ENTORNOS DE DESARROLLO – UF3 (ACTIVIDAD B)

¿Qué es y para qué sirve el Lenguaje Unificado de Modelado (UML)?

UML, en inglés “Unified Modeling Language” (Lenguaje Unificado de Modelado) es utilizado a nivel internacional por numerosas organizaciones y empresas para crear esquemas, diagramas y documentación relativa a los desarrollos de software. A pesar de su nombre, no se trata de un lenguaje, es una serie de normas y estándares gráficos respecto a cómo se deben representar los esquemas que hacen referencia a un determinado software, muestran la manera de representar algo. Cuando hablamos de estándares gráficos, hablamos del modelo base sobre el que nos tenemos que basar para realizar alguna ilustración representativa. Dicho esto, conviene destacar que, UML, define normas para construir muchos tipos de esquemas.

A pesar de que cualquier persona estaría en disposición de usar esta herramienta, UML requiere de conocimientos avanzados en programación y suelen ser muy usado por los analistas funcionales y los analista-programadores de un proyecto.

UML cuenta con dos grandes versiones desde que se generó en la década de los 90:

* UML 1.X – Desde finales de los 90, en los años posteriores fueron apareciendo versiones que introducían mejoras o ampliaban la versión.
* UML 2.X – Surgida en 2005 con algunas versiones posteriores.

A día de hoy, se espera que salga la versión UML 3.X, que sucederá a UML 2.X.

El uso de una versión y otra no es determinante para la finalidad, lo que cuenta es que los esquemas estén bien construidos y razonados. Una cosa importante a destacar es que UML es un conjunto muy amplio de normas, tan grande que prácticamente nadie puede conocerlas todas. Cada institución usa la parte de UML que le conviene.

Existen diversos tipos de diagramas en UML:

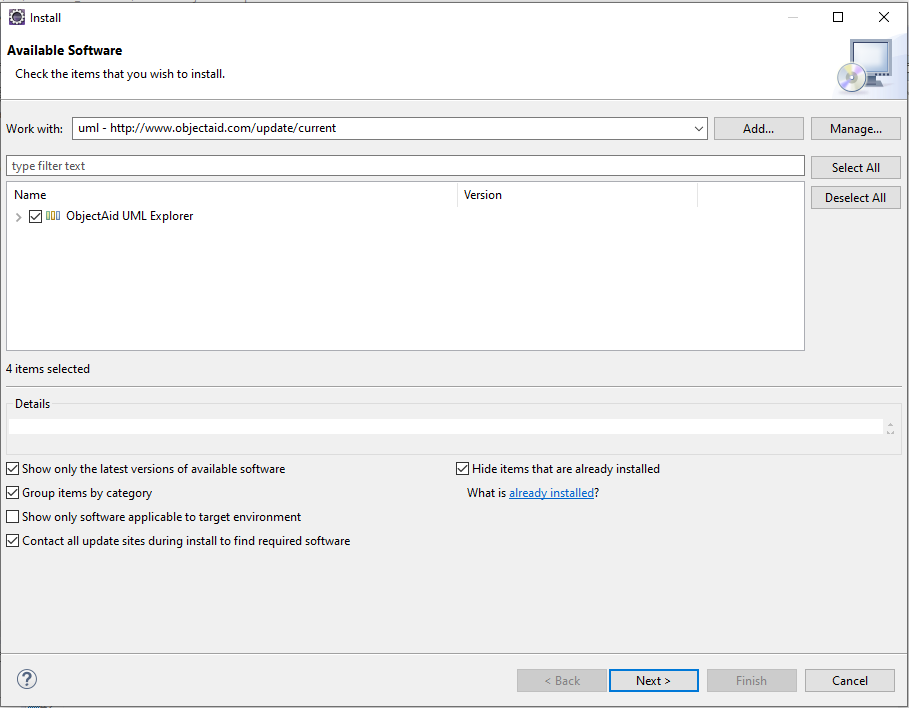
* **Diagramas de casos de uso.** Representan a los actores y casos de uso (procesos principales) que intervienen en un desarrollo de software.
* **Diagramas de clases.** Para UML una clase es una entidad, no una clase software. Un diagrama de clases UML puede ser un diagrama del dominio o representación de conceptos que intervienen en un problema, o también un diagrama de clases software. El sentido de un diagrama UML se lo da la persona que lo construye.
* **Diagramas de secuencia.** Suelen usarse para representar objetos software y el intercambio de mensajes entre ellos, representando la aparición de nuevos objetivos de izquierda a derecha.
* **Diagramas de colaboración.** Suelen usarse para representar objetos o clases y la forma en que se transmiten mensajes y colaboran entre ellos para cumplir un objetivo.
* **Diagramas de estados.** Suelen usarse para representar cómo evoluciona un sistema (cómo va cambiando de estado) a medida que se producen determinados eventos.
* **Otros diagramas.** También existen: diagramas de actividad, diagramas de paquetes, diagramas de arquitectura software, etc.

