GUIÓN 0

INTRODUCCIÓN A JAVA E INTRODUCCIÓN A LOS AGENTES ARTIFICIALES

Javier Gutiérrez Rodríguez

Grupo 4 (17:30)

Introducción a Java

En esta primera parte del guion, a modo de introducción al lenguaje de programación Java, se nos pide que creemos un programa con una series de características para aprender a desenvolvernos con el lenguaje y el framework utilizado.

El programa se basa en el uso básico de ciertas habilidades como la creacion de clases (trabajando a su paso la herencia de clases), la inicialización de datos, la lectura de datos y escritura de datos de ficheros.

El punto 1 se soluciona como en otros lenguajes de programación, la diferencia es a la hora de declarar atributos ya que en Java se hace de forma individual. Cada método o atributo se debe declarar de forma individual si es publico o privado. Por lo demás se trata de crear constructores como se haría en C. El segundo punto se trata de leer la información del teclado, el cual se hace mediante un objeto de la clase *Scanner*. Utilizando la función *next()* se leerá de la entrada estándar (el teclado) una serie de caracteres hasta que se pulse la tecla Enter.

El tercer punto trata sobre el tema de la herencia de clases. En este caso hay que definir una subclase de la clase Alumno (creada en el apartado 1). La solución es hacer que Alumno_IA extienda a Alumno. Para llamar al constructor de la clase padre se utiliza la función *super()*.

El apartado 4 se realiza de la misma manera que el apartado 2, pero utilizando la funcion *nextDouble()* de la clase *Scanner()*.

El ultimo apartado trata la lectura y escritura de un fichero. Para la lectura utilizamos la clase *File*, y la clase *BufferdReader*. Esta ultima la utilizamos para ir obteniendo cada linea del fichero mediante la función *readLine()*. Una vez cogida la linea utilizamos la función *split(String separador)* para separar el texto mediante separadores, en el caso del fichero dado para la lectura los separadores a utilizar son “ ,”. Mientras se va leyendo cada linea se va creando un nuevo alumno y añadiéndolo a un *ArrayList*. Para la escritura realizamos algo parecido pero con otra función designada a la escritura de archivos como *BufferedWriter*. Por cada linea se escribirá con la función *write()*, y para saltar de linea la función *newLine()*.

Introducción a Agentes Artificiales

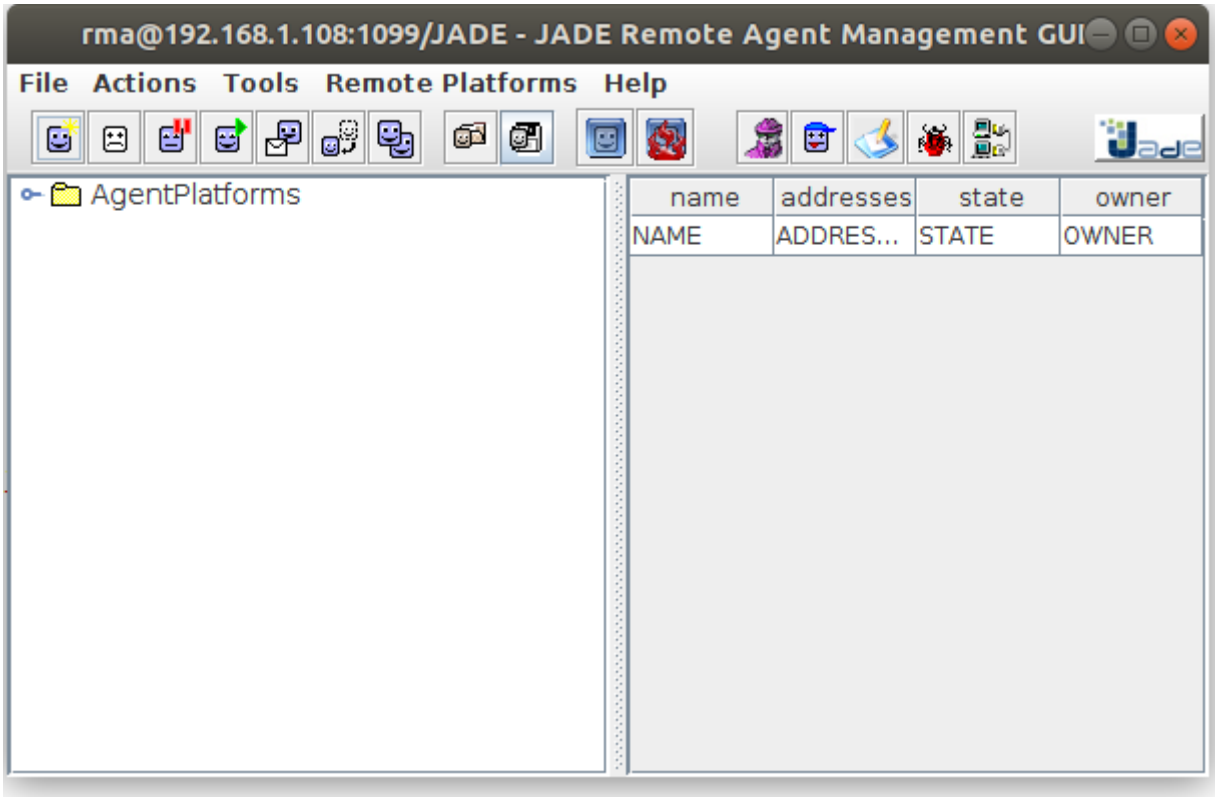
La primera tarea es crear la clase AgenteEsqueleto, para ello creamos la clase como cualquier otra en Java (que no sea main) y le añadimos las funciones de *setup()* y *takedown()*.

```
public class AgenteEsqueleto extends Agent {

    @Override
    protected void setup() {

        System.out.println("Hola me llamo " + this.getName() + " y acabo de ejecutarme.");
    }

    @Override
    protected void takeDown() {
        System.out.println("Hola, me llamo " + this.getName() + " y acabo de morir :c.");
    }
}
```

