

Grado en Ingeniería Informática

Inteligencia Artificial

Curso 2019/2020



Universidad de Jaén

Guión 0

Java

Sistemas Multiagentes



## 1. Introducción a Java

Esta práctica inicial tiene como objetivo que el alumno se familiarice con un entorno de programación basado en el lenguaje Java, así como en el uso de dicho lenguaje.

En la plataforma ILIAS se encuentra disponible un manual básico y resumido de programación orientada a objetos en Java. No obstante, para obtener más información sobre métodos concretos siempre se puede consultar el tutorial Java de Sun (<http://docs.oracle.com/javase/>).

### **Diseñar un programa en Java para resolver el siguiente supuesto:**

1. Definir una clase Alumno, que contenga como mínimo las siguientes características: Nombre, DNI y Correo-E. Escribir dos constructores, uno genérico para crear un alumno, y otro que inicialice todos sus campos a unos valores que se le pasen como parámetros.
2. Definir un método que cree un nuevo Alumno, a partir de una serie de valores que se introduzcan por teclado. Definir otro método que muestre los datos de ese alumno por pantalla.
3. Definir una clase Alumno\_IA, subclase de la anterior, que contenga otros dos campos más: Grupo de Prácticas y Nota de Prácticas.
4. Definir un método que lea desde teclado las 4 notas de prácticas de un alumno de IA, y calcule su nota de prácticas como la media de esos 4 valores. Mostrar el resultado por pantalla.
5. Se dispone de un archivo “datos.txt” (en ILIAS) con un conjunto de alumnos (uno por línea). Definir los métodos necesarios para leer el contenido de dicho archivo, y volcar en un nuevo archivo “pares.txt” una lista con los alumnos cuyo número de DNI sea par.

### **Entrega**

Esta primera práctica no tendrá sesión de defensa.

Cada trabajo práctico debe ir comprimido en un archivo (.zip, .rar, etc.), dentro del cual se incluirá documentación y código fuente (archivos .java).

Además del código, se entregará una memoria en formato PDF describiendo brevemente, en lenguaje natural, cómo se ha resuelto cada uno de los problemas implementados. La memoria no debe incluir código fuente. Extensión recomendada: 3 páginas.

## 2. Introducción a los agentes artificiales

Dentro del campo de la Inteligencia Artificial se considera un agente inteligente a una entidad física o virtual capaz de percibir su entorno, procesar tales percepciones y responder o actuar de manera racional, es decir, de manera correcta creando y maximizando un resultado.

Un agente inteligente es capaz de percibir su medio ambiente con la ayuda de sensores y actuar en ese medio utilizando actuadores.

*¿Qué es un actuador?*

Elemento que reaccionan a un estímulo realizando una acción.

*¿Qué es un sensor?*

Aparato con una propiedad sensible a una magnitud del medio y que al variarla también varía con intensidad la propiedad, es decir, que manifiesta la presencia de dicha magnitud.

Los agentes inteligentes se describen esquemáticamente como un sistema funcional abstracto. Por esta razón, los agentes inteligentes son a veces llamados Agentes Inteligentes Abstractos (AIA) para distinguirlos de sus implementaciones del mundo real como sistemas informáticos, los sistemas biológicos, o de organizaciones.

Las definiciones clásicas hacen énfasis en su autonomía por lo que prefieren el término agente inteligente autónomo. Otras, sin embargo, lo consideran conducta dirigida a objetivos como la esencia de lo inteligente y prefieren un término tomado de la economía "Agente Racional".

