Grado en Ingeniería Informática

Inteligencia Artificial

Curso 2022/2023

Universidad de Jaén

Práctica 1:
Java
Sistemas Multiagentes

Juan Carlos González Martínez - 77691291M
Raúl Gómez Téllez - 77647571P

EJERCICIO 1

1. **Definir una clase Alumno, que contenga las siguientes características: Nombre y Apellido, DNI y correo electrónico. 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.**

Para comenzar, manifestamos la clase Alumno como pública dentro de un nuevo archivo, Alumno.java, y comenzamos a declarar sus variables (nombre, apellido, dni y correo), las cuales tratarán en cadenas de caracteres, string, privados, así como constantes mediante la llamada *private final string*.

A continuación sobrecargaremos el constructor de la clase mediante la creación de: Un constructor por defecto, que inicializará las variables de la clase a cadenas vacías (“ ”).Así como un constructor parametrizado, que dados los valores para cada una de las variables en la llamada al constructor, se las asignará; para ello, realizamos la llamada a la propia clase mediante el uso de la palabra reservada *this* de la forma: *this.variable=variable*.

2. **Definir un método que lea una serie de valores que se introduzcan por teclado y cree un nuevo objeto de la clase Alumno, inicializado con dichos valores. Definir otro método que muestre los datos de ese alumno por pantalla.**

Dentro de la clase principal main, en el archivo Main.java, creamos una nueva función, *leerDatos,* que devolverá un objeto nuevo de tipo alumno. La función inicializará primeramente 4 strings, correspondientes a las variables de Alumno, y un objeto de tipo Scanner (Para lo que tendremos que importar la librería mediante la línea *import java.util.Scanner;* al comienzo de nuestro programa), mediante el empleo de *new* y una llamada a su constructor parametrizado empleando *System.in*, para especificar la entrada de consola de teclado estándar.

Para cada una de las variables, mostraremos un mensaje por pantalla solicitando que se introduzca el valor correspondiente, mediante el empleo de *System.out.print()*; usando el objeto tipo Scanner mencionado anteriormente leeremos el valor introducido por teclado, empleando *nextLine()*, y se lo asignaremos a cada variable mediante un simple operador de asignación.

Una vez se hayan leído todas las variables, se llamará y devolverá un objeto nuevo Alumno creado mediante las variables leídas (Empleando el constructor parametrizado que creamos en el apartado anterior).

Para mostrar los datos del alumno por pantalla, primeramente volveremos al archivo Alumno.java, donde crearemos una nueva función llamada *toString()*, la cual devolverá una cadena de caracteres formada por los datos del alumno, en el main bastará con crear una función que, mediante *System.out()*, llame y muestre la cadena devuelta por esta nueva función.

3. **Definir una clase Alumno_IA, subclase de la anterior, que contenga los siguientes campos adicionales: grupo de teoría, grupo de prácticas, nota asociada a las prácticas 1, 2, 3 y 4.**

En un nuevo archivo llamado Alumno_IA.java y siguiendo el procedimiento del primer apartado, creamos una clase pública de nombre Alumno_AI , esta vez incluimos en la declaración de clase *extend Alumno*, con esto declaramos la relación de herencia de la clase actual a la clase Alumno.

A continuación declaramos las nuevas variables de esta clase, enteros que por su naturaleza no declararemos constantes, así como los dos constructores correspondientes: El constructor por defecto permanecerá vacío, pues las variables se declararán automáticamente así como la llamada al constructor por defecto de su clase padre. Y el constructor parametrizado, que recibirá las variables de esta clase, así como las 4 variables de la clase que hereda, en el cuerpo del constructor realizaremos la llamada al constructor de la superclase mediante el empleo de la función *super()*, así como la declaración de variables usando la palabra reservada *this* como ya mencionamos en el primer apartado.

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.**

Antes de comenzar con el método, crearemos una constante llamada *MAX_NOTAS* que se corresponderá con las notas del alumno, en este caso 4, de esta forma, en caso de que se produzca un cambio a dicho valor, solo habrá que cambiar el asignado a esta constante.

