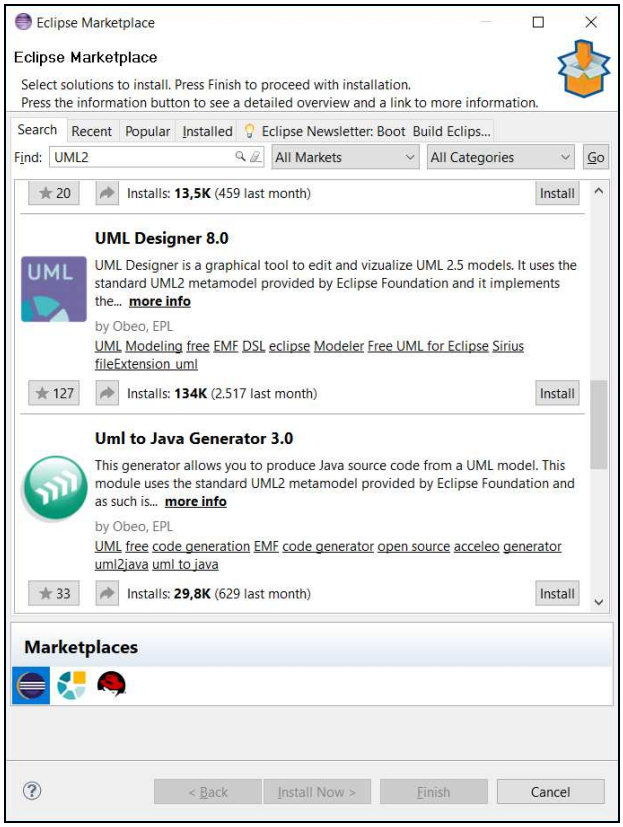
ED – UD05
PRÁCTICA 2: UML CON ECLIPSE

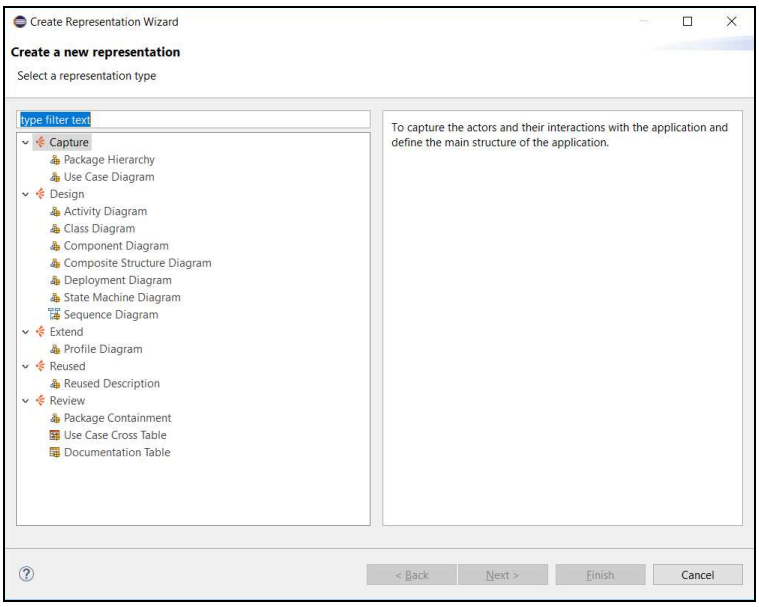
Para poder realizar diagramas UML utilizando *Eclipse* instalaremos el *plugin UML2*. Para ello, desde *Eclipse* abriremos el menú *Help* y seleccionaremos *Eclipse Marketplace*. Desde ahí seleccionaremos *UML2* en *Find* y pulsaremos el botón *Go* para que busque el plugin. Una vez localizados los *plugins* se mostrará una lista de ellos. Hemos de elegir el *plugin* que se corresponda con la versión de *Eclipse* con la que se trabaje. En nuestro caso elegiremos *UML Designer (Eclipse Kepler version)*. Pulsaremos *Install*, aceptaremos las condiciones y a continuación comenzará la instalación.



Una vez terminada la instalación hay que reiniciar *Eclipse*.

Para crear un proyecto abrimos el menú *File / New / UML Project*. Tecleamos el nombre, aceptamos las opciones por defecto y se creará automáticamente el proyecto.