Dentro de nuestro ámbito (*Ciencias de la Computación e Inteligencia Artificial*) el término agente inteligente se refiere a:

*Agente computacional autónomo, con iniciativa y capacidad de explorar y modificar su entorno, y que tiene la posibilidad de comunicarse con otros agentes.*

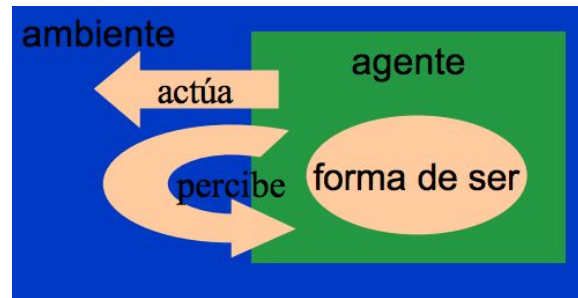
Un símil típico que se hace de los agentes es el de los procesos de un sistema operativo. Dentro de un sistema operativo se invocan procesos y estos procesos se comunican con otros procesos para resolver un problema.

Las clasificaciones clásicas de agentes inteligentes los dividen en distintas categorías atendiendo a su arquitectura:

- Reactivos como los agentes reactivos, o con representación en el mundo.

- Deliberativa como los agentes basados en metas o utilidad.

Un ejemplo abstracto de un agente se puede observar en la siguiente figura:



A continuación, se muestra cómo funcionan los agentes en entornos reales resolviendo problemas:

AGENTE	PERCEPCIONES	ACCIONES	METAS	AMBIENTE
Diagnosticador médico	Síntomas, evidencias, respuestas	Preguntas, análisis, tratamientos	Salud, mínima intrusión	Paciente, hospital
Robot clasificador de piezas	Mapas de píxeles	Recoger piezas y clasificarlas	Poner pieza en caja	Cinta transportadora con piezas
Asistente de e-Mail	Encabezados y textos de mensajes	Clasificar, borrar, responder	Reproducir comportam. del usuario	Cientes y servidores de correo, usuario

## 2.1. Sistemas MultiAgentes

*Un Sistema Multiagente (SMA) es un sistema compuesto por múltiples agentes inteligentes (potencialmente independientes) que interactúan entre ellos para resolver un problema.*

Los SMA se pueden utilizar para resolver problemas difíciles, o sobretodo, problemas que un sistema individual o monolítico no puede resolver. Las principales características que se les asocian a los agentes de un SMA son:

- Capacidad de tomar la iniciativa.
- Capacidad de compartir conocimiento.
- Capacidad de cooperar y negociar.
- Capacidad de comprometerse con metas comunes.

Los ámbitos en los que la investigación de los SMA puede ofrecer un enfoque adecuado incluyen:

- Comercio electrónico.
- Respuesta a desastres.
- Modelado de estructuras sociales.
- Etc.

Más adelante, nosotros trabajaremos con SMA en la práctica 2 de la asignatura, donde veremos cómo los agentes del sistema cooperan, comparten conocimiento, e intentan sobrevivir en un mundo virtual.

Las características principales de los agentes dentro de un SMA serían:

- Autonomía: Deben ser parcialmente autónomos.
- Visión local: No tienen visión del sistema, solo de ellos mismos.
- Descentralización: Nadie controla el sistema.

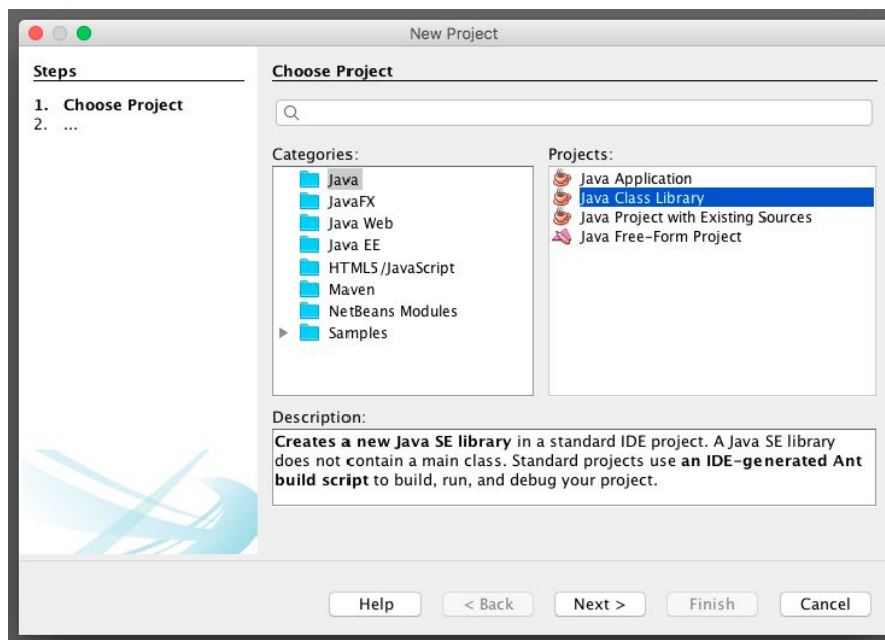
La investigación en los últimos años en el campo de los SMA está sufriendo un crecimiento exponencial, sobre todo por el auge de la cantidad de campos en los se pueden abordar en comunicación, aprendizaje, tolerancia a fallos, fiabilidad, cooperación, coordinación, etc. Con respecto a aplicaciones podemos verlos funcionando en juegos, simulación, películas, defensa, transporte, logística, gráficos, etc.