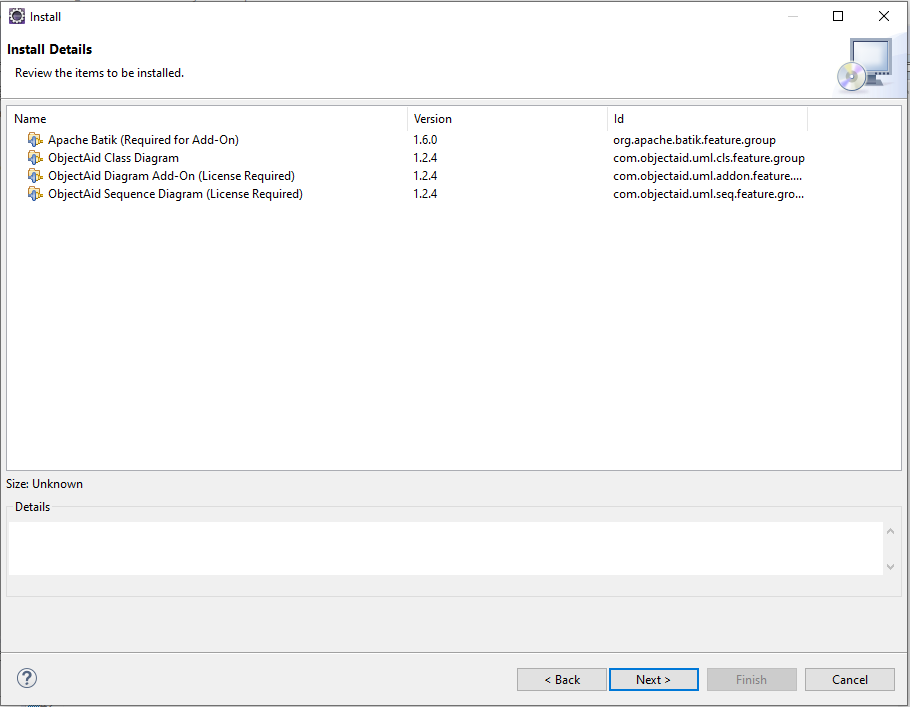
Explica y realiza el Diagrama UML de representación de los siguientes elementos.

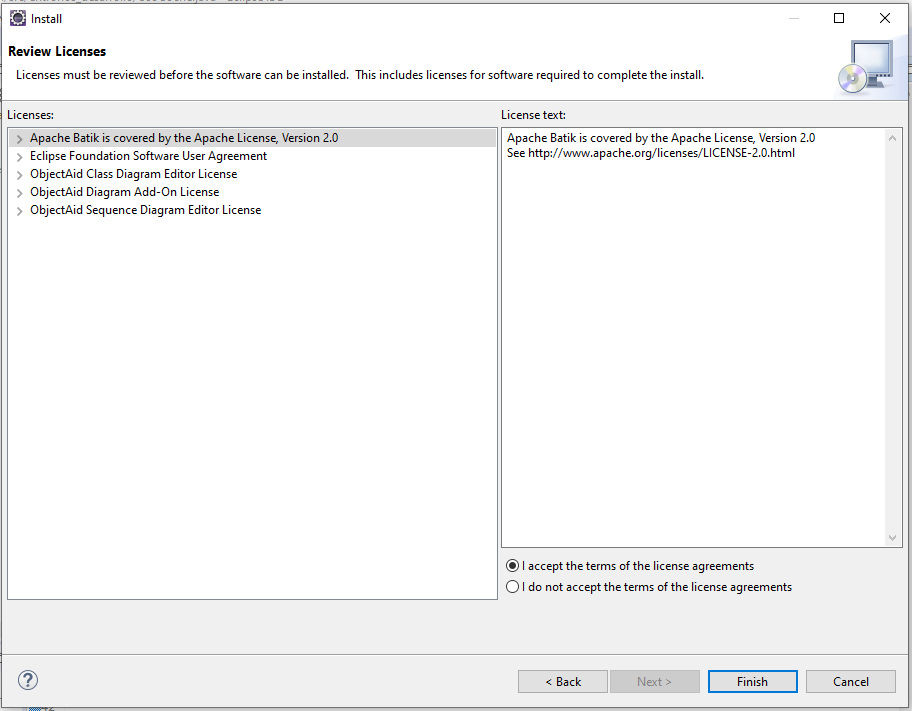
\*Una clase

\*Un objeto

Lo primero que haré es descargar el software pertinente para poder realizar diagramas UML desde el IDE Eclipse. Así pues, vamos a Eclipse y en la pestaña “Help” y después “Install new software”. Una vez ahí, seguimos los siguientes pasos:

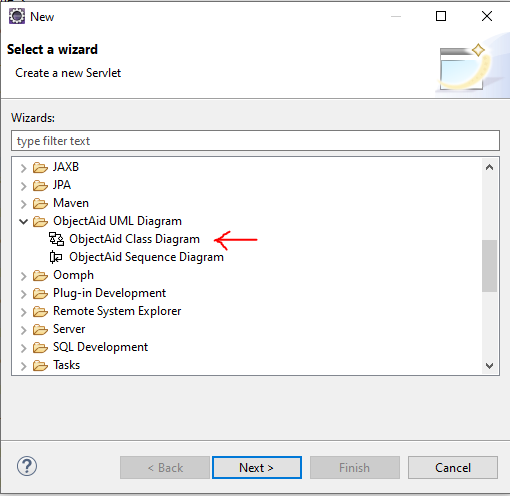




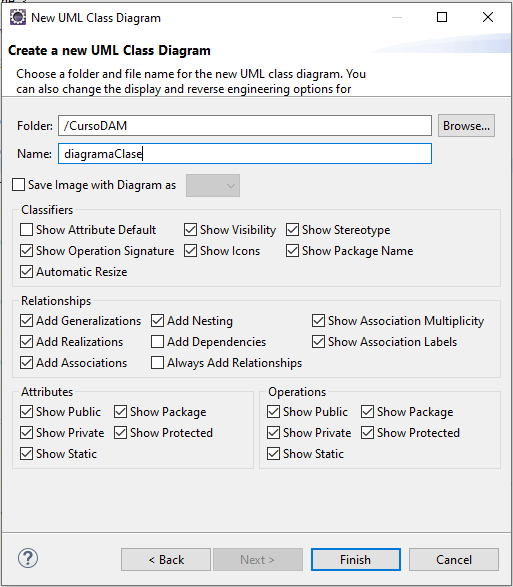


Una vez instalado el software, vamos a activarlo en nuestro entrono de desarrollo.