Para crear un diagrama UML nos posicionamos sobre el proyecto, hacemos clic con el botón derecho del ratón y elegimos *Create Representation*. Desde ahí se selecciona el tipo de diagrama a crear.

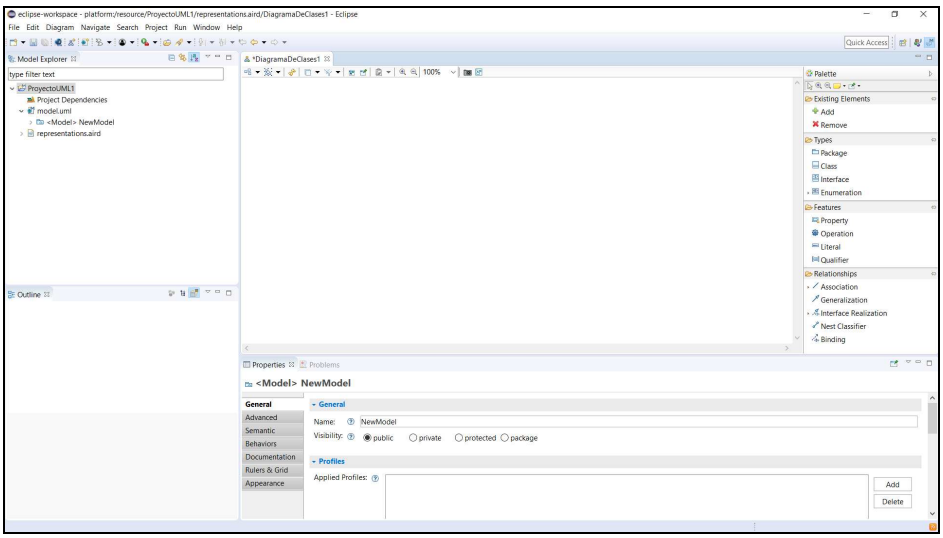


Dentro de los tipos de diagrama nos fijamos en los siguientes:

- Diagramas de comportamiento (*UML Behavioral Modeling*), para crear los diagramas de actividad, estado, casos de uso y secuencia.
- Diagramas de estructuras (*UML Structural Modeling*), entre los que se encuentran los de clases, componentes, estructura compuesta, implementación o despliegue, objetos y paquetes.

CREACIÓN DE DIAGRAMAS DE CLASE

Para crear un diagrama de clases se pulsa el botón derecho del ratón sobre el proyecto, se elige *Create Representation* y, dentro de *UML Structural Modeling*, se selecciona *Class Diagram*. Se pulsa *Siguiente*, se selecciona en qué modelo se creará el diagrama, se teclea un nombre y se abrirá la vista de diseño. En la pantalla aparecerán varias zonas: el explorador del modelo (vista del proyecto), la vista de diseño (donde se irán añadiendo los elementos del diagrama), la pestaña de propiedades (para cambiar las características de los elementos) y la paleta de elementos que se pueden añadir en el diagrama.