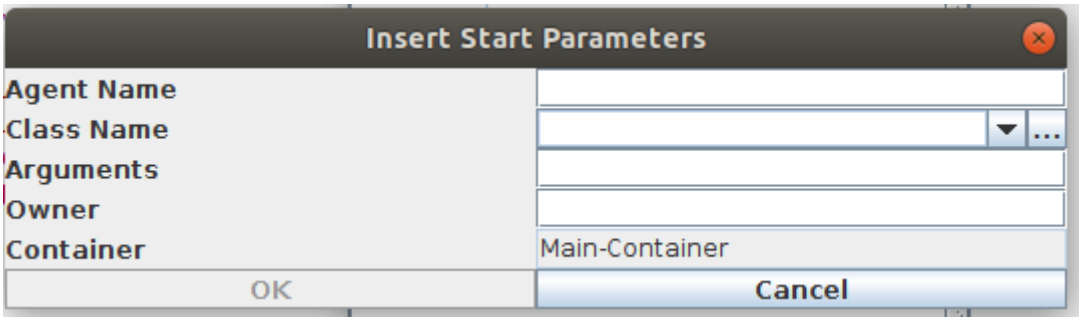
Cuando realizamos la ejecución del programa se lanza una interfaz grafica y unos mensajes en la consola. En la consola podemos observar como se van inicializando los diferentes modulos y recursos de la librería jade.

```
run:
feb 10, 2020 7:50:28 PM jade.core.Runtime beginContainer
INFORMACIÓN: ----------------------------------
    This is JADE 4.4.0 - revision 6778 of 21-12-2015 12:24:43
    downloaded in Open Source, under LGPL restrictions,
    at http://jade.tilab.com/
----------------------------------------
feb 10, 2020 7:50:28 PM jade.imtp.leap.LEAPIMTPManager initialize
INFORMACIÓN: Listening for intra-platform commands on address:
- jicp://192.168.1.108:1099

feb 10, 2020 7:50:29 PM jade.core.BaseService init
INFORMACIÓN: Service jade.core.management.AgentManagement initialized
feb 10, 2020 7:50:29 PM jade.core.BaseService init
INFORMACIÓN: Service jade.core.messaging.Messaging initialized
feb 10, 2020 7:50:29 PM jade.core.BaseService init
INFORMACIÓN: Service jade.core.resource.ResourceManagement initialized
feb 10, 2020 7:50:29 PM jade.core.BaseService init
INFORMACIÓN: Service jade.core.mobility.AgentMobility initialized
feb 10, 2020 7:50:29 PM jade.core.BaseService init
INFORMACIÓN: Service jade.core.event.Notification initialized
feb 10, 2020 7:50:29 PM jade.mtp.http.HTTPServer <init>
INFORMACIÓN: HTTP-MTP Using XML parser com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl$JAXPSAXParser
feb 10, 2020 7:50:29 PM jade.core.messaging.MessagingService boot
INFORMACIÓN: MTP addresses:
http://lenovojavi:7778/acc
feb 10, 2020 7:50:29 PM jade.core.AgentContainerImpl joinPlatform
INFORMACIÓN: ----------------------------------------
Agent container Main-Container@192.168.1.108 is ready.
--------------------------------------------
```