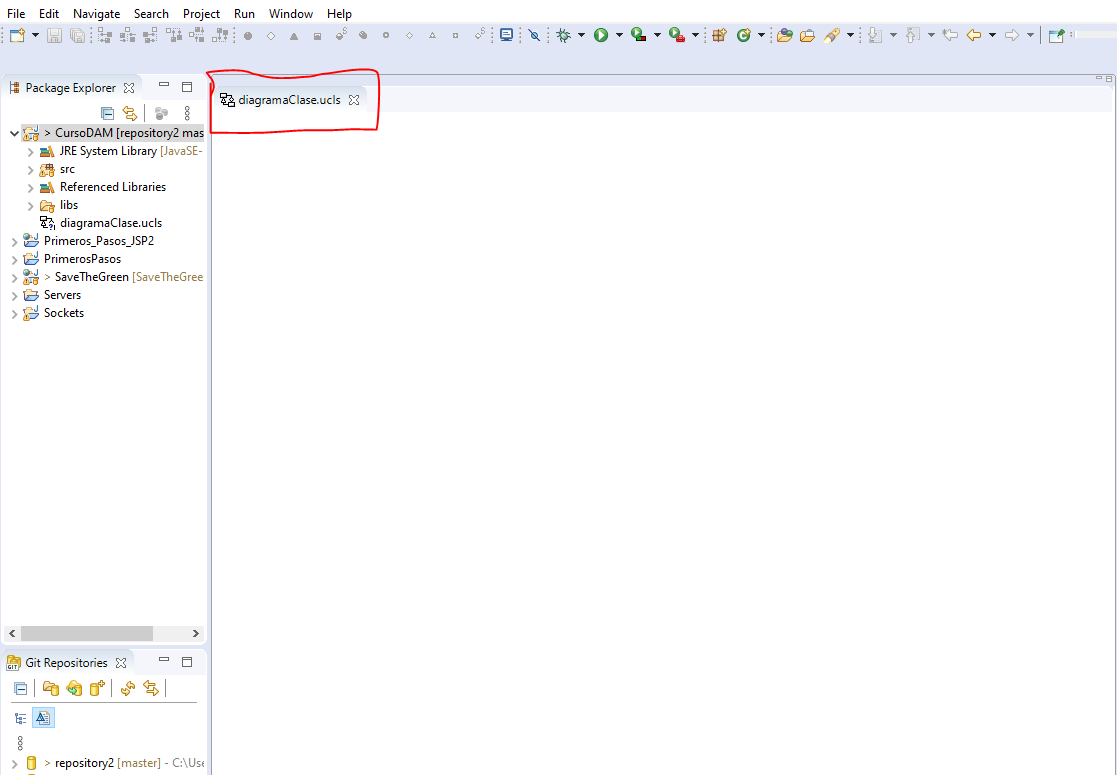
Nos situamos en nuestro proyecto del curso y le damos click con el botón derecho del ratón, vamos a New->Other y elegimos nuestra opción, en nuestro caso queremos hacer un diagrama de una clase.



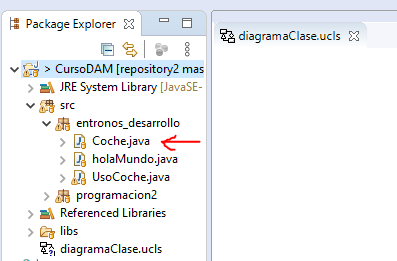
Le ponemos el nombre y aceptamos.



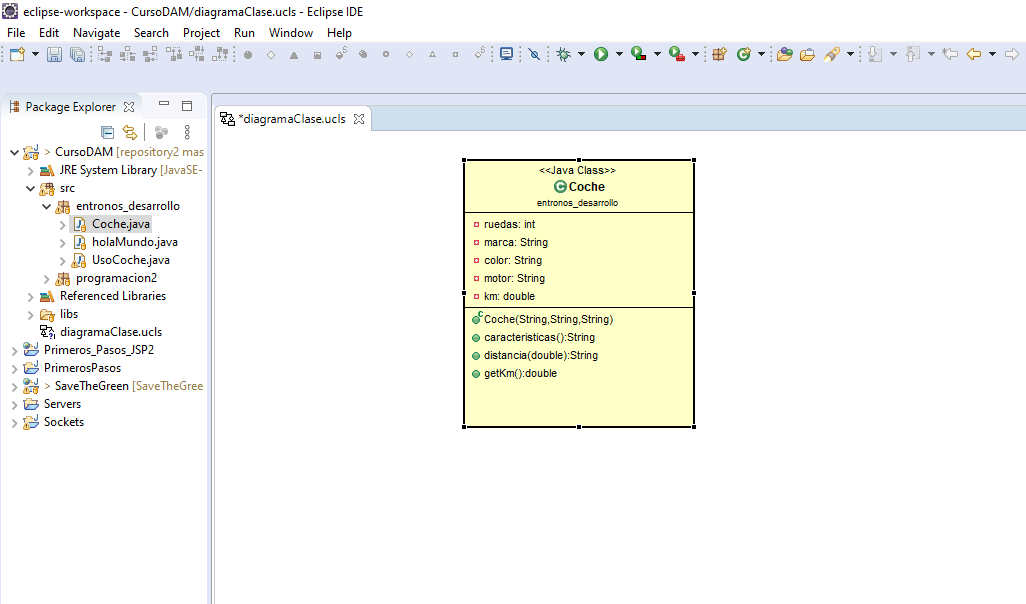
Nos aparece el lienzo en blanco para poder realizar la tarea.



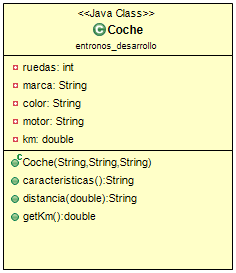
Ahora elegimos la clase de la que queremos hacer un diagrama, en nuestro caso la clase “Coche”.