## 2.2. Creación del Primer Agente

Esta práctica servirá para tener una primera toma de contacto con los SMA que se desarrollarán en mayor profundidad en la asignatura de tercer curso: Sistemas Multiagentes.

### 2.2.1. Creación del proyecto en Netbeans

Vamos a presentar la estructura general que deberán tener todos los proyectos que se crearán para las prácticas. El entorno de desarrollo elegido es NetBeans 8.2, aunque será válido siempre que sea NetBeans 8.x. No estará permitido otro entorno de desarrollo en las prácticas. Además, debemos asegurarnos de que la versión de Java sea la 1.8. Mediante el asistente seleccionaremos la opción de creación de un proyecto para una biblioteca Java:

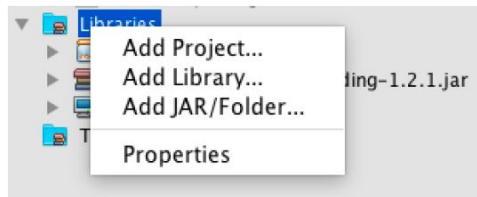


Seleccionaremos un nombre representativo para nuestro proyecto Prueba y completamos la configuración del proyecto asegurándonos que se crea la carpeta `./lib` donde se incluirán las bibliotecas necesarias para el desarrollo del proyecto que no sean las propias de Java, es decir, activar el checkbox *“Use dedicated folder for Storing Libraries”*.

Una vez finalizada la configuración del proyecto debemos asegurar la inclusión de la biblioteca de la plataforma multiagente (`jade.jar`) en el proyecto. Para



ello deberemos ir a la carpeta de bibliotecas del proyecto, y con el botón derecho del ratón, seleccionar la inclusión del fichero “.jar”. Localizamos el archivo “jade.jar” y seleccionamos la opción de copiarlo al directorio “./lib” de nuestro proyecto con la opción “*Add JAR/Folder*”



