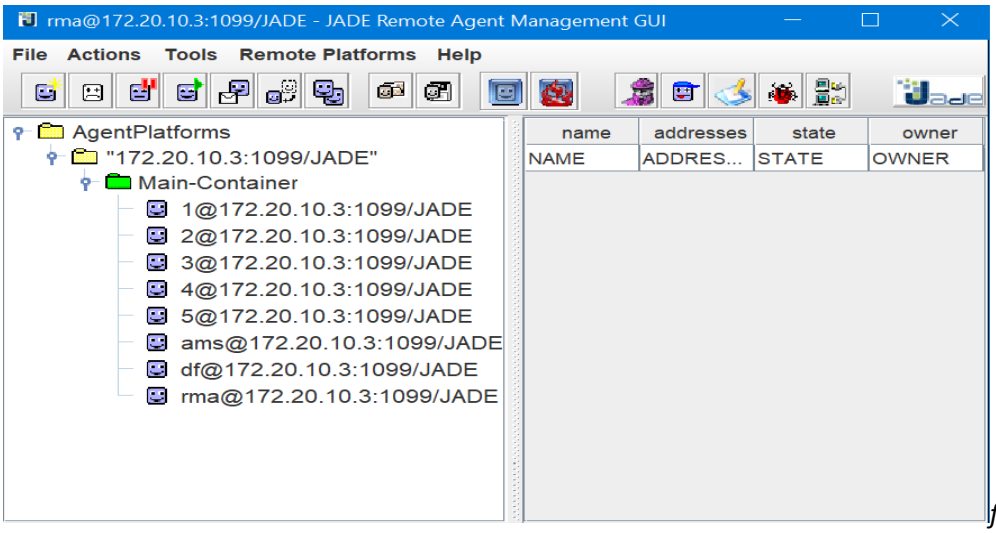

figure :container

<u>Reference</u>

1. https://jade.tilab.com/doc/api/jade/core/ProfileImpl.html

   date visited : 10/04/2022

2. https://jade.tilab.com/doc/api/jade/core/Runtime.html
   date visited : 10/04/2022