El diagrama se hará de forma automática, únicamente tenemos que arrastrar el fichero que contiene la clase (en mi caso “Coche.java”) hacia el lienzo de trabajo.



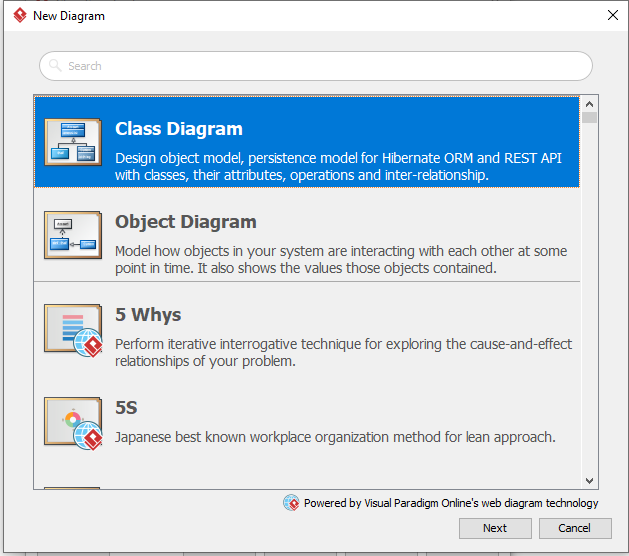
Si quisiéramos guardar la imagen, daríamos click derecho y después “Save as a image” y la guardamos donde queremos.



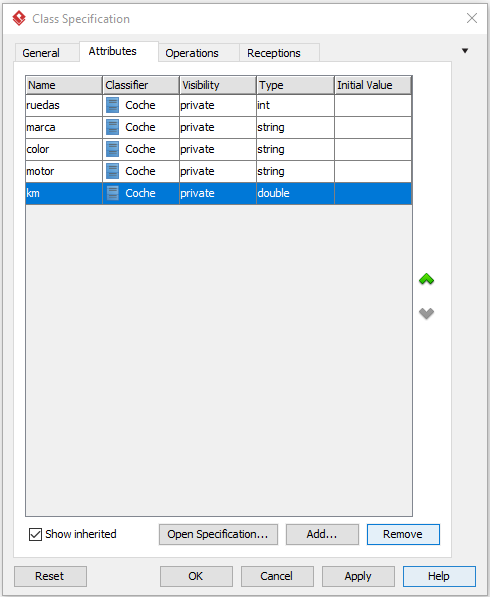
Para la realización del diagrama que representa a un objeto, he optado por hacerlo con el programa “Visual Paradigm”.

Primero realizaré, nuevamente, el diagrama de la clase “Coche” con este software.

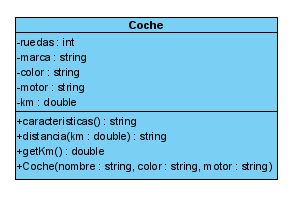
Abrimos Visual Paradigm, accedemos a la pestaña “Diagram”, le damos a “New” y buscamos nuestra opción, en nuestro caso “Class Diagram”.



