

TALLER**AMBIENTE JADE**

Profesor: Jaime Alberto Guzmán Luna

Contenido del taller:

1. JADE
2. FIPA – ACL

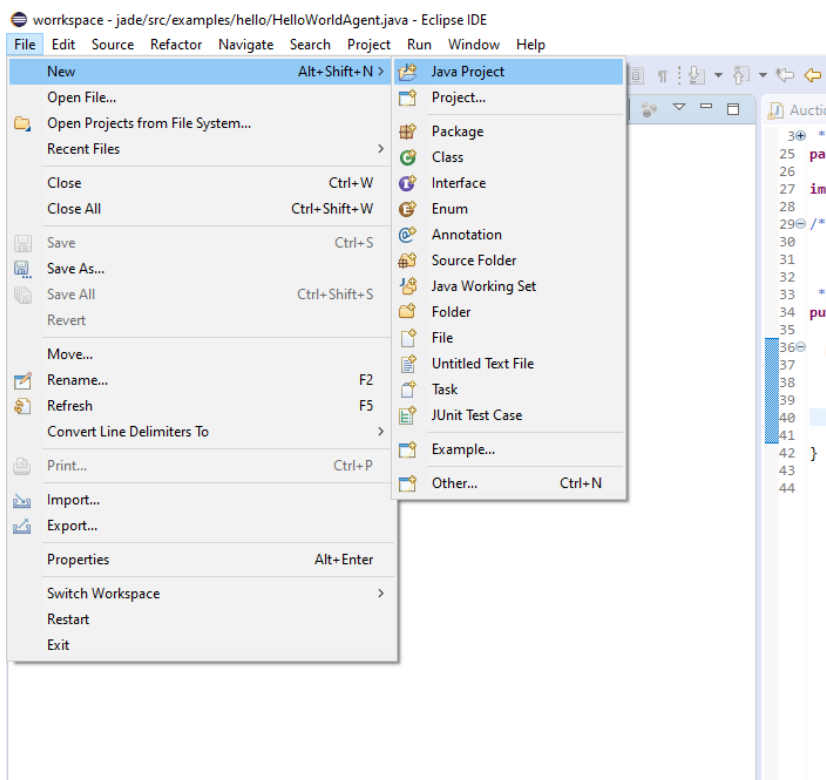
Uso de la herramienta Jade para construcciones de agentes inteligentes, y análisis de un sistema de Subasta, de manera práctica.

Para el desarrollo de esta práctica se adjunta un archivo la librería de Jade, este archivo es usado para hacer uso de Jade y ejecutar los agents, y un proyecto comprimido “Jade-Software-Agents” para analizar y ver puesto en práctica la comunicación y el protocolo de auction de los estándares de comunicación FIPA ACL.

Primero vamos a crear una agente HelloWorld para explicar el funcionamiento de la herramienta Jade y la manera de iniciar nuestros agentes.

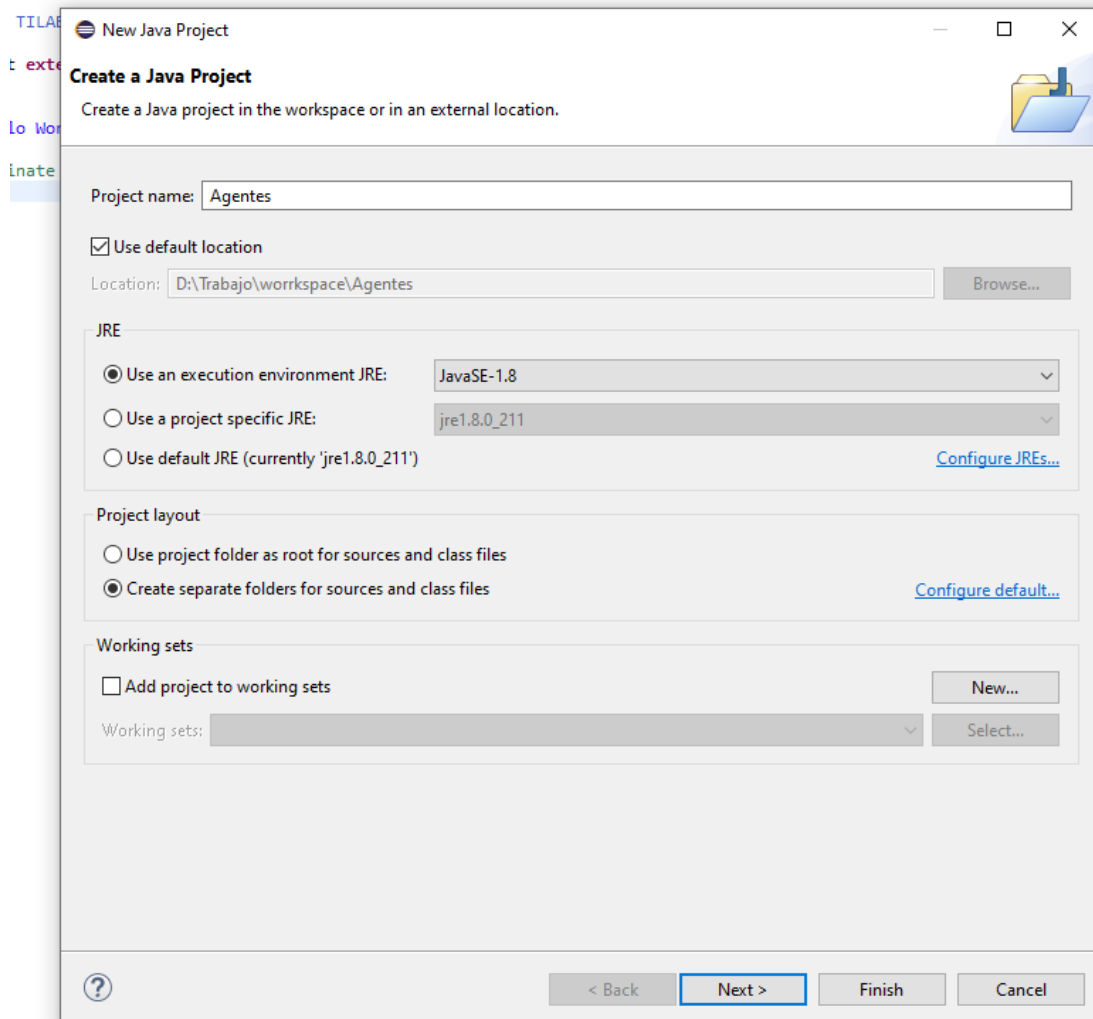
Parte 1

Primero vamos a crear un nuevo proyecto usando el IDE Eclipse:

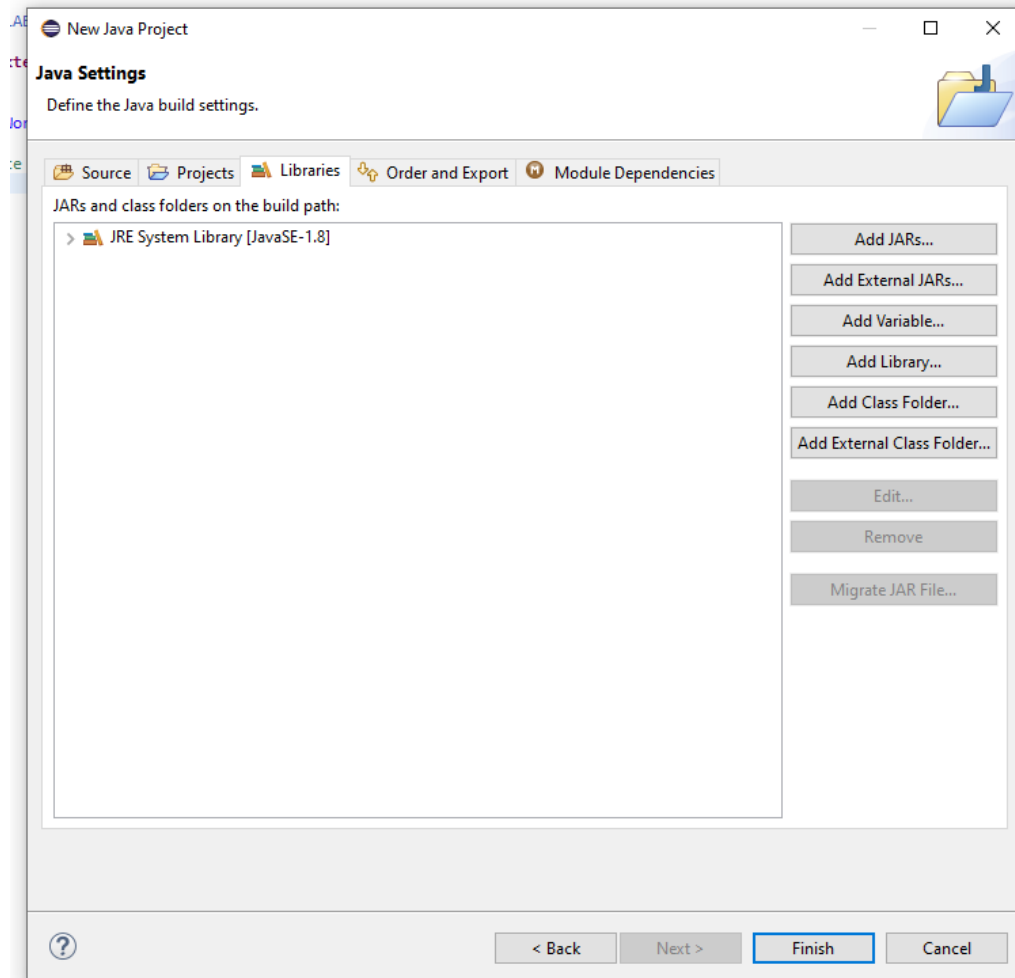


En el lado superior izquierdo seleccionaremos File -> New -> Java Project

Luego se nos desplegará la siguiente ventana y le daremos un nombre a nuestro proyecto en este caso “Agentes”, luego daremos clic en Next...