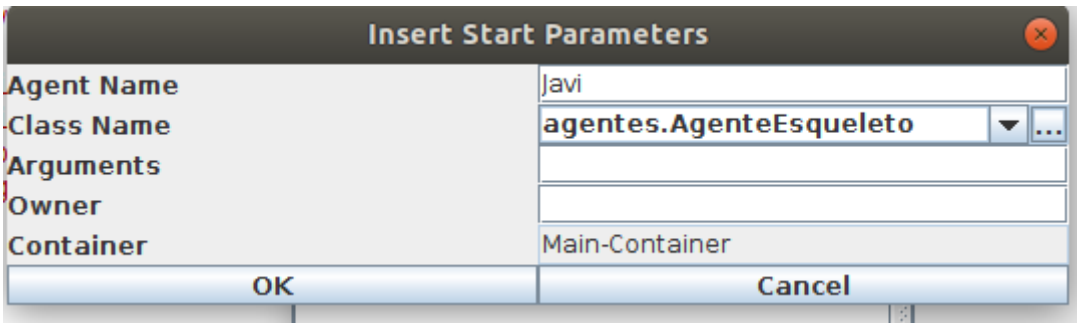
En la parte grafica podemos identificar mas fácilmente los botones y para lo que sirven, como por ejemplo crear un nuevo agente, borrarlo, parar su ejecución, reanudarla, etc.



Para crear un nuevo agente lo que hacemos en seleccionar Main Conteiner y darle al botón de crear un nuevo agente. Nos saldrá la siguiente ventana.



Rellenamos los datos, escogiendo en Class Name la clase agente esqueleto y aceptamos.



Tras esto podemos observar que por la consola sale este mensaje.

```
Hola me llamo Javi@192.168.1.108:1099/JADE y acabo de ejecutarme.
```

Cuando lo borramos sale este mensaje.

```
Hola, me llamo Javi@192.168.1.108:1099/JADE y acabo de morir :c.
```

El apartado final es hacer que nos muestre el nombre del agente sin la IP asociada y que ademas muestre el contenedor en el que se encuentra el agente. Para ello utilizamos la función *getLocalName*(), en vez de *getName*(). Y para mostrar el contenedor en el que se encuentra utilizamos la función que nos devuelve el contenedor, y a su vez utilizamos la función que nos da su nombre.

Finalmente la clase queda asi.

```
 */
public class AgenteEsqueleto extends Agent {

    @Override
    protected void setup() {

        try {
            System.out.println("Hola soy "+ this.getLocalName()+
                    ". Acabo de iniciar mi ejecucion y estoy en " +
                    this.getContainerController().getContainerName());
        } catch (ControllerException ex) {
            Logger.getLogger(AgenteEsqueleto.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    @Override
    protected void takeDown() {
        try {
            System.out.println("Hola me llamo " + this.getLocalName()+
                    ". Acabo de finalizar mi ejecucion y estaba en " +
                    this.getContainerController().getContainerName());
        } catch (ControllerException ex) {
            Logger.getLogger(AgenteEsqueleto.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

Y por la consola se ve el resultado.

```
Hola soy Javi. Acabo de iniciar mi ejecucion y estoy en Main-Container
Hola me llamo Javi. Acabo de finalizar mi ejecucion y estaba en Main-Container
```