Procedo a la realización del diagrama



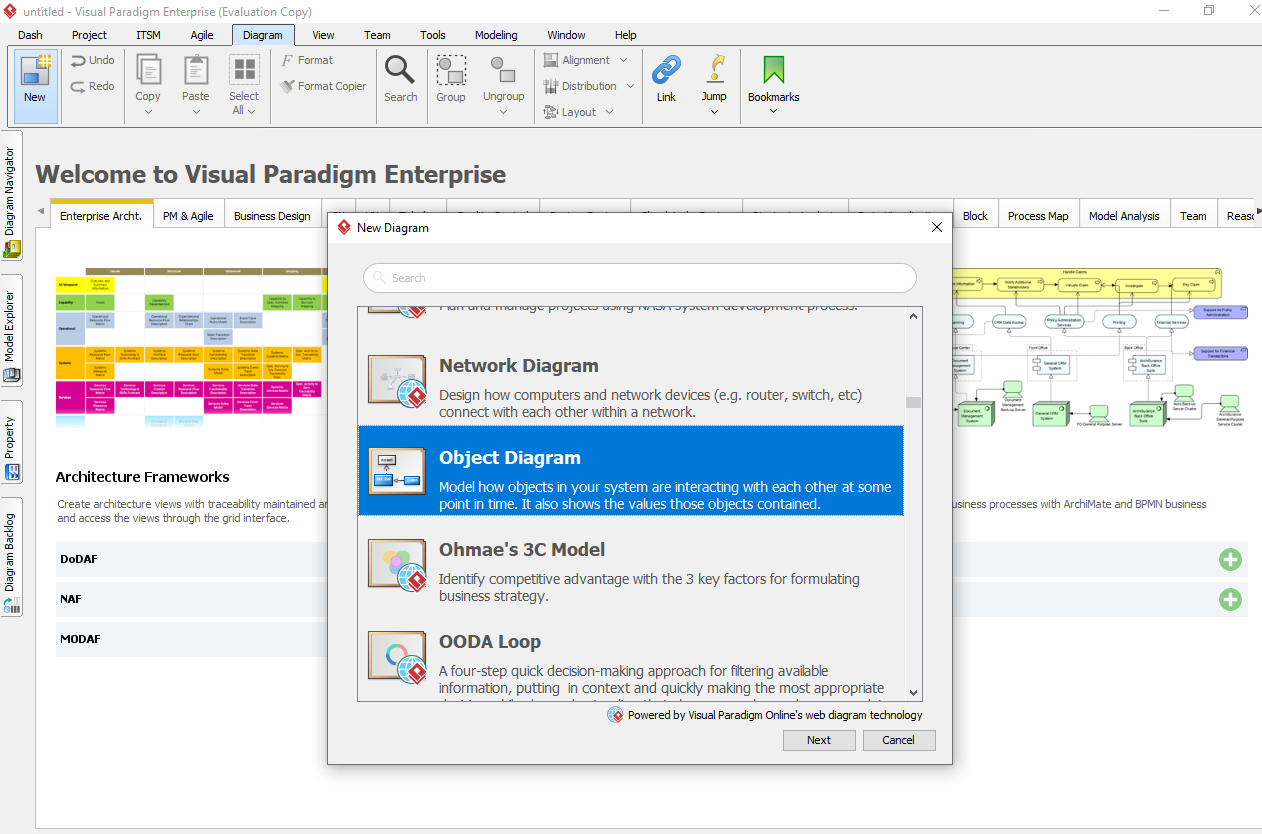
El resultado sería el siguiente

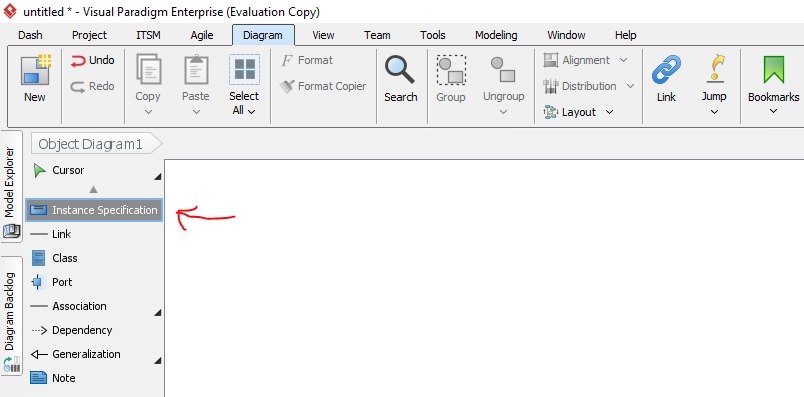


Como se puede ver, la estructura de un diagrama de clase es la siguiente:

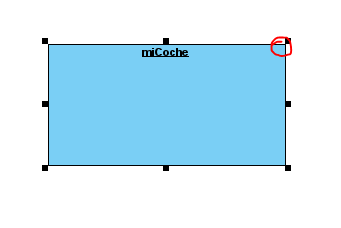
* Cuenta con tres bloques:
  + En el primero viene el nombre de la clase.
  + En el segundo se detallan todas sus propiedades, el símbolo “-“ quiere decir que se trata de una variable con acceso “private”, seguidamente viene el nombre de la variable y, después, el tipo de valor que va a almacenar.
  + En el tercero se especifícan los métodos de la clase. Con el símbolo “+” decimos que tienen un acceso “public”, después viene el nombre del método y, seguidamente, el tipo de valor que nos devuelve, si así fuera el caso y no se tratase de un “void”. Entre paréntesis vendrían los parámetros que necesita el método para poder invocarse, si los necesitase.

A continuación, procedo a la realización del objeto “Coche”, lo llamaré “miCoche”

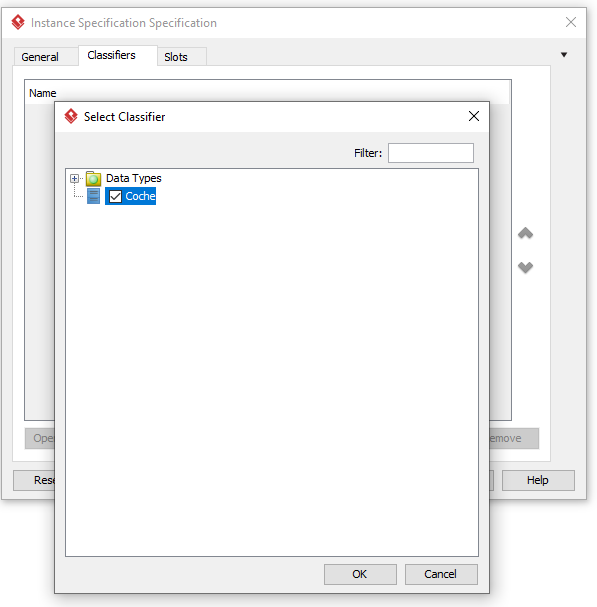




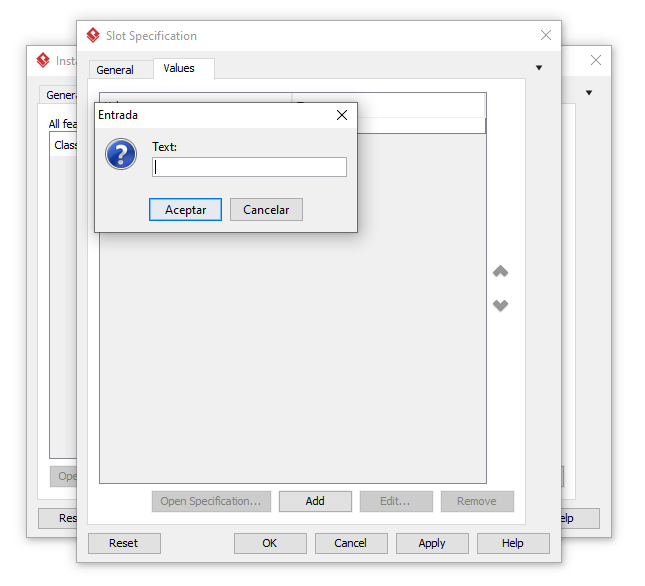
Le damos click al icono de la lupa que aparece en la esquina superior derecha.