Crearemos un nuevo método en main, al que llamaremos *leerNotas*, será de tipo void y recibirá como parámetro un objeto de la clase Alumno_IA, que será el alumno dueño de las notas, declararemos 4 variables para el cálculo de la media: dos enteros (Un contador y la nota actual),un valor en coma flotante que será la media y un Scanner que leerá la nota actual.

Crearemos 2 bucles, el primero se ejecutará mientras el contador sea menor al número de notas máximo, es decir, se ejecutará tantas veces como el máximo de notas sea (La constante previamente mencionada) y añadirá a la variable *media* la nota actualmente leída. A continuación ejecutamos un segundo bucle en el interior de este que se ejecutará mientras la nota leída no cumpla el requisito de estar en el rango (0-4), para su lectura emplearemos el mismo método que en el apartado segundo mediante el uso de un scanner y asignando el valor a la variable *nota* (Que una vez finalice este bucle se sumará a la *media* como hemos mencionado anteriormente).

Una vez termine el bucle, se mostrará por pantalla la nota media, realizando el simple cálculo de la media: La suma total de las notas (Almacenada en la constante *media*) partido el número de estas (La constante MAX_NOTAS).

5. **Se dispone de un archivo “datos.txt” (en PLATEA) con datos sobre alumnos (un alumno 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.**

Comenzaremos creando una nueva función, llamada *leerFichero*, que recibirá como parámetro el nombre del fichero como string y devolverá un ArrayList de objetos tipo Alumno. A continuación inicializamos el ArrayList que se devolverá y un DataInputStream (DIP), que nos permitirá leer las líneas del fichero; Abriremos un *try{ }catch* por si se produjese una excepción en la apertura y lectura del fichero poder capturarla correctamente, así como mostrarla por pantalla. Crearemos un objeto tipo FileInputStream, que contendrá el contenido del fichero al pasarlo como parámetro en su iniciación, y se lo pasaremos como parámetro a la llamada al constructor del objeto DIP que iniciamos anteriormente.

Mediante la función *readline()* correspondiente al DIP, realizaremos un bucle que se ejecutará mientras la línea leída no sea nula, es decir, no se haya terminado de leer el archivo. En el interior del bucle mostraremos por pantalla la línea leída y se almacenará en un vector de string, marcando la separación entre valores con “,” (Pues en el fichero los datos de los alumnos se separan empleando comas). Llamaremos a la función *add()* del ArrayList de Alumnos para añadir un nuevo alumno que se creará con los parámetros leídos en la línea y almacenados en el vector mencionado.

A continuación crearemos una nueva función, *escribirEnArchivo*, de tipo void y que recibe como parámetros una lista de alumnos en un objeto de tipo ArrayList y un string con el nombre del archivo en el que volcar los datos.

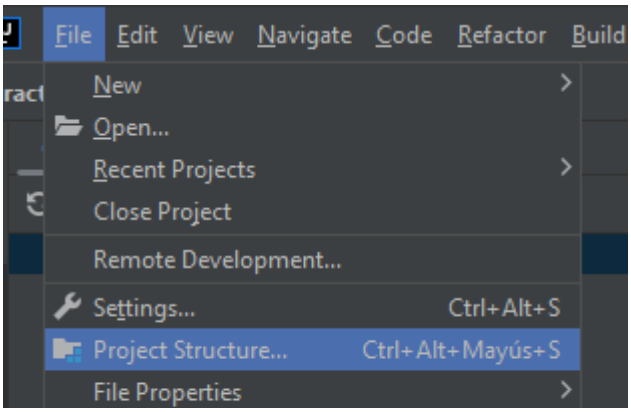
Mediante la creación de un objeto de tipo File, creamos una ruta al archivo destino, y abrimos un *try{ }catch* por si se produjese alguna excepción en los siguientes pasos: Mediante la función *exist()* del objeto file comprobamos su existencia, en caso de que no exista lo crearemos mediante la función propia *createNewFile()*. Creamos dos objetos, *FileOutputStream*(file0) y *DataOutputStream*(d0) que nos permitirá almacenar los datos en la variable d0 que se volcarán en el fichero file0 mediante una llamada al mismo en el constructor parametrizado. A continuación abrimos un bucle *for* que se repetirá tantas veces como alumnos haya:

Dado que el dni de cada alumno es una cadena de caracteres que comienza con un espacio, para comprobar si el dni de los alumnos es par, realizaremos una llamada al getter de la variable dni del alumno actual, mediante la función *replace()* cambiaremos el espacio (“ “) por un vacío (“”), es decir, lo eliminaremos; Mediante la función *parseInt()* cambiamos el tipo de objeto de la cadena del dni sin espacio a un entero, y mediante un módulo de 2 al mismo comprobamos si es par. En caso de que lo sea, bastará con llamar a la función *writeChars()* de d0, que nos permitirá almacenar los datos del alumno mediante una llamada a su método *toString()*, que nos devolverá los datos en forma de cadena de caracteres, sumando un salto de línea (*n*) para mayor legibilidad.

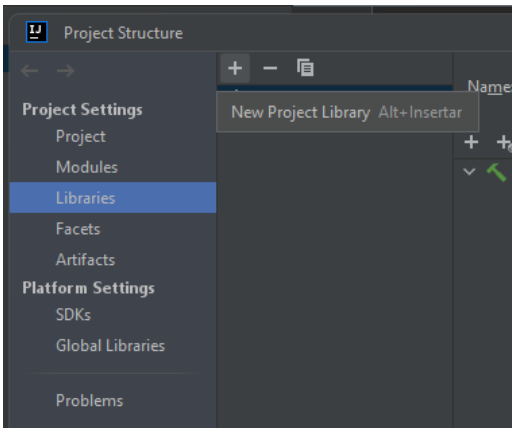
EJERCICIO 2

Paso 1: Añadir la biblioteca Jade a Intellij

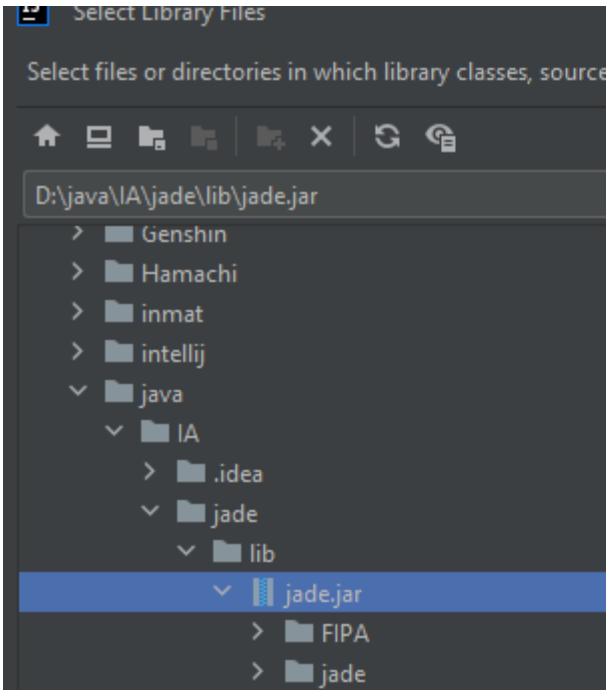
Para esta sección nos vamos a archivo y seleccionamos estructura del proyecto



Tras esto en el menu que nos aparece clicamos en la sección de bibliotecas y pulsamos el + que es el botón de añadir