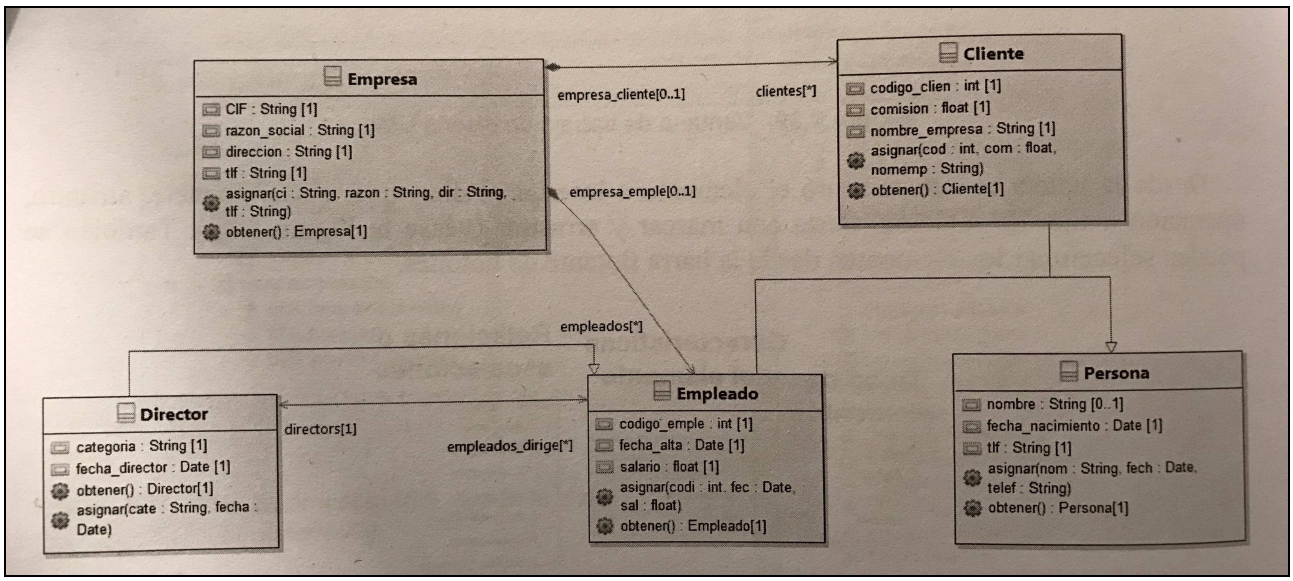


Desde la paleta se seleccionará el elemento a insertar al diagrama: clase, paquete, atributo, operación o tipo de relación. Basta con marcar y arrastrar. También se pueden seleccionar los elementos desde la barra flotante de botones.

EJEMPLO 1:

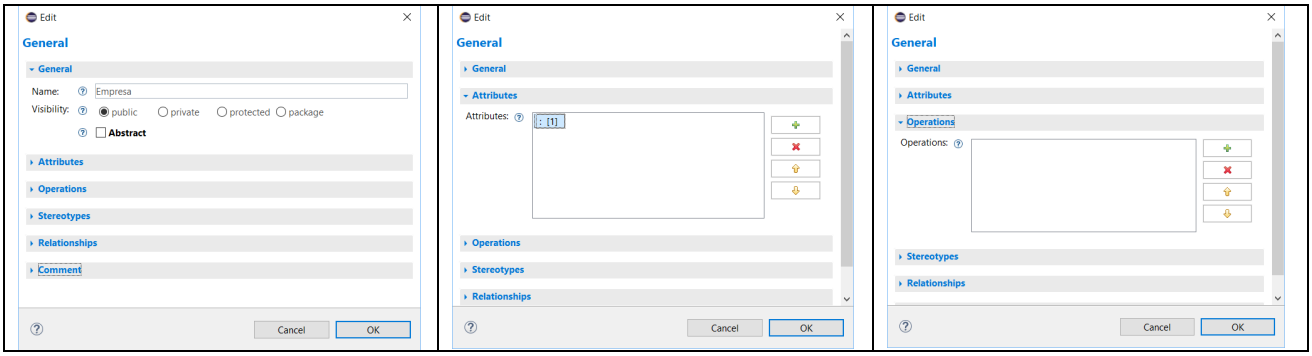
Se trata de realizar un diagrama de clases para representar las relaciones entre *empresa*, *empleados* y *clientes*. Utilizaremos asociaciones de composición y generalización en el diagrama. Los requisitos son los siguientes:

- La *empresa* se compone de *clientes* y de *empleados*. Utilizaremos para estas relaciones sendas asociaciones de composición.
- Datos a almacenar:
    - Datos de la *empresa*: CIF, razón social, dirección y teléfono.
    - Datos de los *clientes*: código de cliente, nombre, fecha de nacimiento, teléfono, empresa para la que trabaja y comisión.
    - Datos de los *empleados*: código de empleado, nombre, fecha de nacimiento, teléfono, fecha de alta en la empresa y salario.
- Como los atributos *nombre*, *fecha de nacimiento* y *teléfono* son comunes para *clientes* y *empleados*, se creará una clase *persona* para esos atributos, y las clases *cliente* y *empleado* heredarán de ella.
- Un *empleado* puede ser *director* de varios empleados. De este *director* se necesita saber también la categoría y la fecha de alta como director. El *director* heredará de *empleado* y además tendrá una asociación [1..*] con *empleado*.
- Para todas las clases crearemos dos métodos:
    - Uno para asignar datos a los atributos (como si fuese el constructor). Este método tendrá tantos parámetros como atributos.
    - Otro que se llame obtener que devolverá un objeto de la misma clase.