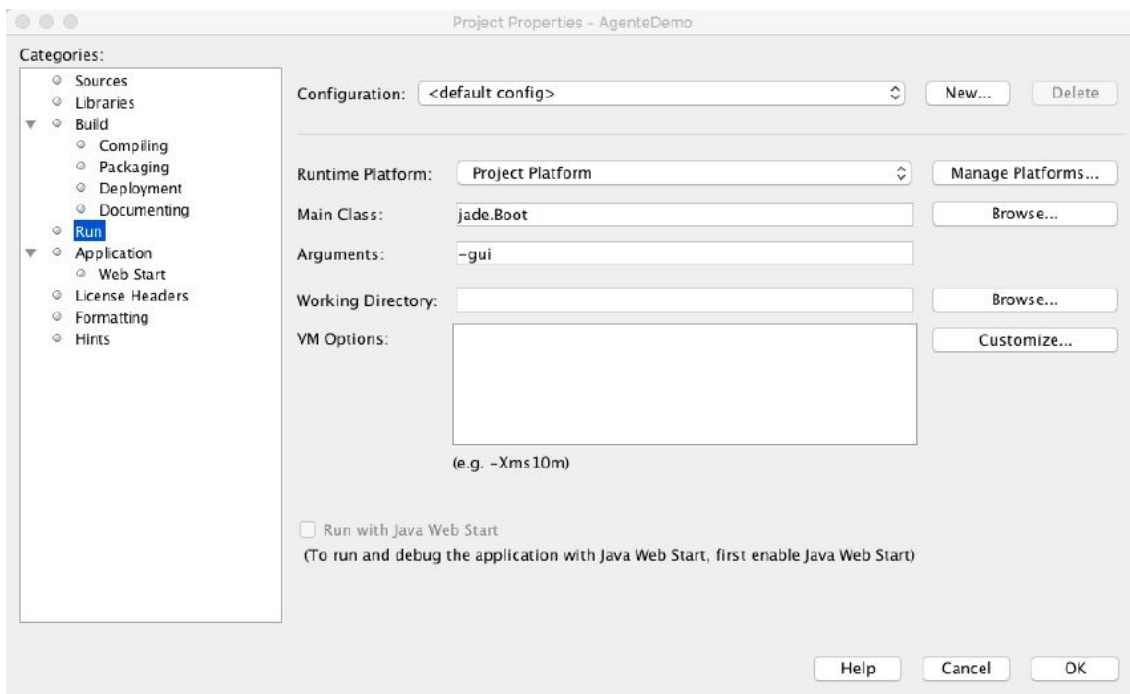
### ¿Qué es JADE?

JADE es el pseudónimo de **Java Agent DEvelopment Framework**.

JADE es una plataforma software para el desarrollo de agentes en Java. Esta plataforma soporta la coordinación de múltiples agentes y proporciona implementaciones estándar de lenguajes de comunicación.

De esta forma ya tendremos configuradas las bibliotecas necesarias para el desarrollo de nuestro proyecto.

El siguiente paso es configurar la ejecución de nuestro proyecto para poder realizar las pruebas necesarias del mismo. Para ello seleccionamos las propiedades del proyecto y localizamos la de ejecución. Configuramos esta propiedad con las opciones que se muestran en la imagen.



Para finalizar la configuración de nuestro proyecto debemos crear el paquete donde se incluirá la clase que permita resolver el problema que tengamos planteado.

- Paquete `agentes`: en este paquete incluimos todas las clases que van a representar a los agentes presentes en la solución de nuestro problema.

De esta forma ya tenemos el proyecto NetBeans finalizado.

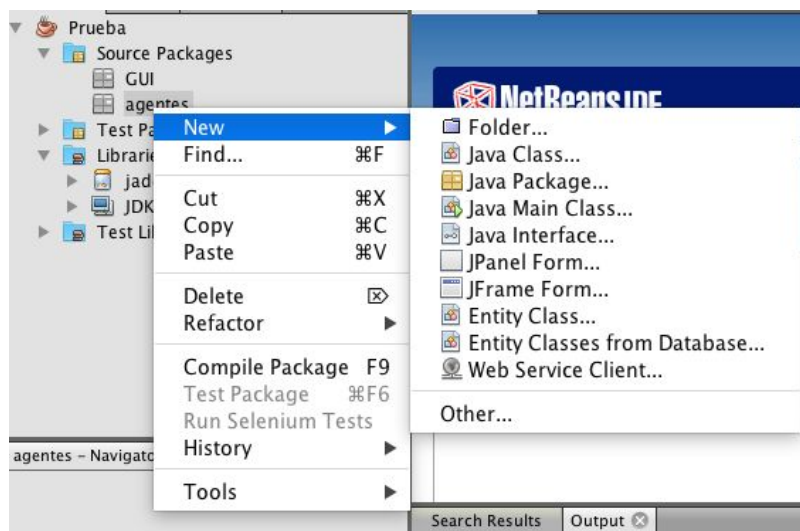
### 2.2.2. Estructura para una Clase de Agente

Ahora se presenta la estructura general que deberá tener una clase que representará nuestro agente en el proyecto. Esta estructura se utilizará como punto de partida y posteriormente se puede personalizar para adaptarla al propósito que deba tener nuestro agente en el sistema.