Al darle clic se nos mostrará una interfaz parecida a la siguiente:

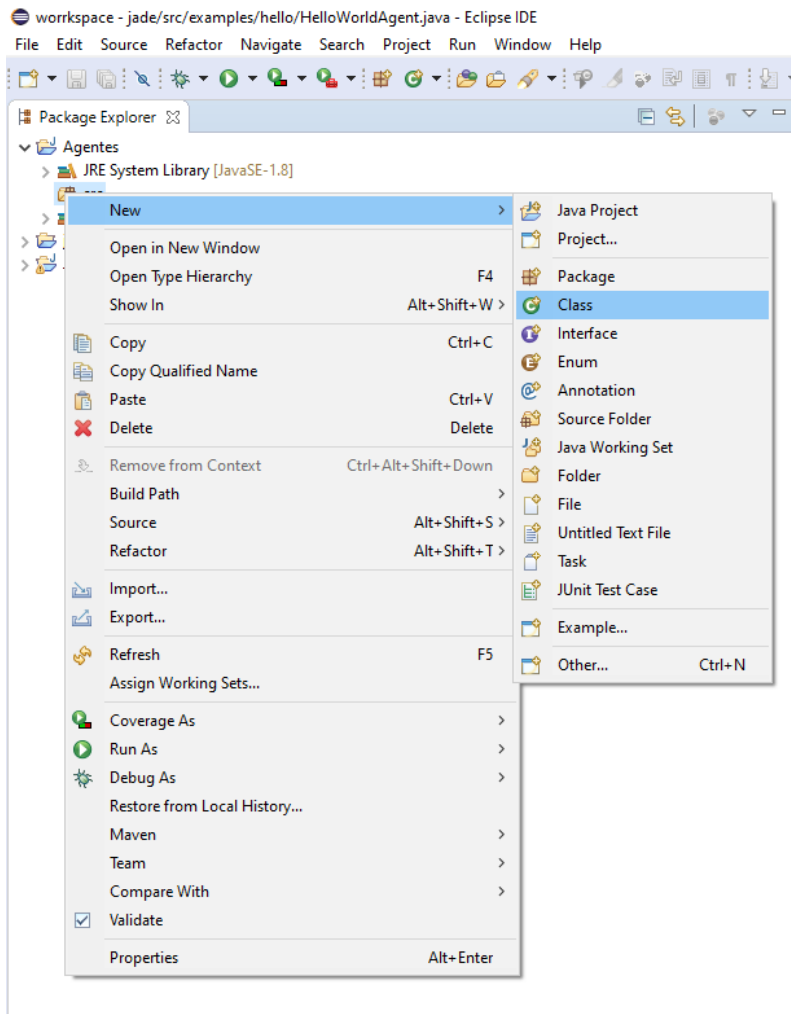


Seleccionaremos la pestaña Libraries y daremos clic al botón “Add External JARS...” en el lado derecho.

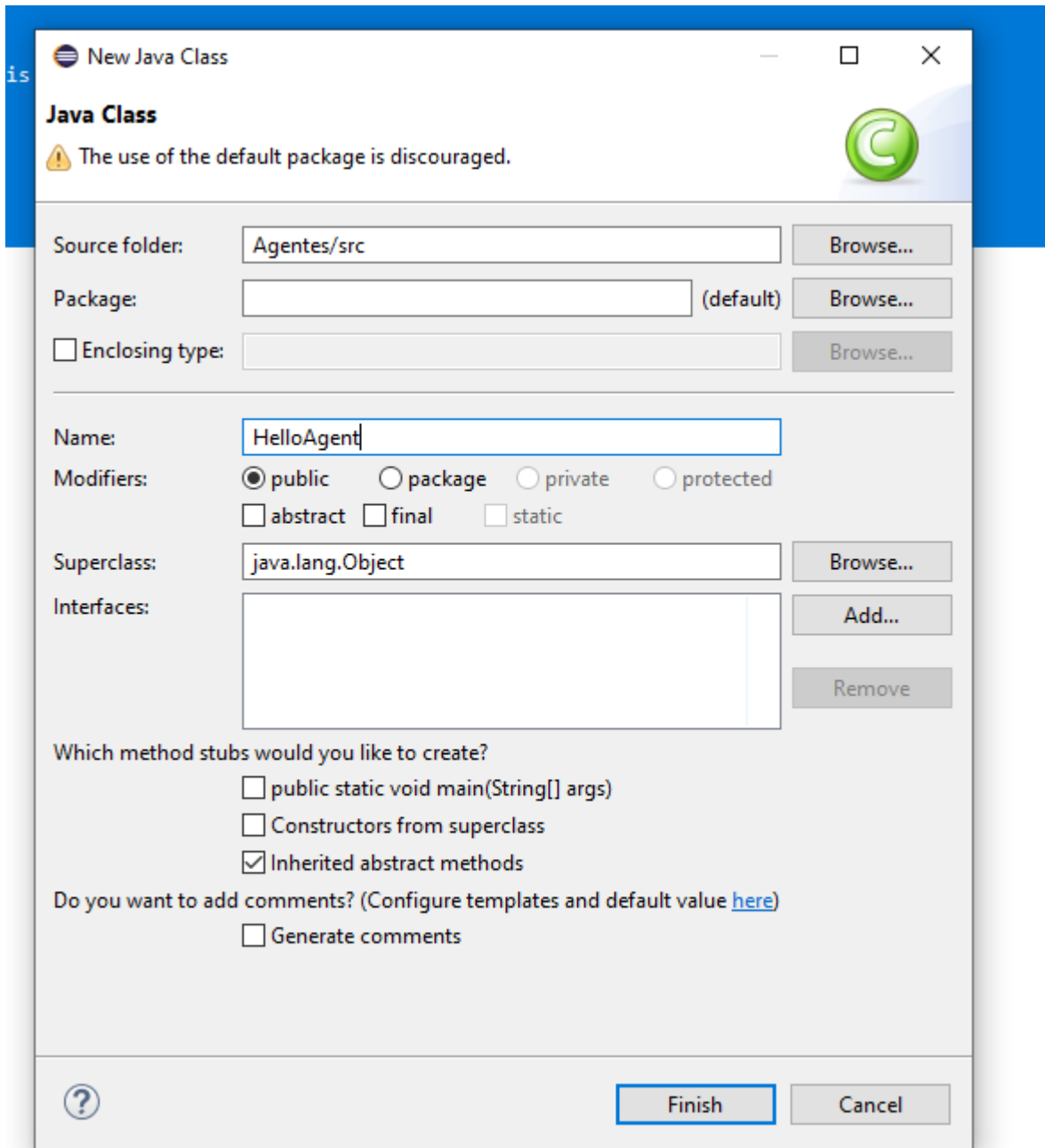
Allí buscaremos y seleccionaremos la librería de Jade que se nos adjunto y por ultimo le daremos en Finish.

Hasta el momento tenemos un proyecto en Java con la librería de Jade, ahora vamos a crear nuestro primer agente:

Para ello crearemos una nueva clase en java al darle clic derecho en la carpeta src -> New -> Class




Se nos presentara una interfaz parecida a la siguiente, donde le daremos un nombre a nuestra clase para el ejercicio lo llamaremos HelloAgent y daremos clic en Finish.



New Java Class

Java Class

 The use of the default package is discouraged.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

Para crear un agente este debería heredar de la clase Agent y definir un metodo setup, donde incluye todos los procesos de inicio del agente.