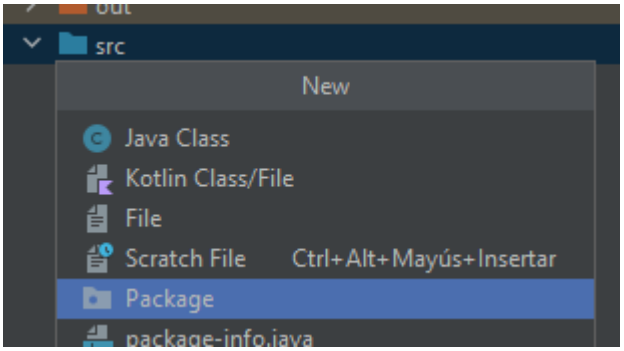


Tras esto seleccionamos el .jar dentro de /jade/lib y pulsamos ok

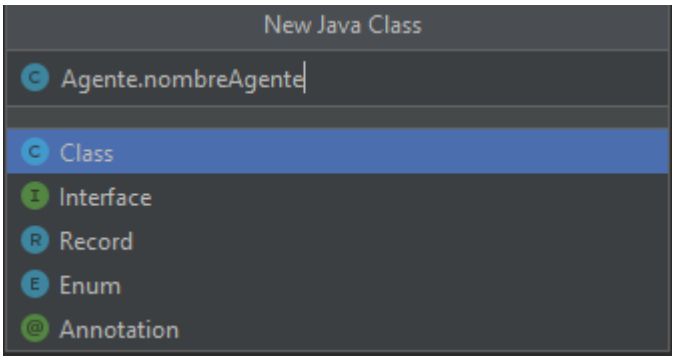


Paso 2 creación del agente

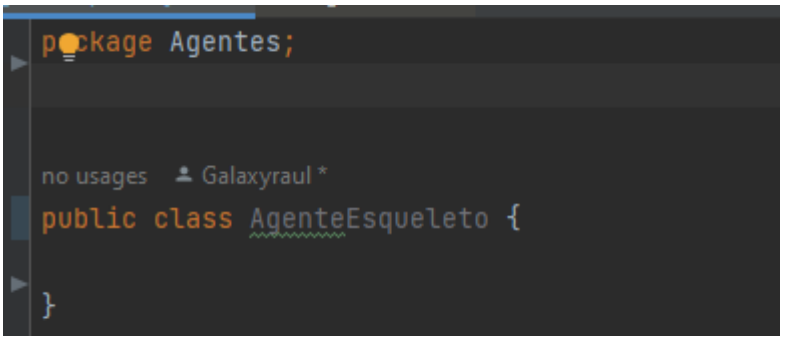
Tras la sección anterior procedemos a crear un nuevo paquete que incluya la palabra agente en este se encontrarán todas las clases que sean agentes cuyos nombres emperara por agente en nuestro caso agente esqueleto para ello pulsaremos alt + ins y seleccionaremos package



Después realizaremos el mismo proceso seleccionando Java class y asignaremos el nombre del agente



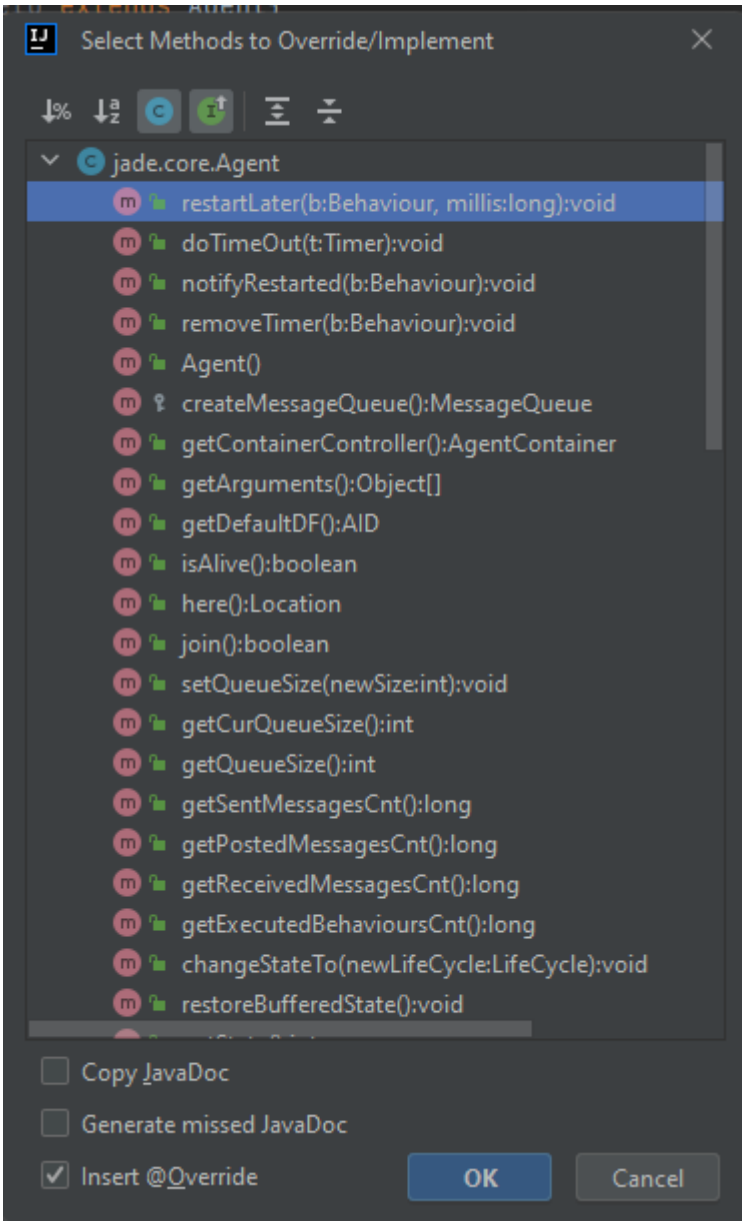
Tras esto se nos quedará una clase tal que así