Al crear una clase, si se hace doble clic sobre la misma se muestra la ventana de creación de la clase. Desde ella podremos teclear el nombre, elegir la visibilidad (*public*, *private*, *protected* o *package*) y marcar el tipo de clase (*Is Abstract*, si va a ser una clase abstracta; *Is Leaf*, si se considera que no se puede convertir en una especialización en el futuro; *Is Active*, si la clase es una clase activa). También se puede asignar un caso de uso a la clase.

Si se desean añadir atributos y operaciones a la clase, podemos movernos por las pestañas *Attributes* u *Operations* y utilizar los botones correspondientes para añadir nuevos elementos, borrarlos o cambiarlos de posición.



A la hora de añadir los atributos se indicará el nombre, la visibilidad, las propiedades del atributo y el tipo. Para los ejemplos, se dejan las opciones que aparecen por defecto. En la siguiente tabla se muestran las propiedades de los atributos:

Propiedad	Valor por defecto	Descripción
Is Read Only	*False*	Si es *true*, el atributo es de solo lectura y no se puede cambiar su valor.
Is Static	*False*	Si es *true*, las instancias de este tipo comparten el mismo valor para este atributo.
Is Leaf	*False*	Si es *true*, no está diseñado para permitir que este atributo se redefina en los tipos derivados.
Is Derived	*False*	Si es *true*, este atributo se calcula a partir de otros atributos, es un atributo calculado.
Is Ordered	*False*	Si es *true*, la colección forma una lista secuencial y ordenada.
Is Unique	*False*	Si es *true*, no hay valores duplicados en la colección.

Al asignar un tipo de dato para los atributos, se abre la ventana para elegir el tipo de dato. Para los ejercicios que se van a realizar nos centraremos solo en algunos tipos de datos: *int*, *float*, *String* o *Date*. Así pues, se seleccionará de la lista *PrimitiveType*.

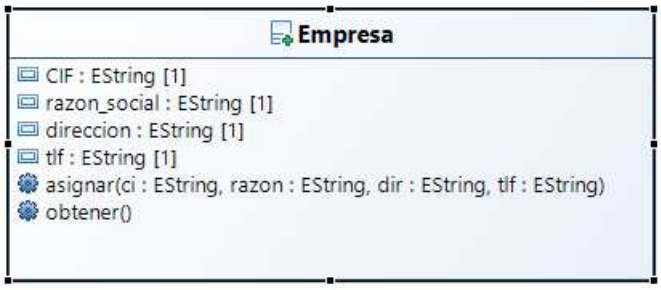
Si no aparece ningún tipo de dato hay que importarlos. Para ello se pulsa el botón derecho del ratón en la ventana de diseño y se elige *Import primitive types* del menú contextual. Seleccionaríamos *Java Primitive types*.

Al crear una operación se teclean el nombre y la visibilidad, y se marca el tipo (se dejan las opciones por defecto a no ser que se indique otra cosa en el enunciado). A continuación se añaden

los parámetros si los tiene. Para ello se pulsa la pestaña *Parameters* y el botón *Añadir* (+). En la nueva ventana que aparece se escribe el nombre del parámetro, se selecciona el tipo de dato y se marca el tipo de parámetro (*in* para indicar que es un parámetro de entrada, *inout* si es de entrada-salida, *out* si es de salida y *return* si la operación devuelve un valor).

También se pueden cambiar las propiedades de la clase desde la pestaña de propiedades. Si esta no se muestra se abre desde el menú *Window / Show view / Properties*. Desde aquí se podrán ver, añadir y modificar los atributos, las operaciones, las relaciones de la clase, la apariencia y la semántica. Cambiaremos las propiedades de la semántica para indicar la multiplicidad de la clase cuando tiene asociaciones.

Para elaborar el ejercicio, primero se crean las clases con los atributos y operaciones, siguiendo lo indicado en los párrafos anteriores. Una vez creadas las clases se crean las relaciones.



Entre *Empresa* y *Cliente* se crea una asociación de composición que va del compuesto (*Empresa*) al componente (*Cliente*). Se selecciona la asociación de la paleta y se pincha de la clase *Empresa* a la clase *Cliente*, desde el nombre de clase origen al nombre de clase destino.