Nuestra clase lucirá de la siguiente manera:

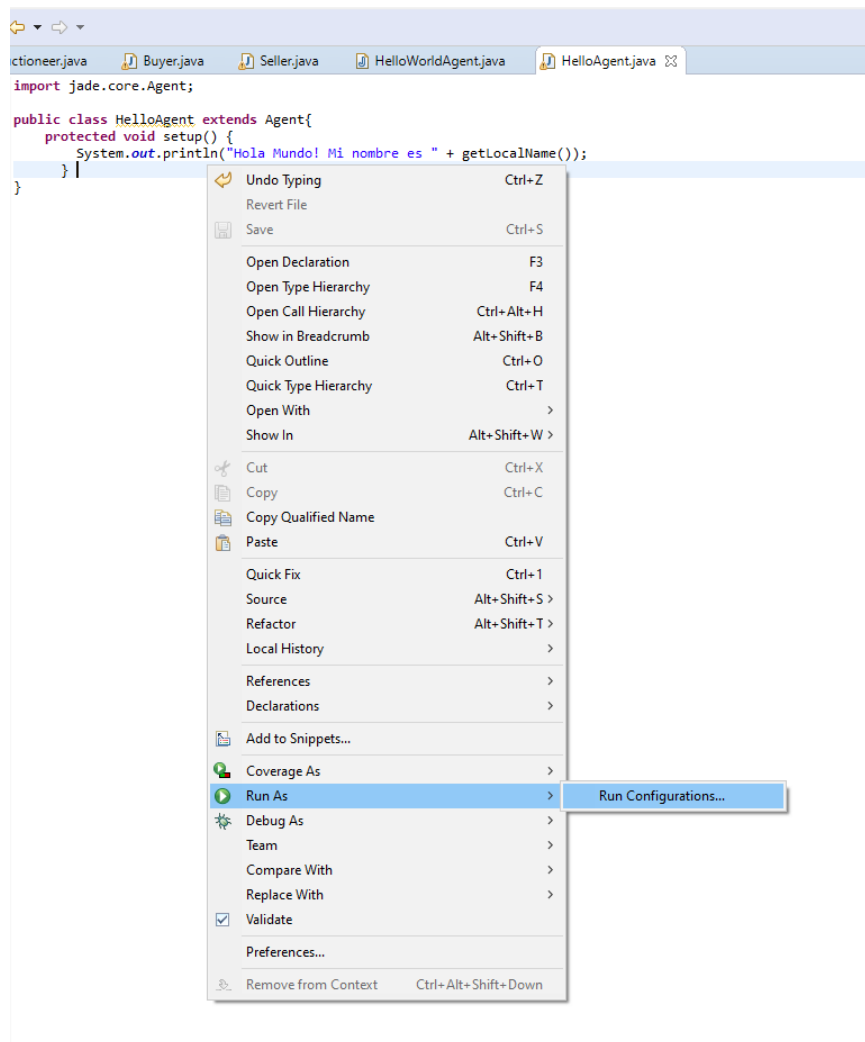
```
import jade.core.Agent;

public class HelloAgent extends Agent{
    protected void setup() {
        System.out.println("Hola Mundo! Mi nombre es " + getLocalName());
    }
}
```

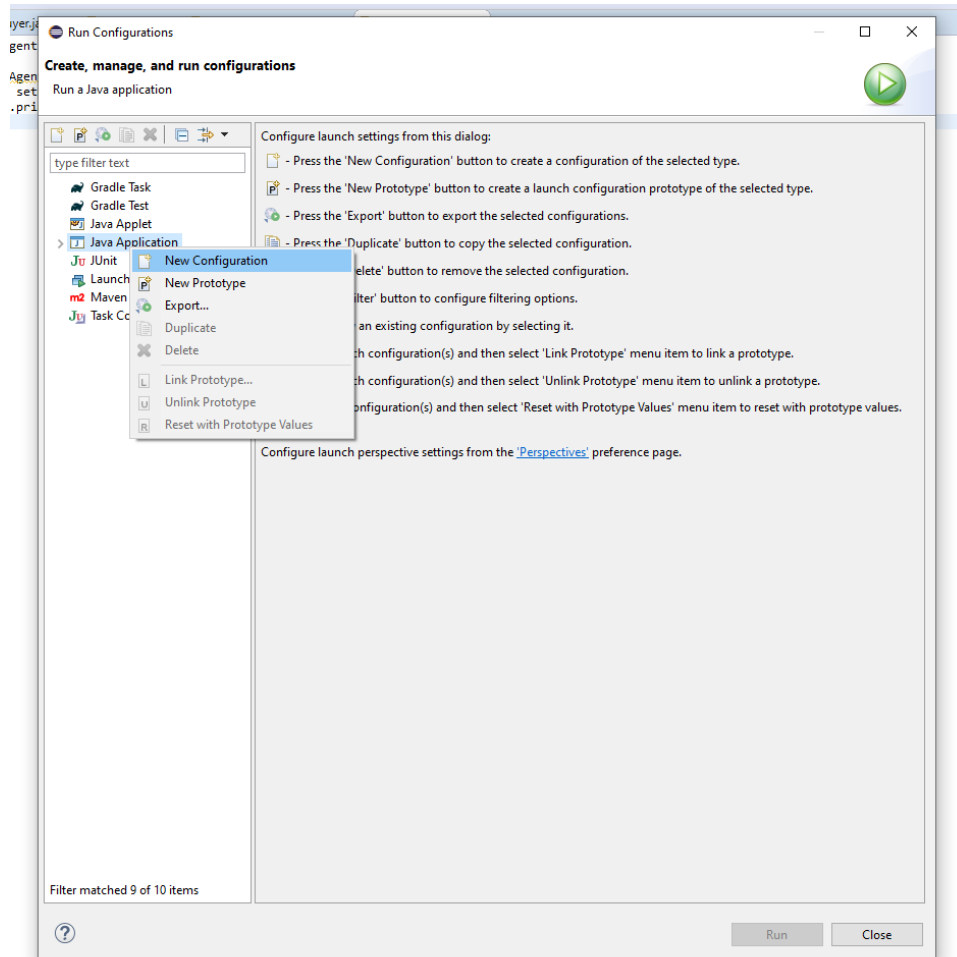
```
}
```

El funcionamiento de este agente es de meramente imprimir un mensaje del nombre que se le dio como agente, con el fin de ilustrar la manera de funcionar Jade.

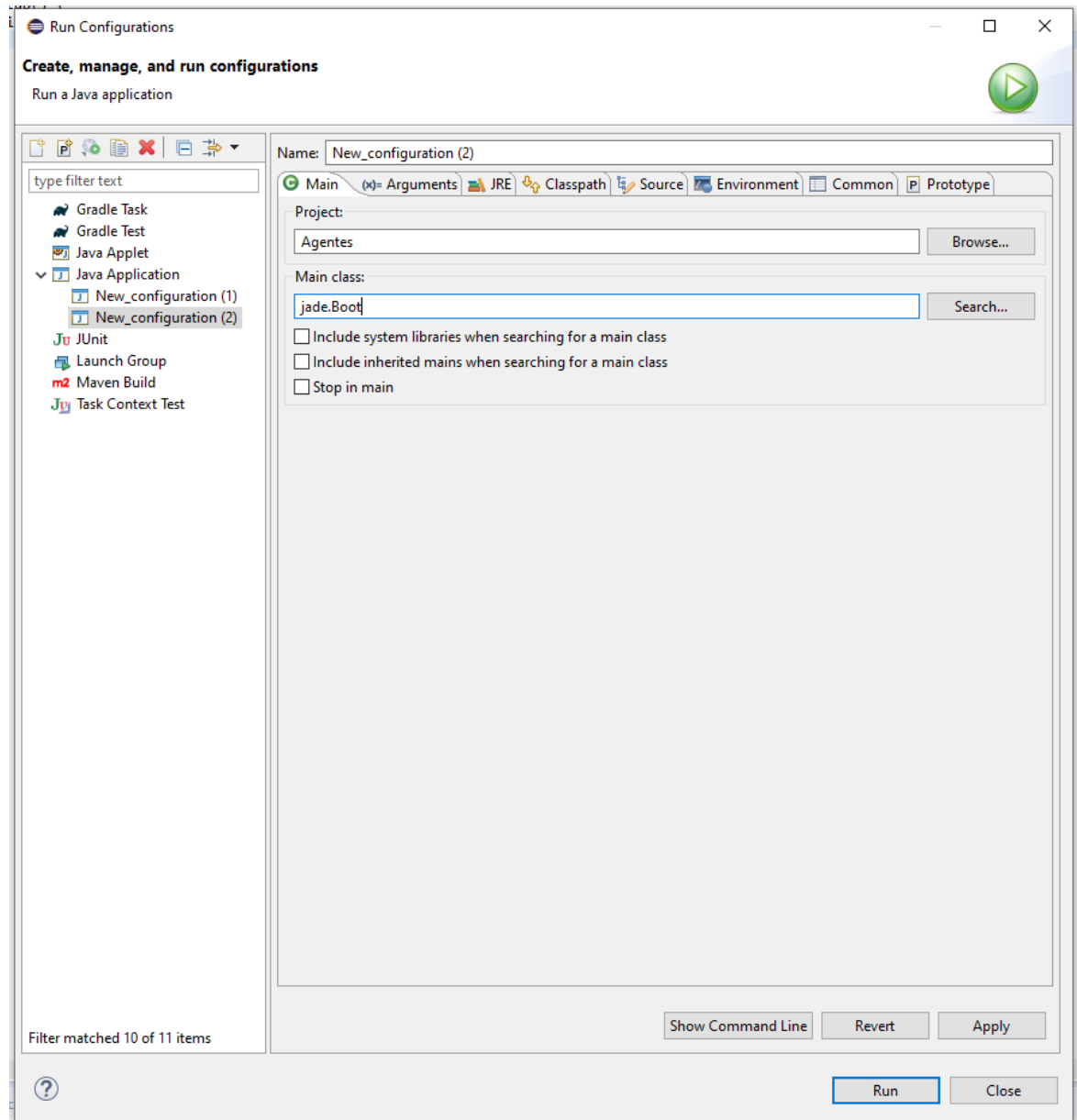
Para poder correr nuestro agente daremos clic derecho en el código de la clase HelloAgent y daremos clic en Run As -> Run Configurations...