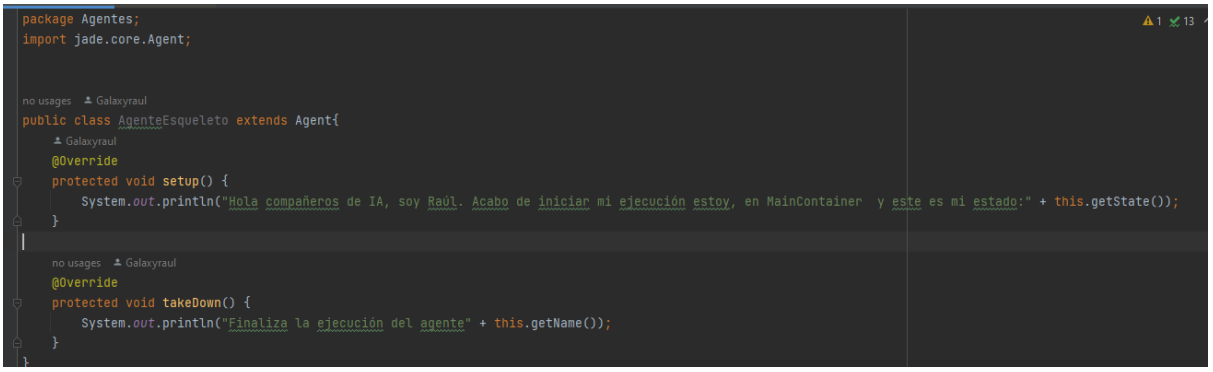
A continuación importaremos el núcleo del agente Jade y pondremos que la clase AgenteEsqueleto herede de Agente

```
package Agentes;
import jade.core.Agent;


no usages  Galaxyraul *
public class AgenteEsqueleto extends Agent{

}
```

Después implementaremos los métodos setup y takeDown heredandolos de agente para ello pulsaremos ctrl + o que nos abrirá un menu de métodos para heredar seleccionamos los adecuados y los sobreescribimos para que queden acorde a los requerimientos de la practica