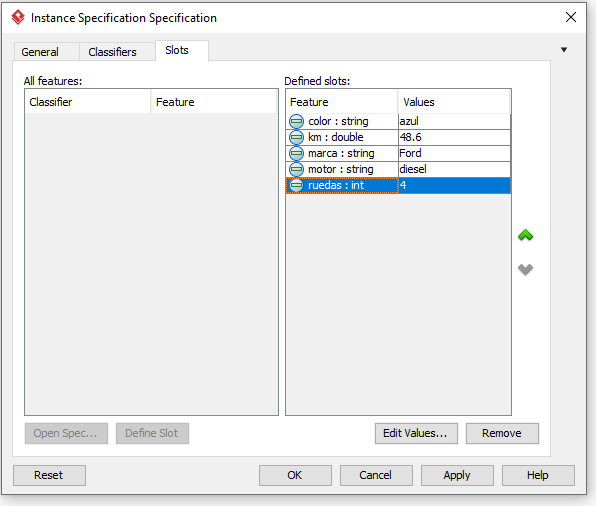
Seguidamente, damos a la pestaña “Classifiers” y seleccionamos nuestra clase.



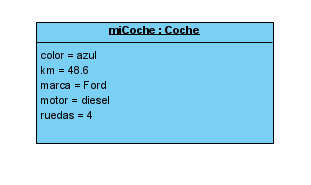
Después, damos a la pestaña “Slots”, seguidamente “Edit values”, “add”, “Text” e introducimos el valor que queremos, “aceptar” y “ok”.



Quedaría así nuestro panel de propiedades.



Y nuestro diagrama de objeto quedaría así.



Su estructura cuenta con dos bloques, y sería la siguiente:

* En la parte de arriba viene detallado el nombre del objeto seguido de la clase a la que está instanciando.
* En el segundo bloque se detallan las propiedades del objeto. Primero la variable a la que va a dar valor y, seguidamente del “=”, viene el valor que va a almacenar.

Instalarse una de las herramientas de Diseño de diagramas de clase que hay de libre distribución para Netbeans/Eclipse y a partir de las especificaciones de un determinado escenario propuesto por ti realizar lo siguiente:

1. Describir las especificaciones del escenario propuesto por ti.
2. Realiza el diagrama UML de clases basándote en las especificaciones del inciso a).
3. A partir del diagrama de clases de inciso b) genera el código. Te puedes ayudar de una de las herramientas como por ejemplo: “SDE de Visual Paradigm” que tiene una versión de prueba.

Deberás explicar el proceso de los tres incisos y adjuntar capturas de pantalla.

**a)**

Para la realización de este ejercicio voy a trabajar con el programa “Visual Paradigm”.

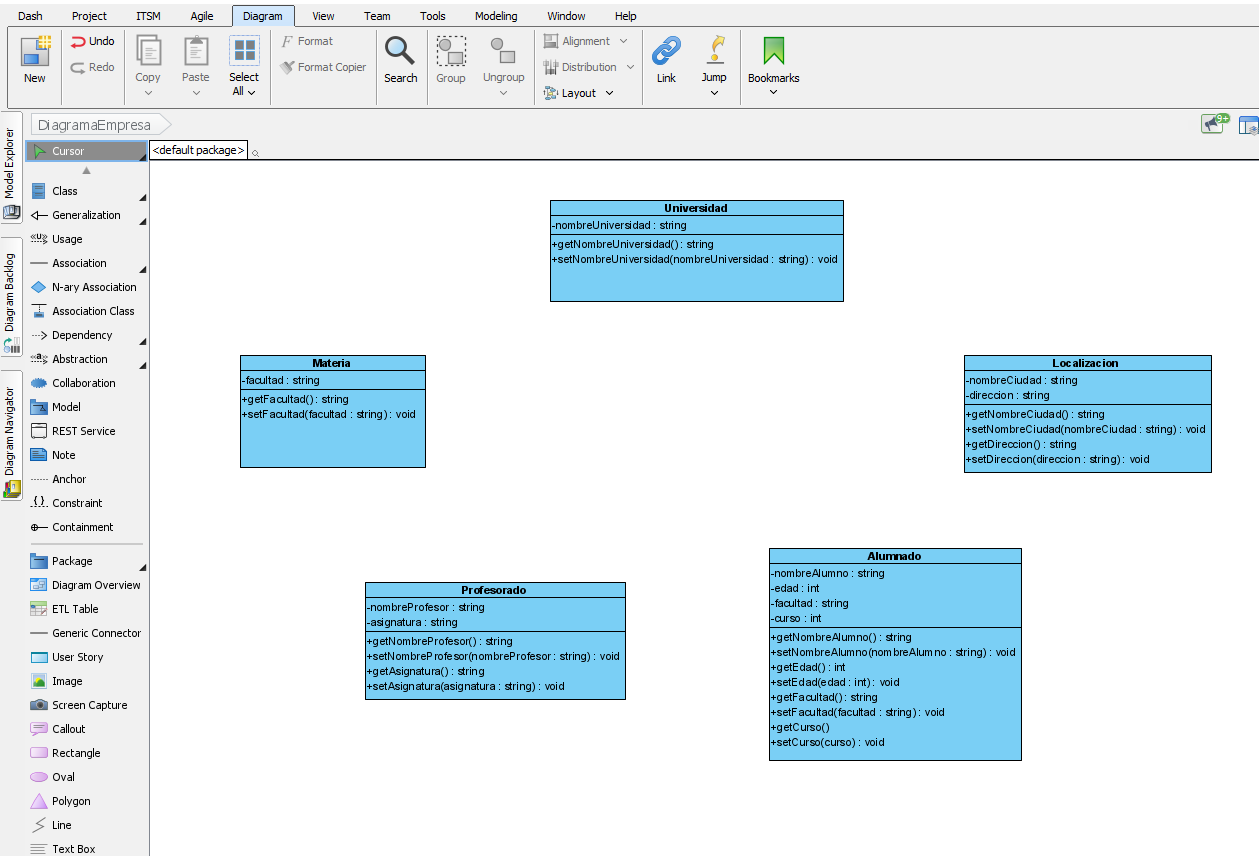
Como escenario para la elaboración del diagrama, he propuesto la posible estructura que existe de una universidad y su entorno. Consta de lo siguiente:

* Clase “Universidad”, que es la que almacena el nombre de la universidad.
* Clase “Materia”, que es la que almacena la licenciatura que se ejerce en esa universidad. Hereda de la clase “Universidad”.
* Clase “Localizacion”, que almacena la ciudad y la dirección de la universidad. Hereda de la clase “Universidad”.
* Clase “Profesorado”, encargada de almacenar el nombre del profesor y la asignatura que enseña. Hereda de la clase “Materia”.
* Clase “Alumnado”, se encarga de almacenar el nombre, la edad, la facultad en la que está inscrito y el año que está cursando. Hereda de la clase “Materia”.

Todos los atributos especificados en las clases cuentan con sus respectivos métodos “getters” y “setters”.

**b)**

La imagen del diagrama con todas las clases mencionadas y sus atributos y métodos quedaría así.



Para la elaboración de los paneles de cada una de las clases he seguido los mismos pasos que ya he descrito en el ejercicio nº2 de esta misma actividad. No lo detallo otra vez para evitar caer en la redundancia (quizás antes me excedí en la explicación).

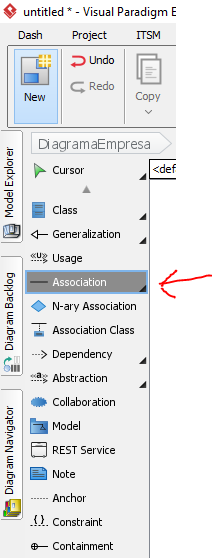
Ahora procedo a la muestra de relaciones entre las clases. Una relación es una conexión entre dos clases que incluimos en el diagrama cuando aparece algún tipo de relación entre ellas en el dominio del problema.

Las relaciones se caracterizan por su cardinalidad, que representa cuántos objetos de una clase se pueden involucrar en la relación, pueden ser de tres tipos:

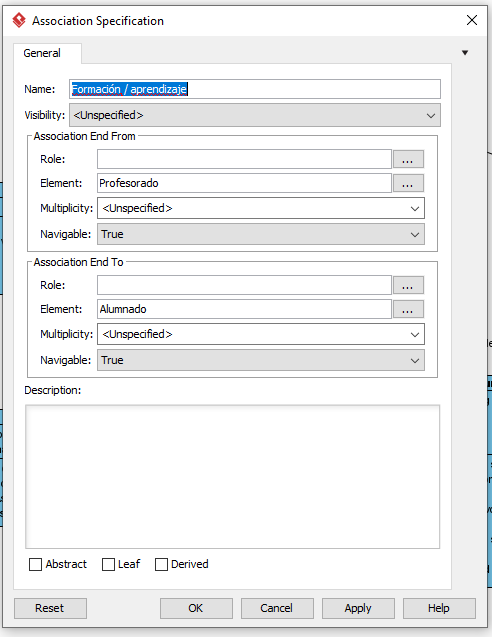
* De herencia.
* De composición.
* De agregación.

Podemos crear las relaciones de dos maneras:

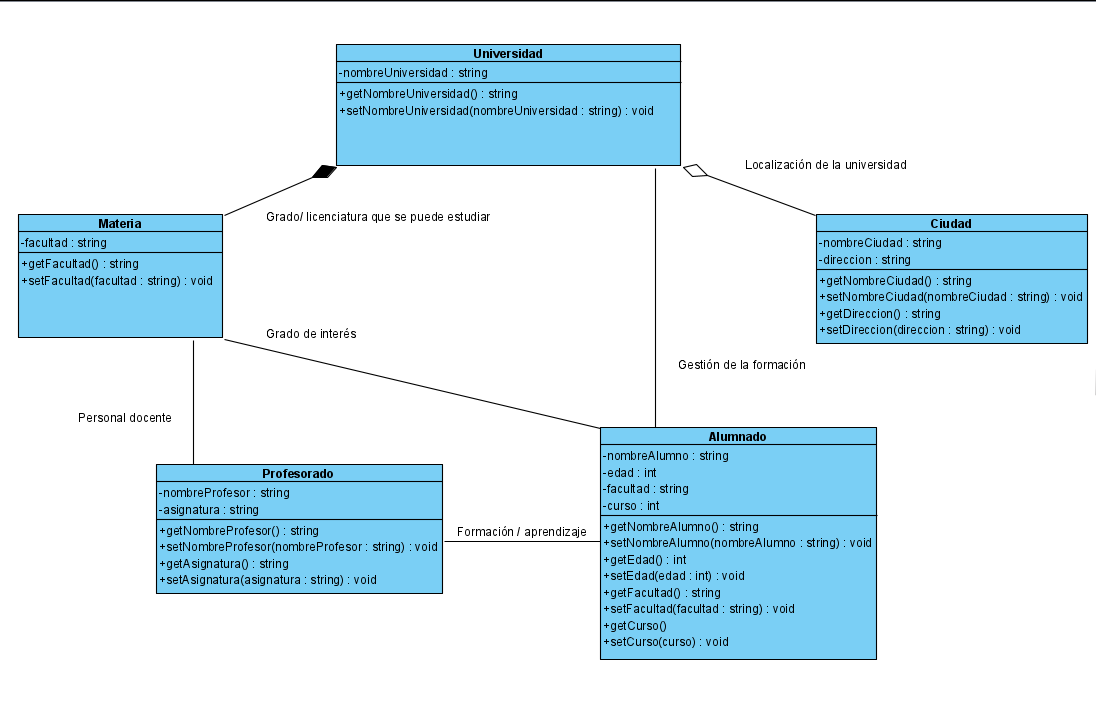
* Accediendo a la herramienta desde el panel izquierdo. Esta se llama “Association”.
* Haciendo clic derecho dentro del panel de trabajo, fuera de las tablas.