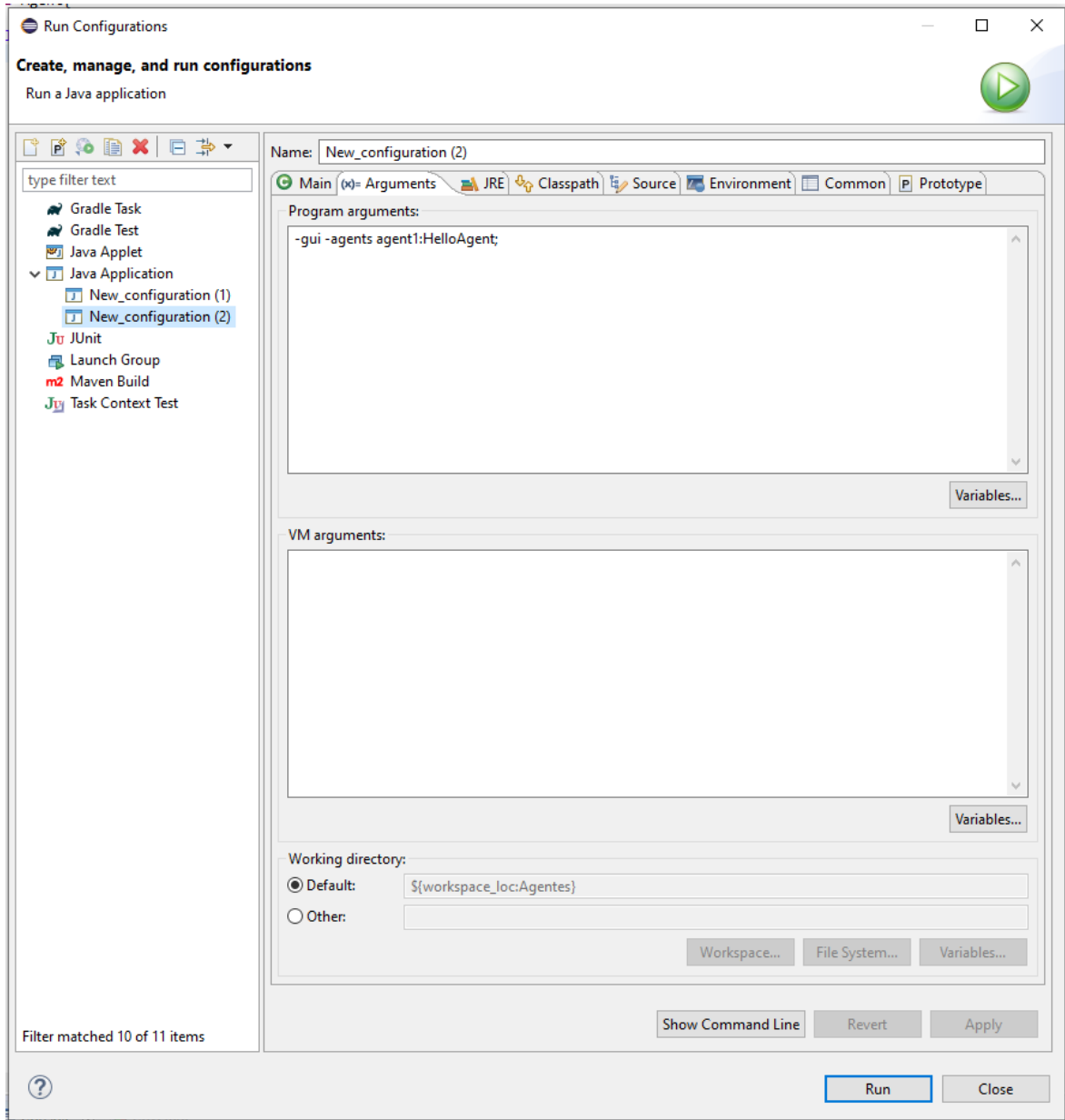
Se nos despliega la siguiente interfaz donde añadiremos una nueva configuración para correr nuestro programa, daremos clic derecho en Java Application -> New Configuration



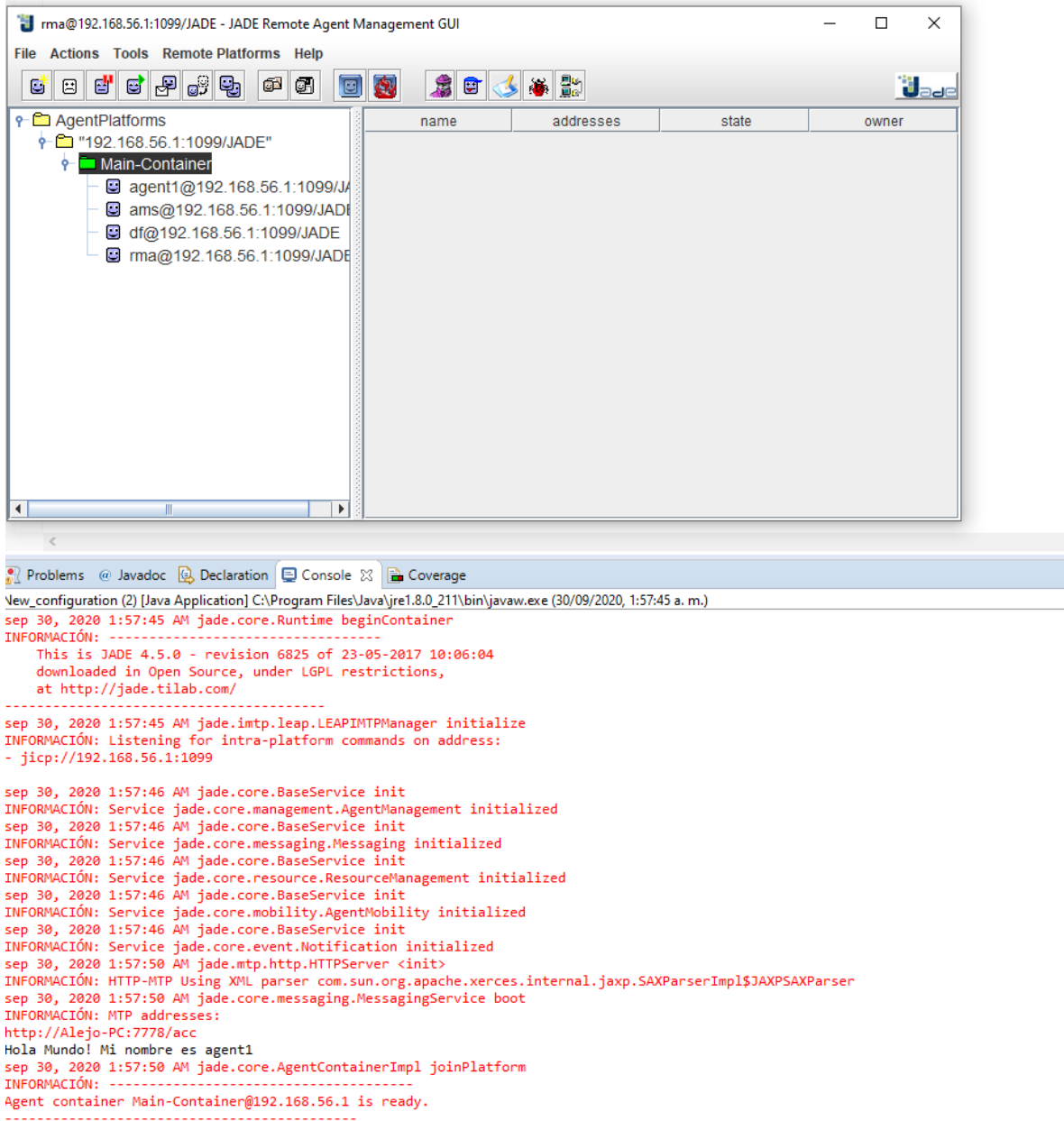
En el campo Main class escribiremos jade.Boot y daremos clic en Apply



Luego en la pestaña Arguments, en el campo Program arguments agregaremos lo siguiente `-gui -agents agent1:HelloAgent;`, esto especifica que deseamos que se abra la interfaz grafica de Jade usando la palabra `"-gui"` y que ejecutaremos unos agentes `"-agents agentes..."`, la creación de los agentes sigue el siguiente formato **nombre_agente:ubicacion_de_la_clase**, esta clase es la que contiene el agente, si deseamos crear mas de un agente los podemos hacer separando por `“;”`.



Luego de configurado para poder correrlo le daremos Run y se nos desplegara lo siguiente:



The screenshot shows the JADE Remote Agent Management GUI and its console output. The GUI window, titled "rma@192.168.56.1:1099/JADE - JADE Remote Agent Management GUI", displays a tree view of AgentPlatforms. Under "192.168.56.1:1099/JADE", there is a "Main-Container" which contains several agents: "agent1@192.168.56.1:1099/JADE", "ams@192.168.56.1:1099/JADE", "df@192.168.56.1:1099/JADE", and "rma@192.168.56.1:1099/JADE". The main table in the GUI has columns for "name", "addresses", "state", and "owner".