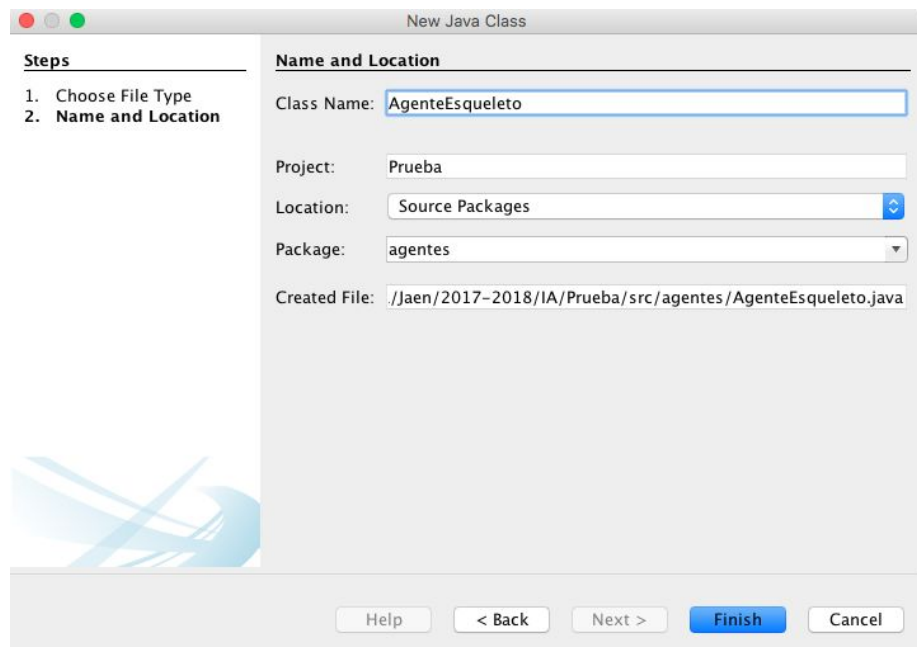
Como norma de estilo, todas las clases que representen a un agente deberán estar incluidas en el paquete `agentes` de vuestro proyecto y siempre empezarán por la palabra `Agente`.

En el ejemplo que nos ocupa será `AgenteEsqueleto` que representa la estructura que deberá tener cada uno de los agentes que creamos en nuestros proyectos.

Para la creación de una clase se selecciona con el botón derecho encima del paquete en el que deseas incluirla, por ejemplo:



Y seleccionas la opción de `Java Class`. Como nombre de la clase vamos a utilizar el nombre `AgenteEsqueleto`.



## Declaración del Agente

```
/**
 *
 * @author pedroj
 * Esqueleto de agente para la estructura general que deben tener todos los
 * agentes
 */
public class AgenteEsqueleto extends Agent {
    //Variables del agente
}
```

Como podéis comprobar en la imagen anterior para la declaración del `AgenteEsqueleto` se utiliza la cláusula `extends Agent`.

Sin embargo, **no funciona** porque necesitamos importar el paquete `jade` en la clase. Para ello, necesitas incluir la sentencia `import jade.core.Agent` justo debajo de la definición del paquete.

Una vez definido se pueden incluir las variables que sean necesarias teniendo en cuenta que las variables deben ser privadas. Además, es importante destacar que NO es necesario ningún constructor. Solo debemos sobrescribir los métodos `setup` y `takeDown`.

## Método setup()

Éste es el método donde estará la inicialización del agente y la inclusión de las tareas principales del mismo.

```
@Override
protected void setup() {
    //Inicialización de las variables del agente

    //Configuración del GUI

    //Registro del agente en las Páginas Amarillas

    //Registro de la Ontología

    System.out.println("Se inicia la ejecución del agente: " + this.getName());
    //Añadir las tareas principales
}
```

En la clase con la que estamos trabajando (se muestra una plantilla más adelante) se describe el orden que deberá seguir los distintos elementos presentes en el método. No todos los elementos tienen por qué estar presentes en todos los agentes que se crean.