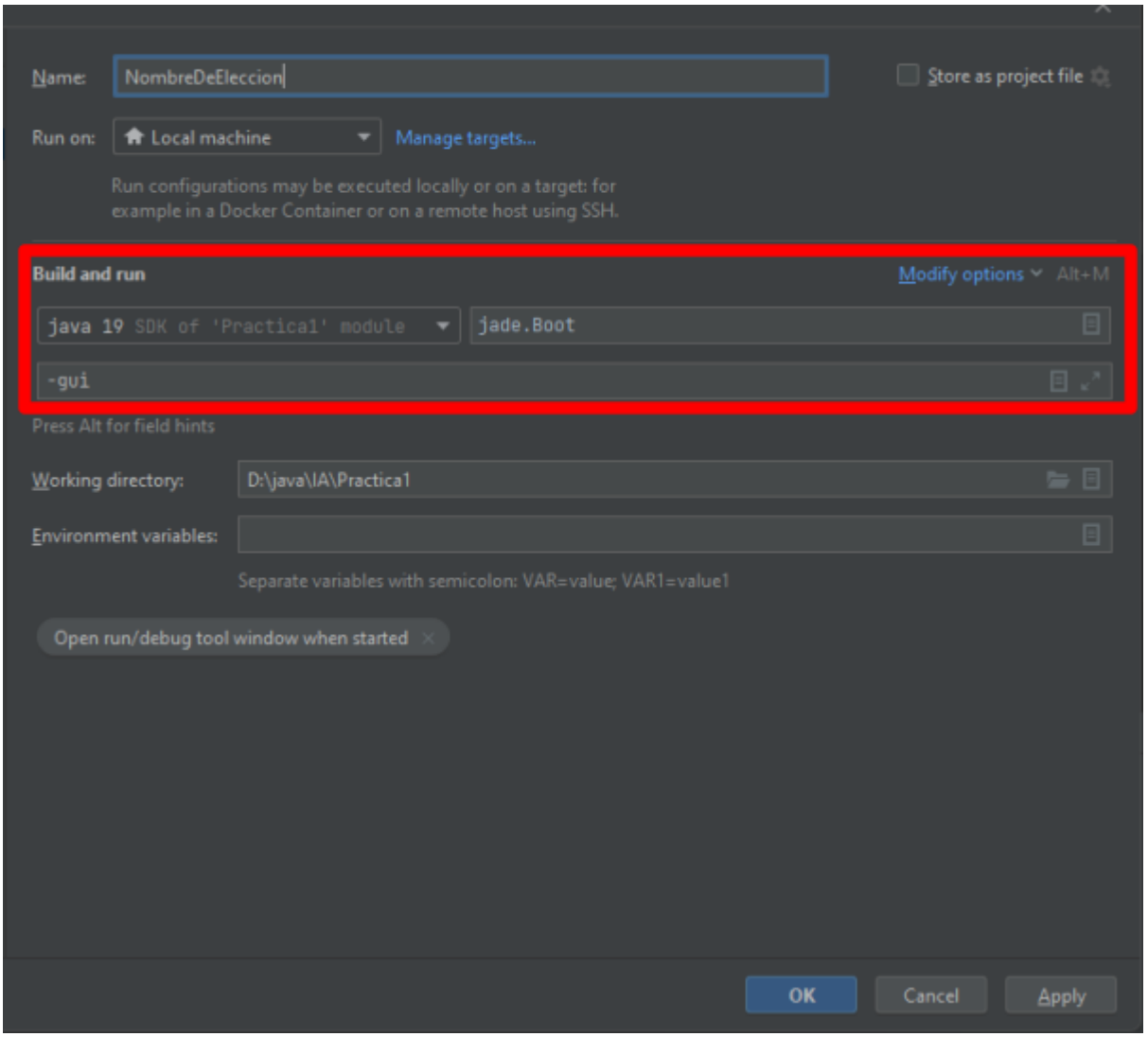


Se nos quedaría tras estos pasos algo parecido a esto

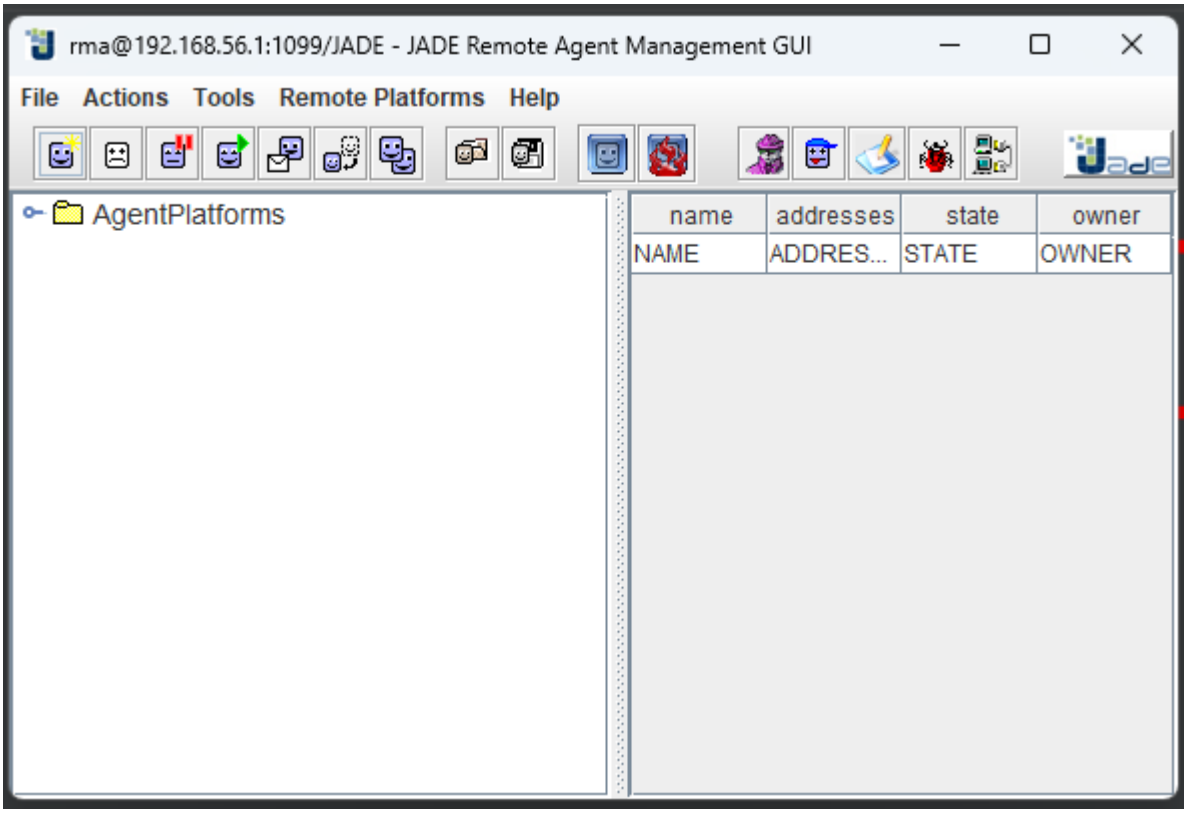


```java
package Agentes;
import jade.core.Agent;


no usages  Galaxyraul
public class AgenteEsqueleto extends Agent{
    Galaxyraul
    @Override
    protected void setup() {
        System.out.println("Hola compañeros de IA, soy Raúl. Acabo de iniciar mi ejecución estoy, en MainContainer  y este es mi estado:" + this.getState());
    }

    no usages  Galaxyraul
    @Override
    protected void takeDown() {
        System.out.println("Finaliza la ejecución del agente" + this.getName());
    }
}
```