The console output shows the following messages:

```

New_configuration (2) [Java Application] C:\Program Files\Java\jre1.8.0_211\bin\javaw.exe (30/09/2020, 1:57:45 a. m.)
sep 30, 2020 1:57:45 AM jade.core.Runtime beginContainer
INFORMACIÓN: -----
This is JADE 4.5.0 - revision 6825 of 23-05-2017 10:06:04
downloaded in Open Source, under LGPL restrictions,
at http://jade.tilab.com/
-----
sep 30, 2020 1:57:45 AM jade.imtp.leap.LEAPIMTPManager initialize
INFORMACIÓN: Listening for intra-platform commands on address:
- jicp://192.168.56.1:1099

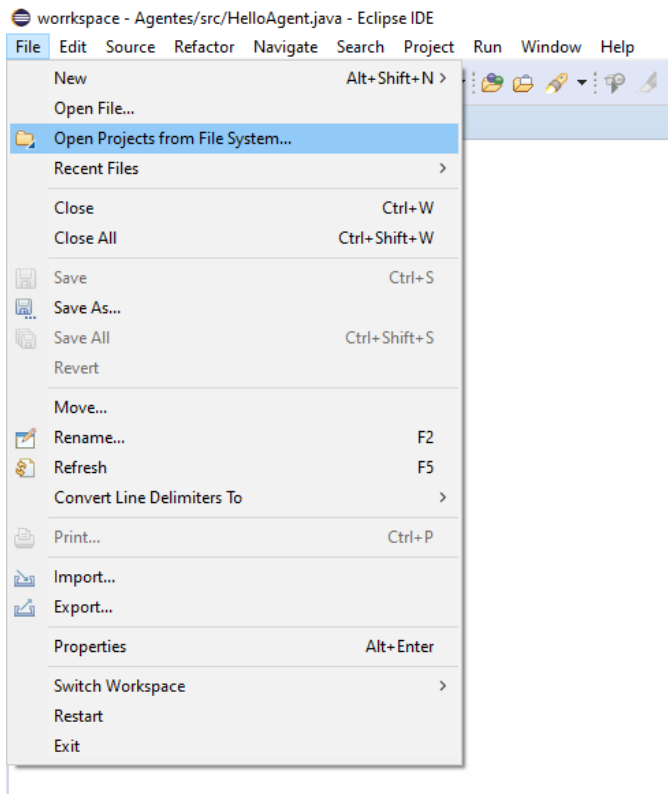
sep 30, 2020 1:57:46 AM jade.core.BaseService init
INFORMACIÓN: Service jade.core.management.AgentManagement initialized
sep 30, 2020 1:57:46 AM jade.core.BaseService init
INFORMACIÓN: Service jade.core.messaging.Messaging initialized
sep 30, 2020 1:57:46 AM jade.core.BaseService init
INFORMACIÓN: Service jade.core.resource.ResourceManagement initialized
sep 30, 2020 1:57:46 AM jade.core.BaseService init
INFORMACIÓN: Service jade.core.mobility.AgentMobility initialized
sep 30, 2020 1:57:46 AM jade.core.BaseService init
INFORMACIÓN: Service jade.core.event.Notification initialized
sep 30, 2020 1:57:50 AM jade.mtp.http.HTTPServer <init>
INFORMACIÓN: HTTP-MTP Using XML parser com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl$JAXPSAXParser
sep 30, 2020 1:57:50 AM jade.core.messaging.MessagingService boot
INFORMACIÓN: MTP addresses:
http://Alejo-PC:7778/acc
Hola Mundo! Mi nombre es agent1
sep 30, 2020 1:57:50 AM jade.core.AgentContainerImpl joinPlatform
INFORMACIÓN: -----
Agent container Main-Container@192.168.56.1 is ready.
-----

```

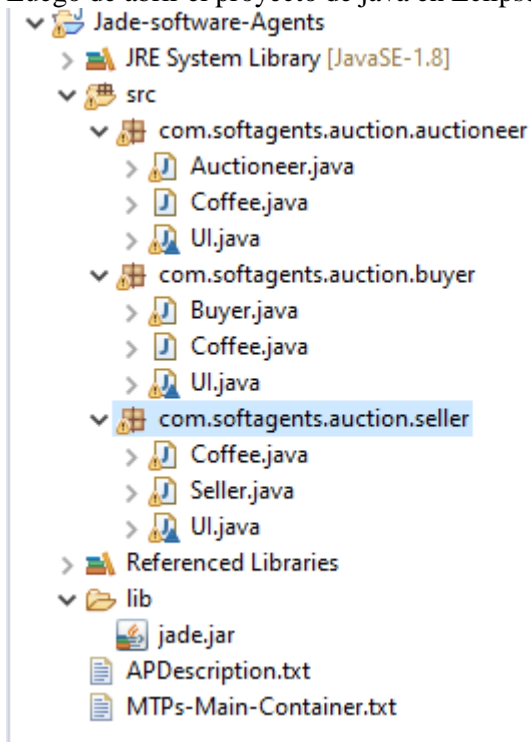
Por consola nos muestra lo que hace nuestro agente imprimir el nombre del agente, además de desplegarse una gui, donde veremos los agentes que están actualmente interactuando y han sido creados.

Parte 2

Análisis de un ejercicio de subasta “English auction agents in jade”, para el desarrollo de este ejercicio descomprima el archivo de “Jade-Software-Agents”, y abra el proyecto de java en eclipse, esto lo puede hacer desde la esquina superior izquierda File -> Open Projects... y seleccionar la carpeta donde esta ubicado el archivo que descomprimio.



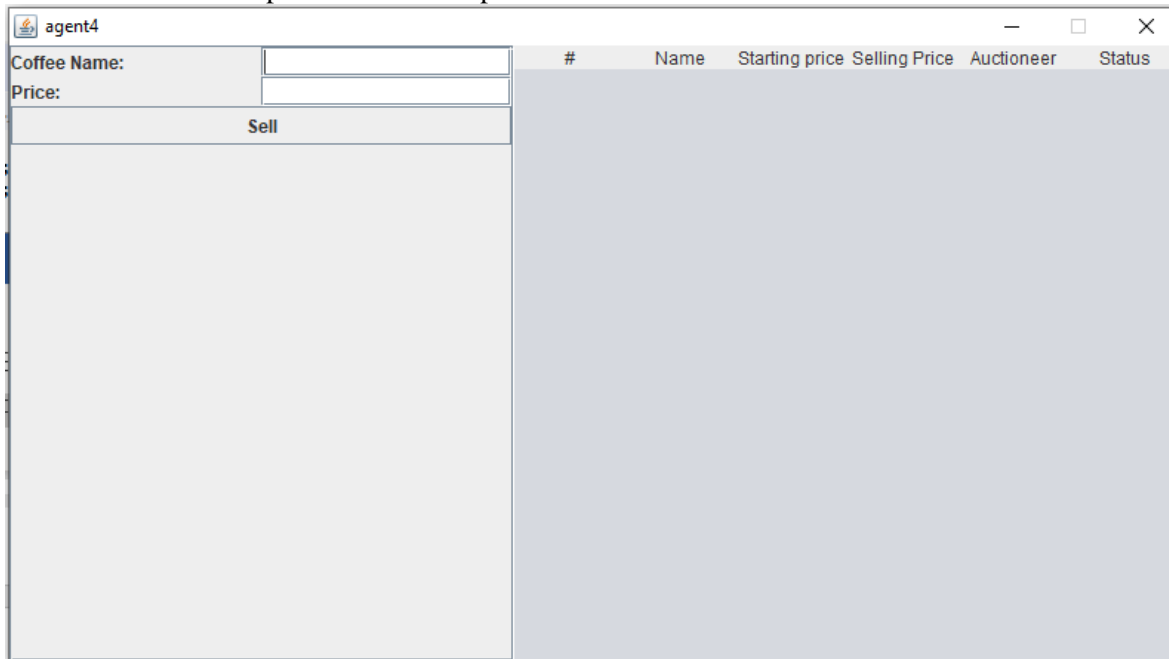
Luego de abrir el proyecto de java en Eclipse allí podrá evidenciar la siguiente estructura:



El funcionamiento de este sistema es de la siguiente manera:

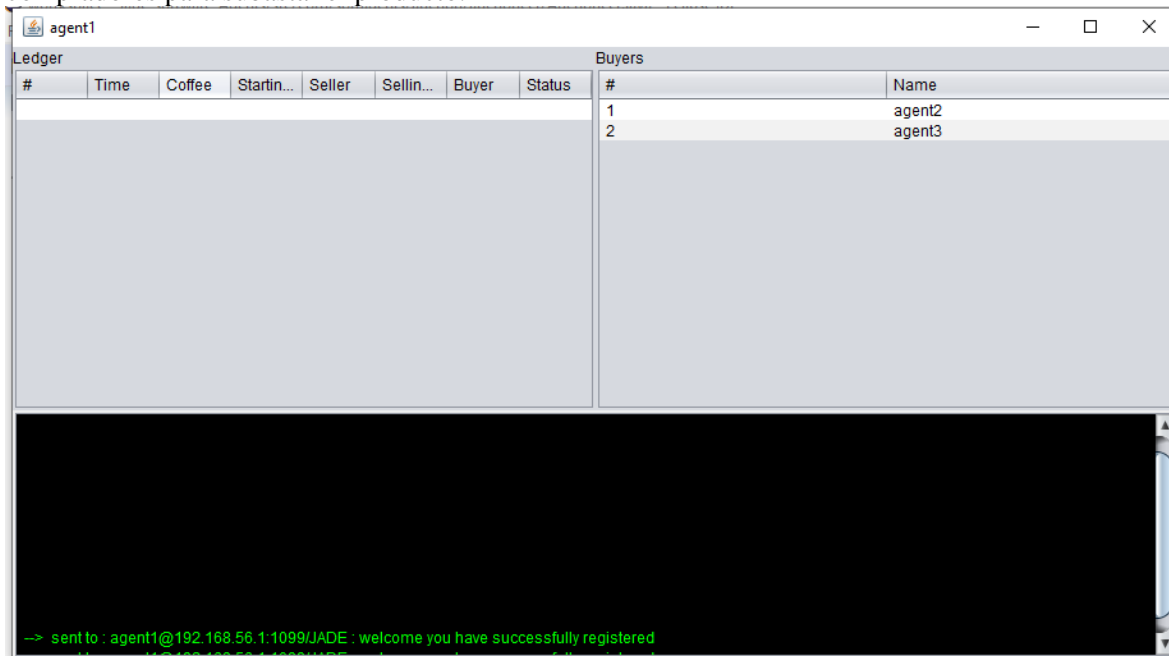
Se crean como mínimo un agente Auctioneer, Seller y Buyer, encargados de la subasta, ofrecer productos y comprar productos respectivamente.

El agente Seller es el encargado de proveer los productos que serán ofertados en la subasta, para esto se brinda una interfaz para crear nuevos productos.



The screenshot shows a window titled 'agent4'. On the left, there is a 'Sell' form with two input fields: 'Coffee Name:' and 'Price:'. To the right of these fields is a large, empty table with the following headers: '#', 'Name', 'Starting price', 'Selling Price', 'Auctioneer', and 'Status'.

El agente Auctioneer será el encargado de gestionar la subasta, se comunicara con los diferentes compradores para subastar el producto.

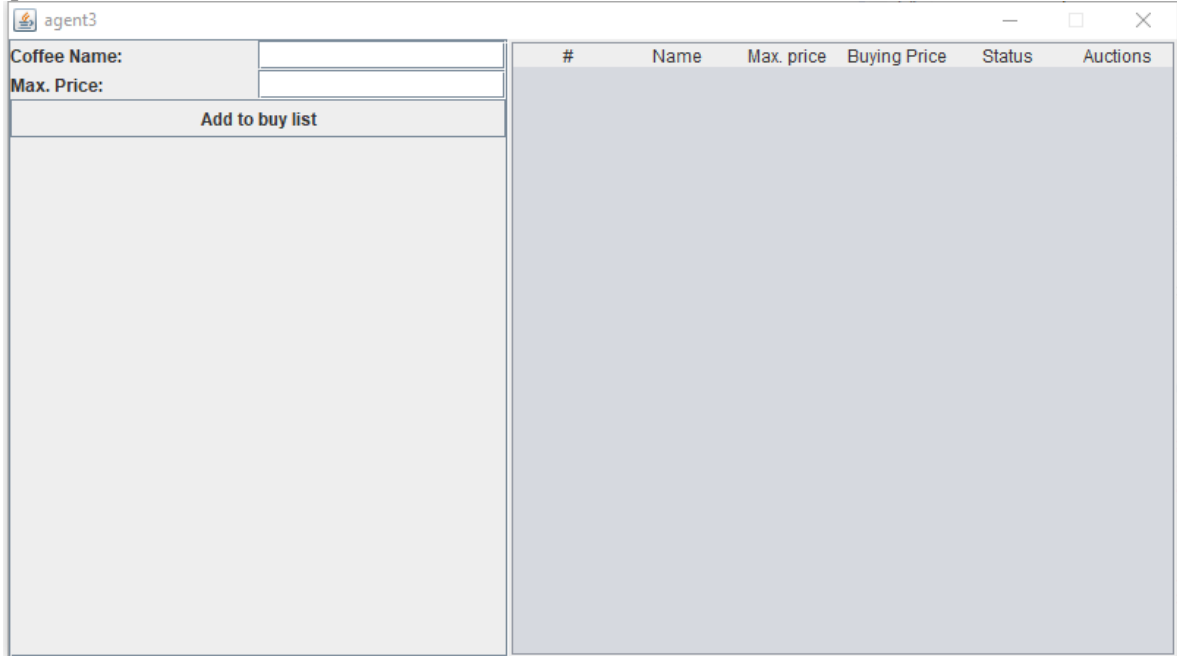


The screenshot shows a window titled 'agent1'. It contains two main sections: a 'Ledger' table on the left and a 'Buyers' table on the right. The 'Ledger' table has headers: '#', 'Time', 'Coffee', 'Startin...', 'Seller', 'Sellin...', 'Buyer', and 'Status'. The 'Buyers' table has headers: '#', 'Name', and 'Status'. Below these tables is a black console area with green text output.

#	Name
1	agent2
2	agent3

```
--> sent to : agent1@192.168.56.1:1099/JADE : welcome you have successfully registered
```

Por último el agente Buyer, es el encargado de comprar por un precio máximo, el objeto subastado que desea.



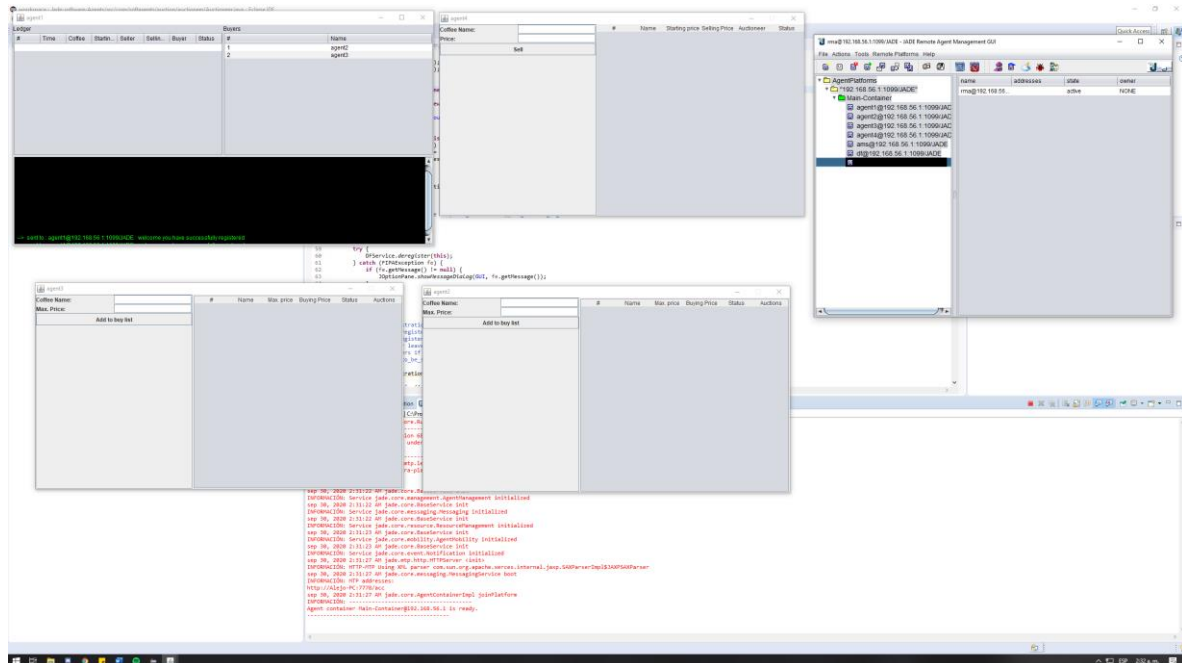
The screenshot shows a window titled 'agent3'. On the left, there is a form with two input fields: 'Coffee Name:' and 'Max. Price:'. Below these fields is a button labeled 'Add to buy list'. On the right, there is a table with the following headers: '#', 'Name', 'Max. price', 'Buying Price', 'Status', and 'Auctions'. The table body is currently empty.