## Método takeDown()

El método será invocado cuando el agente finalice su ejecución o el programado invoque el método `doDelete()` de forma explícita.

```
@Override
protected void takeDown() {
    //Desregistro del agente de las Páginas Amarillas

    //Liberación de recursos, incluido el GUI

    //Despedida
    System.out.println("Finaliza la ejecución del agente: " + this.getName());
}
```

En la imagen se muestran las acciones que deberán hacerse en este método, son todas aquellas necesarias para que la finalización del agente se realice de forma ordenada y sin dejar recursos pendientes en el sistema.

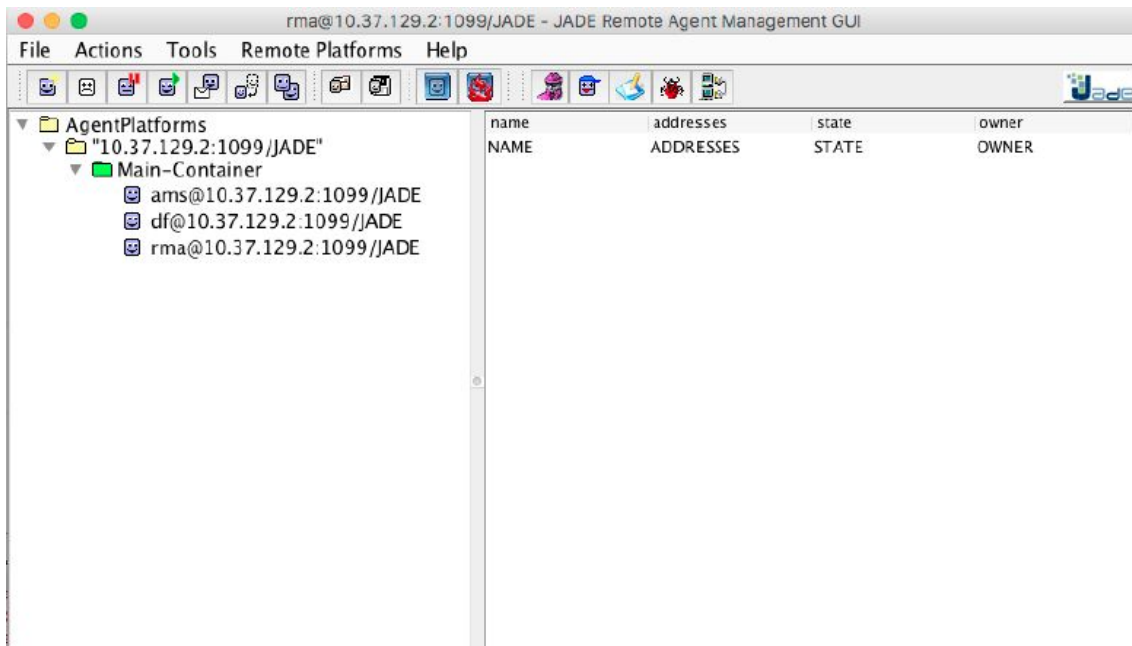
Para completar la clase del agente se incluirán los métodos propios para su normal funcionamiento, incluyendo los métodos de acceso para los atributos definidos previamente. Por último, se incluirán las clases internas que representan las tareas propias que el agente deberá llevar a cabo. Más adelante se mostrará un ejemplo relativo y en posteriores guiones veremos más claramente lo que quiero decir.

