# **CREATION OF MULTIPLE AGENTS**

Definitions of classes and methods used:

## Class Chain

### **1. Java Runtime**

The runtime class is used to represent the running state of the Java virtual machine. It is used to encapsulate the Java virtual machine [process.](https://developpaper.com/tag/process/) The application cannot create its own runtime instance. If you want to obtain a runtime instance in the program, you can obtain the related runtime [object](https://developpaper.com/tag/object/) through the getruntime () [method.](https://developpaper.com/tag/method/) The specific methods are as follows: *Runtime run = Runtime.getRuntime();*

“runtime. Getruntime();” method creates an instance object of runtime, and calls the availableprocessors() method, freememory() method and maxmemory() method of the object respectively to print out the number of processors, free memory size and maximum available memory size of the current virtual machine.

### **2. Profileimpl**

Creates a profile object implementation. The profile is configured to launch a main-container on the localhost.

### -p.setParameter(Profile.**MAIN\_HOST** , "localhost");

Sets the address of the main-container launched to localhost. JADE is a distributed system. A platform can have containers on remote systems as well as locally.

### -p.setParameter(Profile.**GUI**, "true");

Starts up the rms agent when set to true. It displays the container tree in the RMA agent window. You see the main container with the df, ams and RMA agents along with the agents created.

* **RMA**: the Remote Management Agent which handles the GUI interface
* **ams**: the Agent Management Service - the core agent which keeps track of all JADE programs and agents in the system
* **DF**: the Directory Facility, a yellow page service, where agents can publish their services.

**3. ContainerController:**

This class is a Proxy class, allowing access to a JADE agent container. Invoking methods on instances of this class, it is possible to request services from in-process agent containers. This class must not be instantiated by applications. Instead, use the createMainContainer() method in class Runtime.

#### 4. AgentController

This interface is a Proxy, allowing access to a JADE agent. Invoking methods on instances of this class, it is possible to trigger state transition of the agent life cycle. This class must not be instantiated by applications. Instead, use the createNewAgent() method in class AgentContainer.

#### 5. CreateNewAgent

Creates a new JADE agent, running within this container with the following parameters;

nickname - A platform-unique nickname for the newly created agent. The agent will be given a FIPA compliant agent identifier using the nickname and the ID of the platform it is running on.

className - The fully qualified name of the class that implements the agent. We called this class Agents. args - An object array, containing initialization parameters to pass to the new agent.

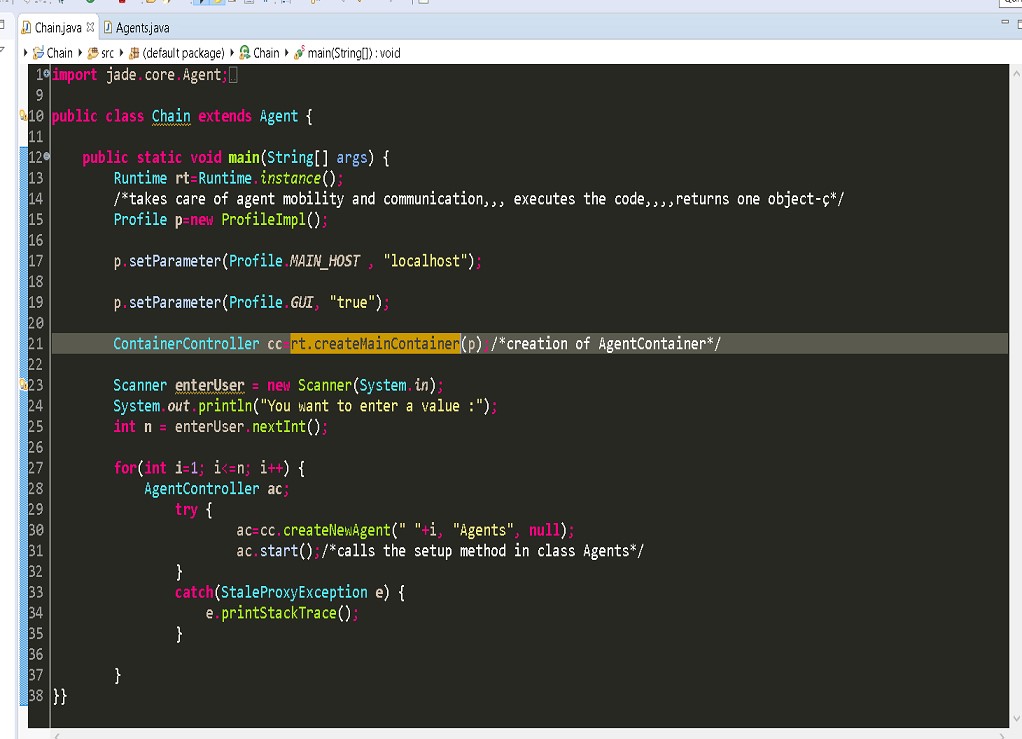


Fig: Class Chain

## Class Agents

This contains the setup method that is used for ordinary tasks of an agent. It is called by *ac.start();*

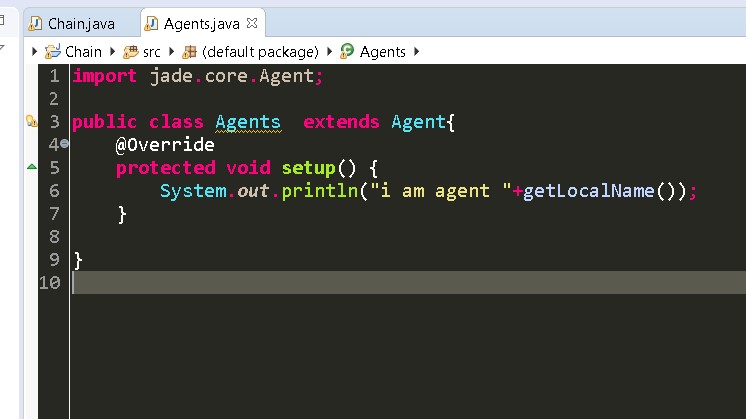


Fig: Class Agents

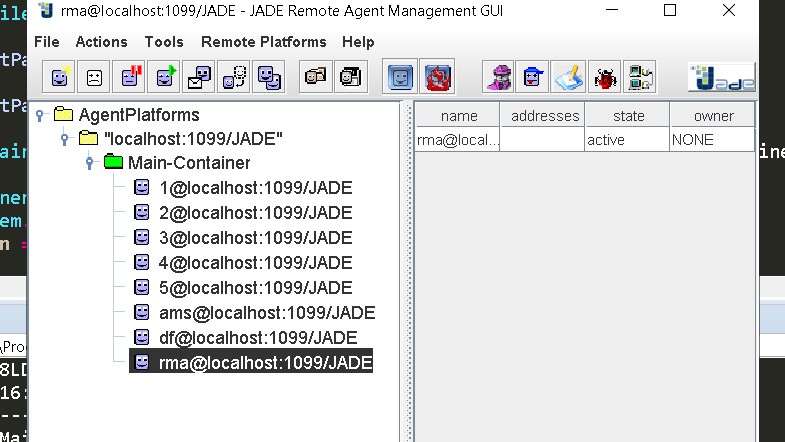


Fig: The Execution