Al crear las asociaciones se generan automáticamente sus nombres. Observa que llevan el nombre de la clase con la que se asocian seguida de una *s*. Observa también las multiplicidades de la asociación que aparecen entre corchetes.

La multiplicidad que se ha creado del lado de la clase *Empresa* es [0..1], y del lado de la clase *Cliente* [*]. Esto quiere decir que 1 empresa tiene muchos clientes. Esto se traduce en *Java* en que la clase *Empresa* tendrá un *HashSet* de objetos *Cliente* llamado *clientes*, y en que la clase *Cliente* tendrá un objeto de la clase *Empresa* llamado *empresas* (aunque *Eclipse* no crea esta segunda relación para asociaciones de este tipo).

Se hace lo mismo entre las clases *Empresa* y *Empleado*. Los nombres de las asociaciones se pueden cambiar para hacer más legible el diagrama. Basta con hacer doble clic sobre el nombre en el diagrama, o también en la asociación para que aparezca su ventana de propiedades, y abrir los elementos que componen la asociación. En nuestro caso el extremo de la clase *Empresa* se llama *empresa_cliente* y el de la clase *Cliente* se llama *clientes*.

Si una clase va a tener varias asociaciones con otra clase es conveniente cambiar los nombres de las asociaciones pues esos nombres se usan al generar el código *Java* a partir del modelo, y es necesario que no sean iguales para evitar errores de compilación.

Para crear las asociaciones de generalización entre *Persona-Cliente* y *Persona-Empleado*, se selecciona la asociación de generalización de la paleta de diseño y se marca desde la clase que la hereda a la clase heredada, es decir, de *Cliente* a *Persona* y de *Empleado* a *Persona*. En *Java*, la

generalización crea una herencia: así, la clase *Cliente* y la clase *Empleado* serán *extends* de la clase *Persona*.

Se hace lo mismo entre las clases *Director-Empelado*. La clase *Director* hereda de la clase *Empleado*, con lo que se selecciona la relación de generalización de la paleta y se arrastra de la clase *Director* a *Empleado*. En el caso de este tipo de asociaciones no se añaden nombres.

Finalmente se crea la asociación entre *Director* y *Empleado*. Es una asociación normal con una multiplicidad de [1..*], es decir, un *Director* dirige a muchos empleados, y un *Empleado* es dirigido por un director. Se selecciona el símbolo asociación de la paleta de diseño y se arrastra de *Director* a *Empleado*. En este caso da igual el orden. Por defecto, la asociación que se crea es muchos-muchos, es decir, [*] en ambos sentidos. Es necesario cambiar la multiplicidad de la asociación para dejarla en [1..*].

Para cambiar la multiplicidad de una asociación se selecciona la misma y se abre la pestaña de propiedades. Si no está visible, se abre el menú *Window / Show view / Properties*.

Al lado de la clase *Director* hay que poner la multiplicidad de [1], y al lado de la clase *Empleado* la de [*]. Dentro de la pestaña de propiedades destacamos los siguientes apartados:

- General. En este apartado se puede poner el nombre a la asociación y se pueden ver las clases que asocia.
- Semantic. Aquí se indica la semántica de la asociación, es decir, el significado y el sentido. En este apartado es donde se cambia la multiplicidad. Está formado por la parte donde se indican las propiedades de la asociación, *<Association>*, y las partes donde se indican las propiedades, *<Property>* del papel que juegan las clases que forman la asociación (en nuestro caso, *Empleado* y *Director*).

Para cambiar la multiplicidad se abren las propiedades de la asociación y, en la propiedad *Lower* se indica el valor mínimo de la participación de la clase en la asociación, mientras que en *Upper* se indica el valor máximo. Así, en el ejercicio, la clase *Empleado* en esta asociación tiene de valor mínimo [0] (también se podía haber considerado 1) y de valor máximo [*], ya que un director dirige a 0 ó varios empleados. Y la clase *Director* tiene una multiplicidad de valor mínimo [1] y de valor máximo [1] ya que es un director el que dirige.

Si el modelo se convierte a *Java* con *Eclipse*, la clase *Director* tendrá un *HashSet* de objetos *Empleado* llamado *empleados_dirige*, y la clase *Empleado* tendrá un objeto de la clase *Director* llamado *directors*.