Por último para lanzar el programa necesitaremos configurar una nueva configuración de ejecución para ello iremos a run>editConfigurations y pulsamos + y seleccionamos aplicación y escribimos lo que se muestra en la siguiente imagen

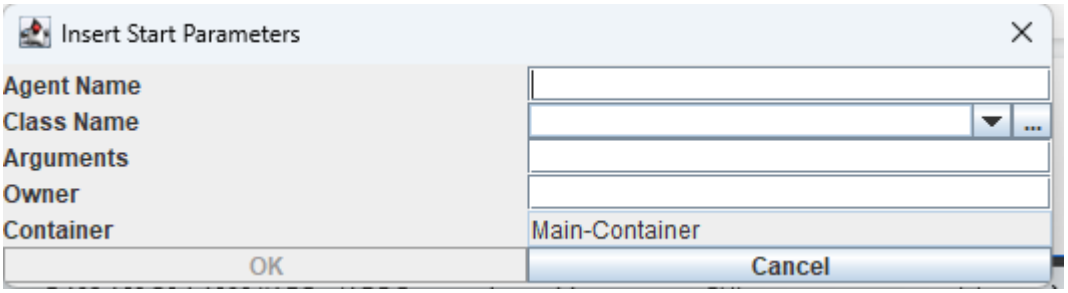


Tras esto pulsamos aplicar y ok ya solo nos queda ejecutar
Se nos abrirá la ventana del agente



Abrimos la carpeta y la siguiente, seleccionamos Main container y hacemos click en Nuevo agente

Se nos abrirá una ventana como esta en la que introduciremos el nombre y con el desplegable la clase a la que pertenece



Tras pulsar ok habremos concluido la configuración de nuestro agente