## ¿Cómo quedaría el AgenteEsqueleto?

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package agentes;
7  import jade.core.Agent;
8
9  /**
10   *
11   * @author pedroj
12   * Esqueleto de agente para la estructura general que deben tener todos los
13   * agentes
14   */
15  public class AgenteEsqueleto extends Agent {
16      //Variables del agente
17
18      @Override
19      protected void setup() {
20          //Inicialización de las variables del agente
21
22          //Configuración del GUI
23
24          //Registro del agente en las Páginas Amarillas
25
26          //Registro de la Ontología
27
28          System.out.println("Se inicia la ejecución del agente: " + this.getName());
29          //Añadir las tareas principales
30      }
31
32      @Override
33      protected void takeDown() {
34          //Desregistro del agente de las Páginas Amarillas
35
36          //Liberación de recursos, incluido el GUI
37
38          //Despedida
39          System.out.println("Finaliza la ejecución del agente: " + this.getName());
40      }
41
42      //Métodos de trabajo del agente
43
44
45      //Clases internas que representan las tareas del agente
46
47  }
```

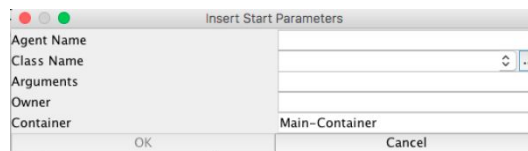
## Ejecución de un agente

Si ejecutamos el proyecto deberá aparecer en pantalla la siguiente imagen:

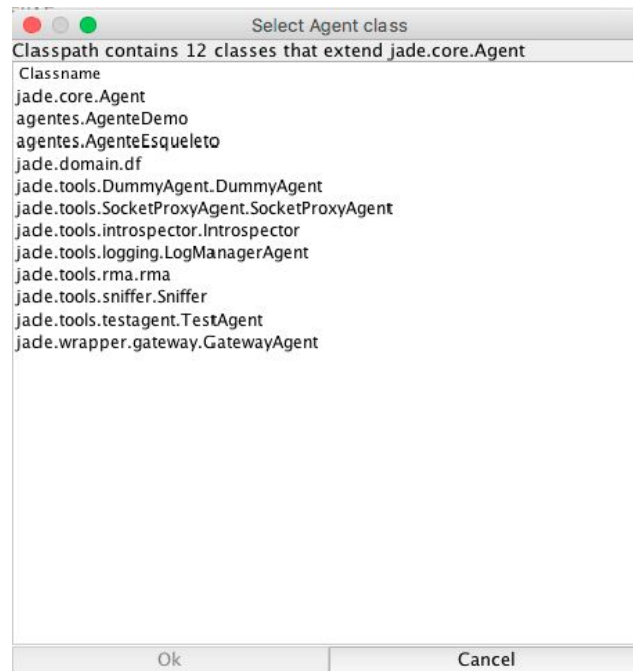


Lo que se muestra en la imagen es el contenedor **Main-Container** con tres agentes **ams**, **df** y **rma** que presenta una interfaz gráfica para la plataforma de agentes. De esta forma podremos trabajar con ella de una manera más cómoda.

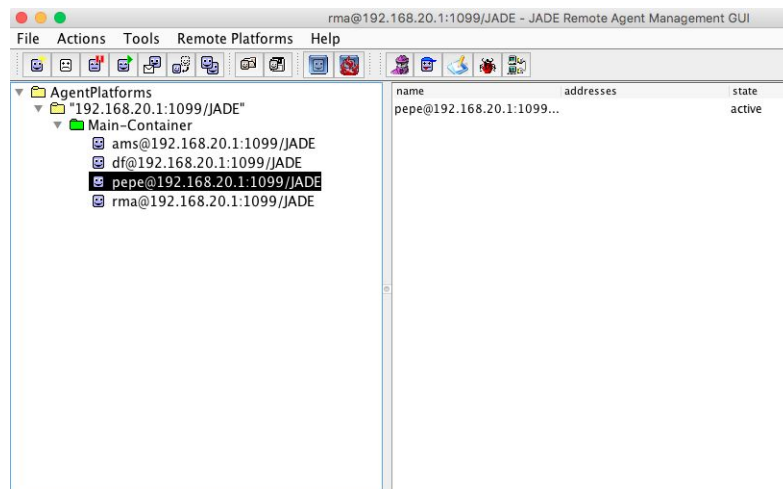
Si queremos ejecutar un agente, debemos seleccionar el contenedor donde lanzarlo (en nuestro caso en **Main-Container**) y pulsar en el icono para crear un agente:



En la figura anterior vemos la ventana que se crea para la creación del agente. Debemos elegir un nombre y la clase a la que pertenece al agente. Para ello podemos pulsar en el botón desplegable que nos mostrará todas las clases de agente que conoce nuestro proyecto.

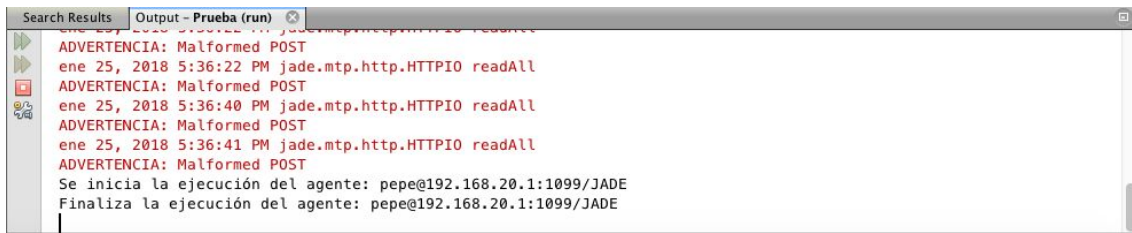


Si completamos todos los pasos de forma apropiada se mostrará en el contenedor nuestro nuevo agente:



Es muy importante también controlar la consola de NetBeans que es donde aparecen los mensajes que hemos configurado en nuestros métodos, tal y como se muestra en la figura:





```
ene 25, 2018 5:36:22 PM jade.mtp.http.HTTPIO readAll
ADVERTENCIA: Malformed POST
ene 25, 2018 5:36:40 PM jade.mtp.http.HTTPIO readAll
ADVERTENCIA: Malformed POST
ene 25, 2018 5:36:41 PM jade.mtp.http.HTTPIO readAll
ADVERTENCIA: Malformed POST
Se inicia la ejecución del agente: pepe@192.168.20.1:1099/JADE
Finaliza la ejecución del agente: pepe@192.168.20.1:1099/JADE
```

## 2.3. Tareas a resolver

En esta primera práctica de multiagentes el trabajo a resolver tiene un tiempo estimado de trabajo de 1 hora, por lo que se considera que puede resolverse en clase.

Se debe entregar un documento con la descripción gráfica y textual de las distintas tareas que se detallan a continuación:

1. Crear el `AgenteEsqueleto`.
2. Realizar su ejecución en NetBeans.
3. Analizar el resultado devuelto tanto en la interfaz gráfica como en la consola.
4. Crear un nuevo agente con tu nombre y mostrar el resultado.
5. Modificar el resultado para que en la consola se muestre el agente con tu nombre sin la IP asociada y, además, muestre el nombre del contenedor en el que se encuentre ubicado. Por defecto este contenedor es "Main-Container". El mensaje a mostrar sería similar al siguiente:

*"Hola amigos, soy <tu\_nombre>. Acabo de iniciar mi ejecución y estoy en <nombre\_contenedor>."*

## 3. Normativa

Lee atentamente la normativa de entrega de prácticas:

- Solo se admitirá el formato **PDF**. No se corregirán guiones en cualquier otro formato al indicado.
- El documento incluirá una portada con:
  - o Identificación de los dos alumnos (Nombre, apellidos y DNI).
  - o Identificación del guión.
- El nombre del fichero tendrá el siguiente formato *"Ape11-Ape12-Ape21-Ape22-GuionX.pdf"* donde
  - o Ape11 es el primer apellido del primer alumno.
  - o Ape12 es el segundo apellido del primer alumno.
  - o Ape21 es el primer apellido del segundo alumno.
  - o Ape22 es el segundo apellido del segundo alumno.
  - o X es el número del guión.

## 4. Entrega y evaluación

La puntuación máxima de la práctica será de 1 punto. La entrega se llevará a cabo a través de la actividad correspondiente en ILIAS. El plazo de entrega termina el 13 de febrero de 2020 a las 23:59.