Tema 6: Diagramas de clase en UML

¿Qué aprenderás?

- Aprender la notación de los diagramas de clase en UML.
- Elaborar diagramas de clases en UML.
- Indentificar clases, atributos, métodos y visibilidad en la programación para diseñar diagramas de clases.
- Identificar distintos tipos de relaciones en los diagramas de clases.
- Comprender qué es la ingeniería inversa y aplicarla en los diagramas de clases.

¿Sabías que…?

- La ingeniería del software es una de las ramas de la ciencias que estudia la creación de software confiable y de calidad.
- Es posible generar diagramas UML a partir del código en algunos IDE.

6.1. Introducción a UML

6.1.1. Antes de UML

Cuando se queria representar la estructura o el comportamiento de un proyecto de software, cada programador hacía servir la notación que veia conveniente para explicar el código a sus compañeros. Esto podía ser un problema si el programador cambiaba de proyecto y no era posible preguntarle acerca de sus esquemas o diseños.

Cuando se diseña un proyecto de software es necesario que todo el equipo comprenda todo lo que se ha de hacer y que se corresponda con lo que necesita el cliente.

Sin un estandar, es más complicado transmitirlo de forma correcta a los desarrolladores. Puede dar pie a diferentes interpretaciones de la estructura o funcionamiento. Si cada programador entiende una cosa diferente, la solución final probablemente no será lo que pide el cliente.

En conclusión, antes de que existiera UML la comunicación entre los desarrolladores muchas veces podía llegar a ser muy confusa y difícil de comprender.

Debido a este contexto y con el gran tamaño y complejidad de muchos de los proyectos de software se hace necesario establecer un lenguaje uniforme de modelado que permita que esta información llegue a todos los desarrolladores y analistas del proyecto.

Después de UML
Entonces surge UML (Unifiied Modeling Language). Se trata de una herramienta que permite a los creadores de sistemas generar diseños del comportamiento y de la estructura del software.

- Permite representar los diseños en una forma fácil de comprender.
- Fácil de comunicar a otras personas.
- Su objetivo es que sea fácil de entender y rápido de crear.

Al ser un lenguaje unificado con unos estándares que todos los programadores entienden, podemos realizar diferentes tipos de diagramas que nos hacen más fácil la tarea de representar muchos apartados del software.

Existen muchos tipos de diagramas UML, aunque en la práctica no siempre será necesario hacer servir todos los diagramas para un proyecto. Normalmente y en el nivel que nos encontramos los que más se hacen servir son los diagramas de clase, de secuencia, de casos de uso y de actividad.

6.1.2. Ventajas de UML

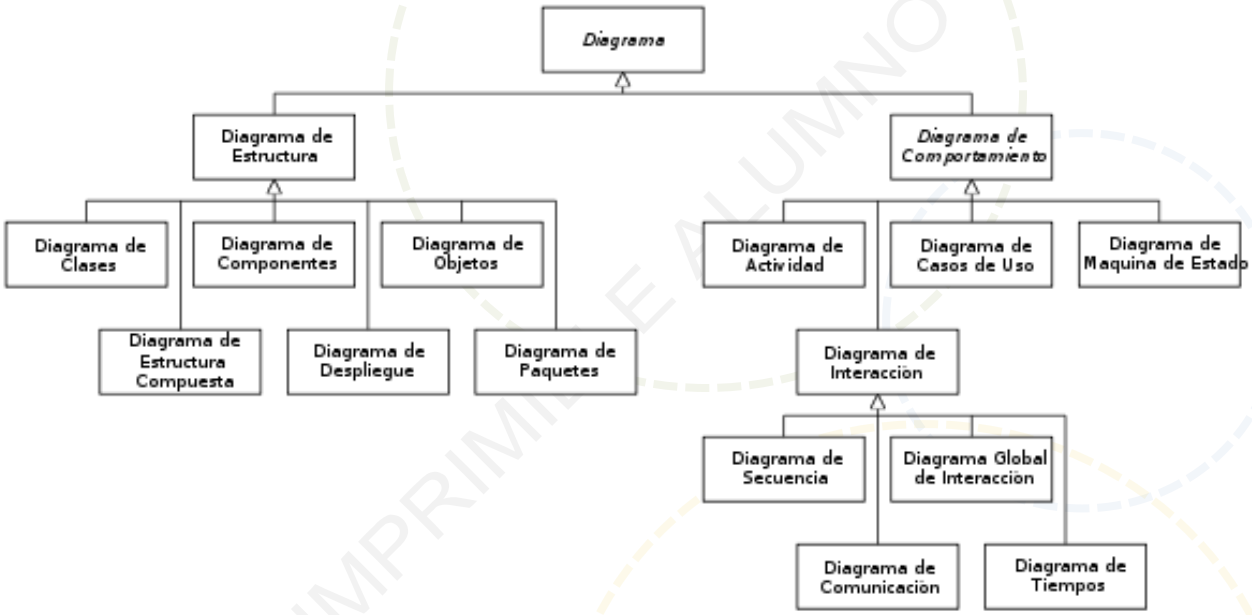
- Permite tener una visión global o específica del proyecto.
- Es un estandard respaldado por OMG (Object Management Group).
- Permite realizar modelos o diagramas de una aplicación.
- Cuenta con una notación gráfica fácil de interpretar.
- Se pueden complementar los diagramas con explicaciones escritas.
- Se pueden hacer servir los diagramas que se consideren oportunos segun el proyecto.
- Ayudan a que el proyecto sea reutilizable y mantenible.