Una vez seleccionamos el tipo de relación que queremos hacer, hacemos clic en la clase que queremos y arrastramos la línea de relación hasta la clase que deseamos. Después le podemos poner un mensaje informativo.



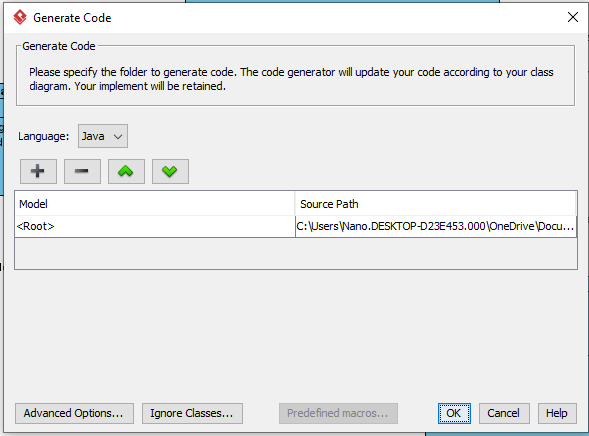
El resultado final del diagrama sería el siguiente:

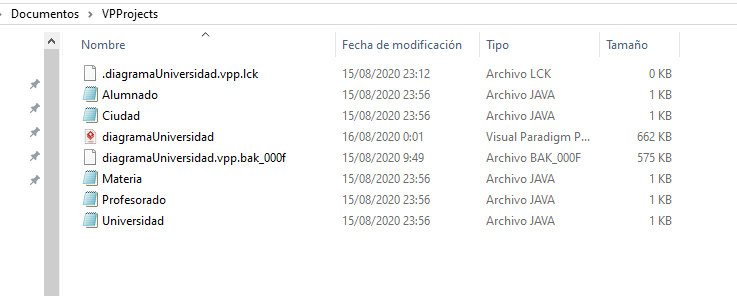


**c)**

Para generar el código desde el propio diagrama lo haremos de la siguiente forma (desde el mismo software “Visual Paradigm”):

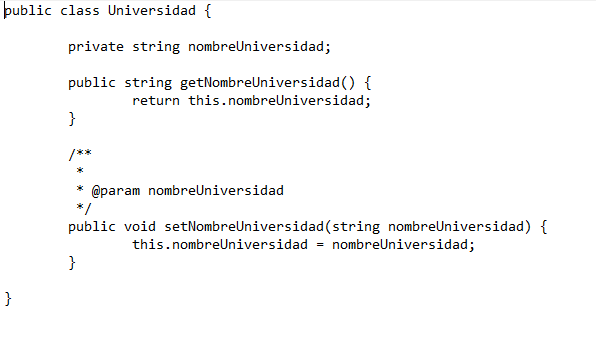
Pulsamos en la pestaña “Tools”, seguidamente le damos al apartado “Code”, accedemos a la opción “Generate Java Code” y pulsamos “Ok”. Para acceder a los archivos, iré a la ruta que me marca por defecto (o a la que yo le haya marcado).



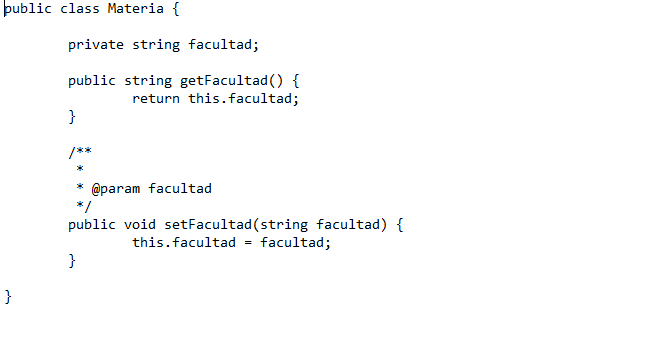


Así queda el código Java generado automáticamente desde el diagrama:

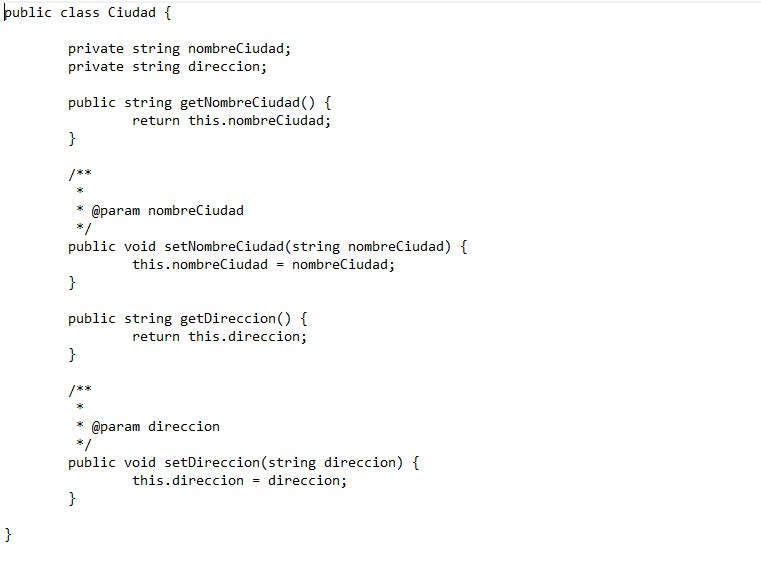
* **CÓDIGO CLASE “Universidad”**



* **CÓDIGO CLASE “Materia”**



* **CÓDIGO CLASE “Ciudad”**



* **CÓDIGO CLASE “Profesorado”**



* **CÓDIGO CLASE “Alumnado”**