Para correr los agentes cree una nueva configuración como lo vimos con el ejercicio del agente HelloAgent. Para este caso en el campo Main class use el mismo valor "jade.Boot" y en los argumentos escriba lo siguiente:

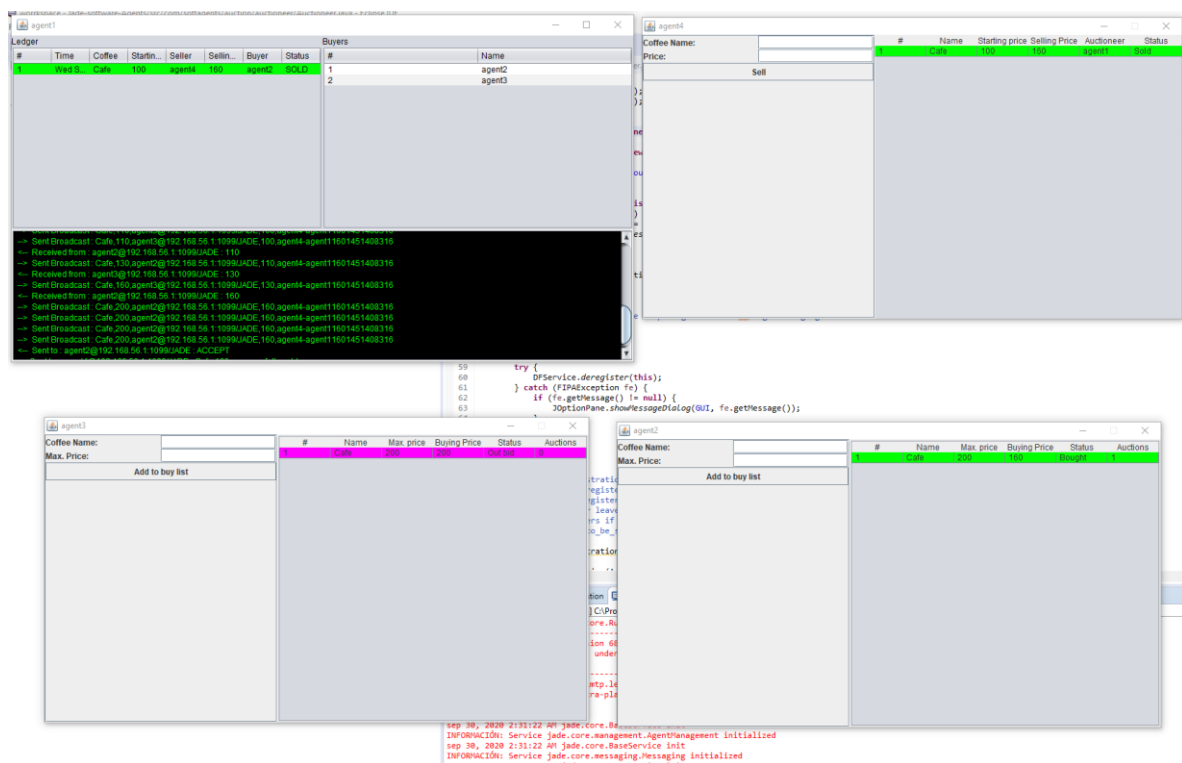
```
-gui -agents
agent1:com.softagents.auction.auctioneer.Auctioneer;agent2:com.softagents.auction.buyer.Buyer
;agent3:com.softagents.auction.buyer.Buyer;agent4:com.softagents.auction.seller.Seller
```

Estos argumentos se encargan de crear un agente Auctioneer (encargado de gestionar la subasta), un agent Seller (encargado de proveer los productos que serán subastados), y dos agentes Buyers (encargados de dar valores a los productos por los cuales van a ofertar)

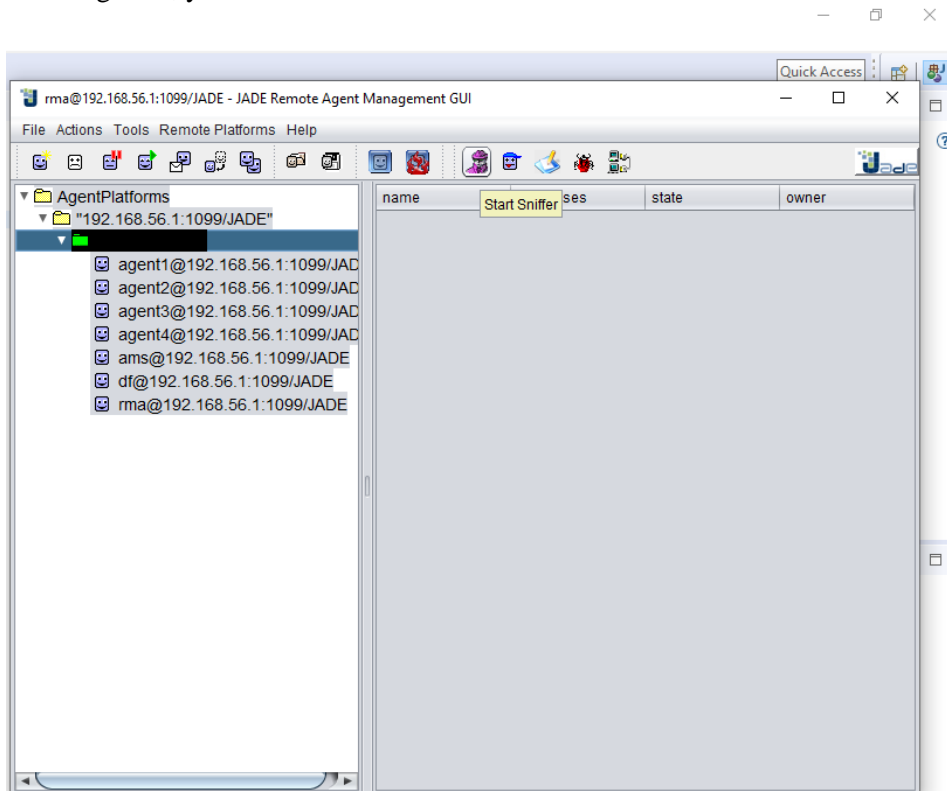
Luego de correr el programa se le desplegarán las siguientes ventanas, en este caso hay dos agentes Buyers.



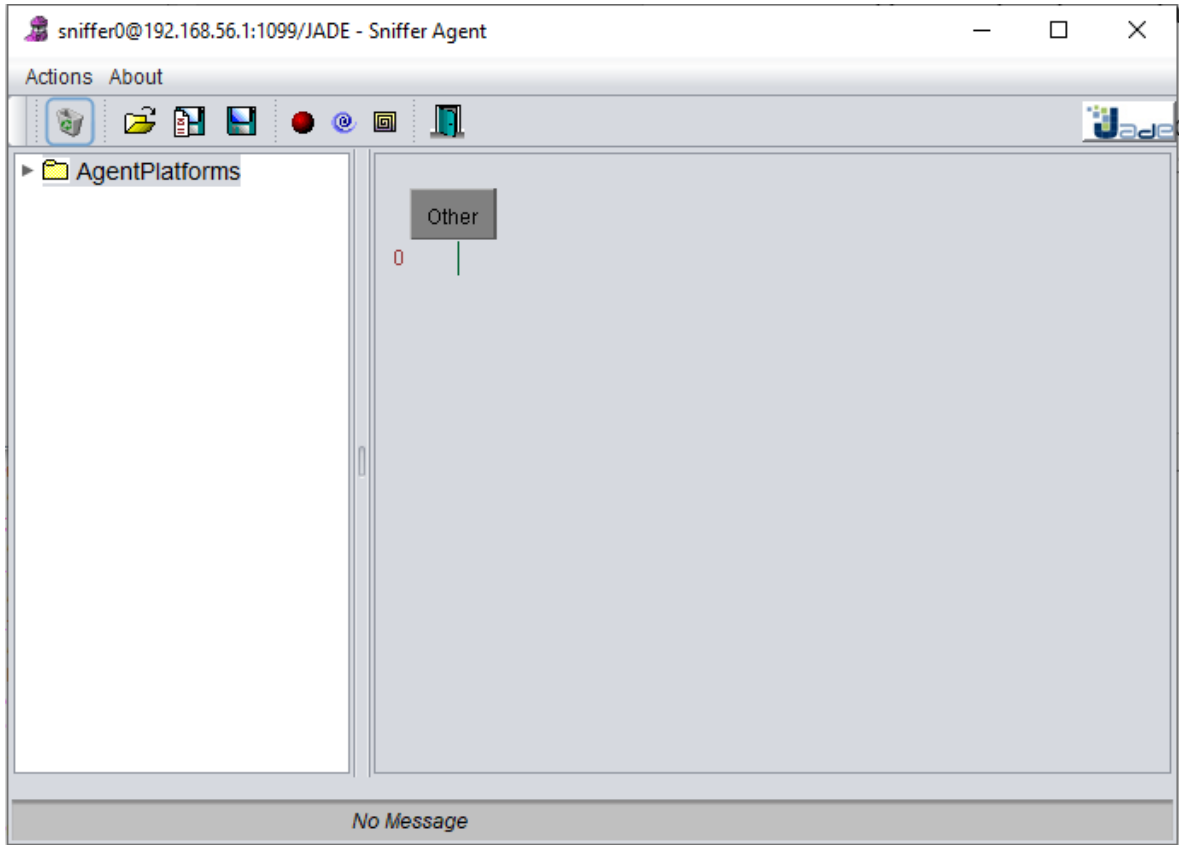
Luego de crear un nuevo objeto “Cafe” con un precio de 200 y generar nuevas intenciones de compra en los agentes Buyers, para el agent3 el precio máximo que pagaría sería de 100 y para el agent2, el precio máximo que pagaría sería 200, luego de correr la subasta se evidencia el siguiente resultado, el agent2 es el ganador de la subasta, el agent3 no pudo ofertar mas por el objeto, y finalmente fue vendido el objeto al agent2, y al agent4, recibió la noticia de que su producto fue vendido, el resultado se evidencia de manera gráfica en la siguiente imagen:



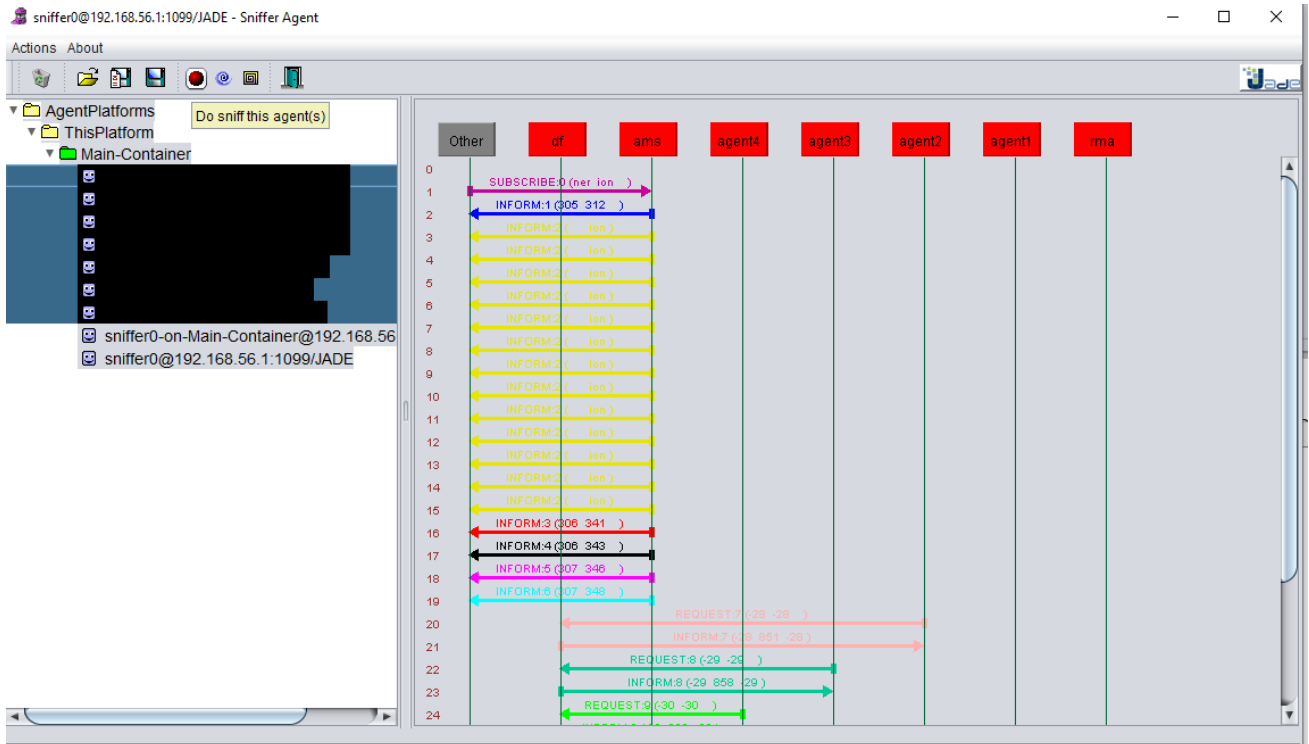
Una herramienta muy útil de Jade es el sniffer, que nos permite ver las comunicaciones que han sido efectuadas entre los agentes, para esto desde la gui del programa Jade, seleccionaremos el “container” de los agentes, y daremos clic en el botón de Start Sniffer.



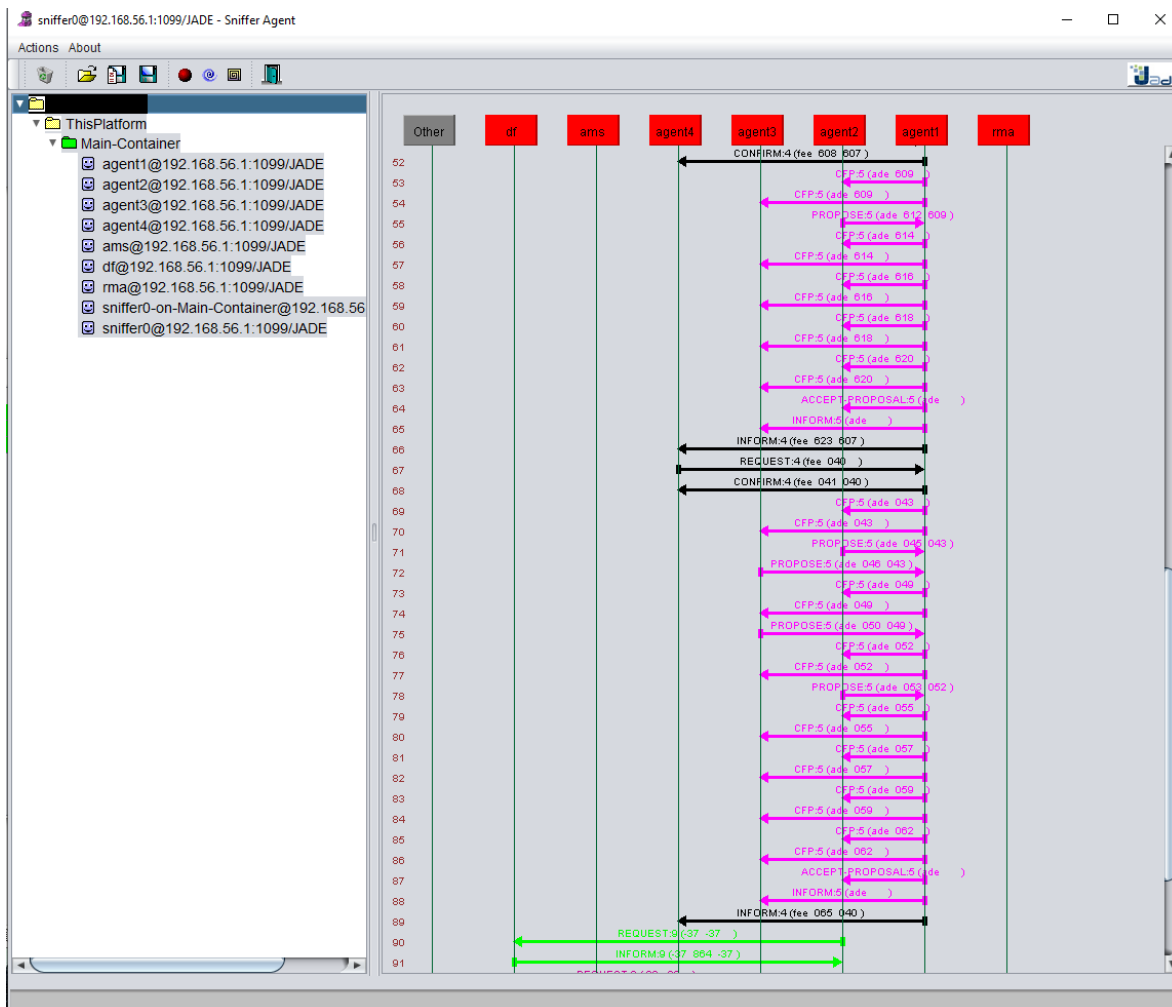
Se nos desplegara la siguiente interfaz:



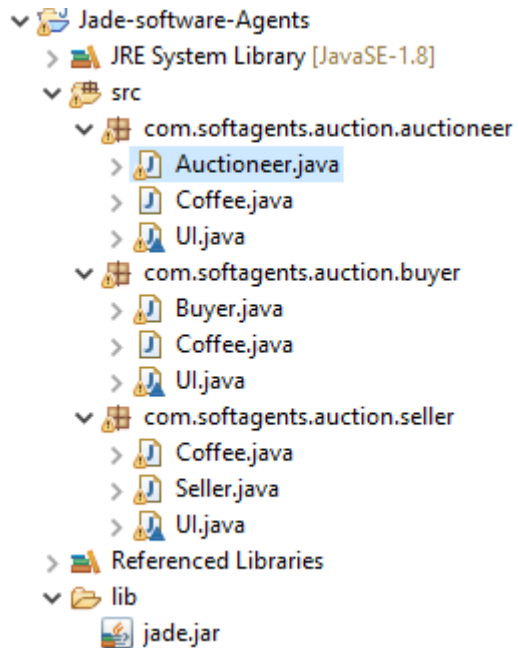
Luego al seleccionar todos nuestros agentes (excluya los agentes creados por el sniffer), y seleccionando el botón Do sniff (Boton Rojo en la parte superior), se nos empezara a visualizar las comunicaciones que están ocurriendo entre los diferentes agentes.



Si analizamos el resultado de una subasta entre los agentes podemos notar la similitud con el modelo de comunicación para una auction planteada en los estandares FIPA ACL:



El código y modo como se comunican los agentes lo puede evidenciar en cada una de las clases Auctioneer, Seller, Buyer, definidas en el proyecto, allí se evidencian las acciones y la comunicación que esta ocurriendo entre los agentes.



Material de apoyo:

- Pagina principal de Jade, y acceso a tutoriales y ejemplos: [<https://jade.tilab.com/>]
- Proyecto análisis English auction agents in jade: [<https://github.com/ETdvlpr/Jade-software-Agents>]