Existen muchas herramientas online gratuitas que nos permiten representar diferentes diagramas, aunque con un papel y un bolígrafo nos resultará muy fácil representar estos diagramas ya que no son complejos de realizar.

¿Qué tipos de diagramas podemos crear?

Hay muchos tipos de diagramas que podemos representar, no estudiaremos todos. Principalmente se dividen en diagramas que representan la estructura diagrama que representan el comportamiento.

- **Estructura o visión estática del proyecto:** Para describir el sistema a través de los elementos que lo forman.
- **Comportamiento o visión dinámica del proyecto:** Para describir el sistema explicando lo que ha de pasar.



¿Por qué son necesarios tantos tipos de diagramas?

Porque en general un sistema se puede interpretar desde distintos puntos de vista. Podemos diseñarlo de tal manera que veamos cómo se relacionan sus componentes entre sí o bien para ver como actuan las diferentes funcionalidades entre ellas.

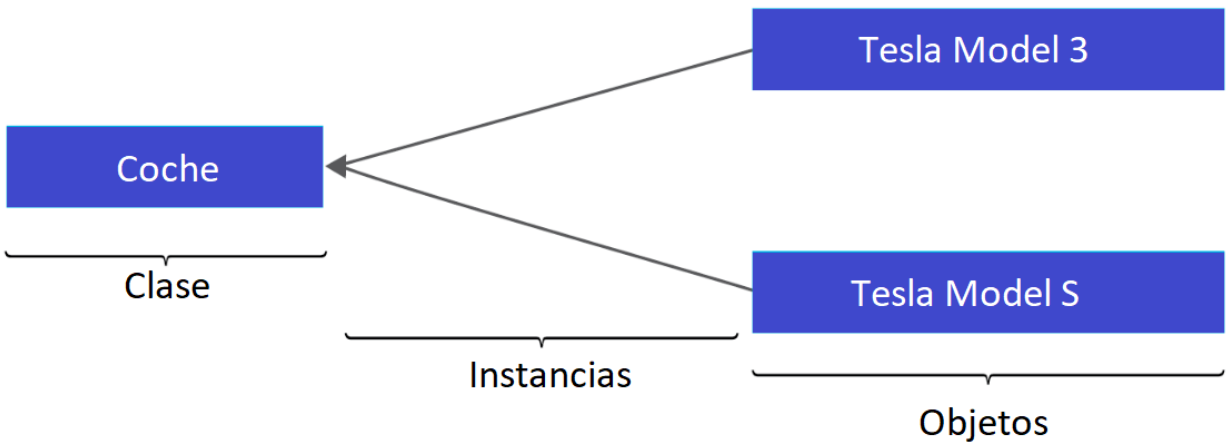
Por ejemplo, en el contexto de un software de simulación de conducción, tendremos que representar un coche a nivel estructural. Este tendrá un volante, ruedas motor, frenos, etc. En cambio necesitaremos otro enfoque para ver el comportamiento del coche como por ejemplo el proceso de encendido del motor (introducir llave, girar llave, pisar embrague, girar llave de nuevo, etcétera), el proceso de aceleración (primero poner una marcha, acelerar…), etc.

6.2. Objetos

Antes de empezar con los diagramas de clases de UML es fundamental entender unos conceptos básicos sobre el paradigma de la programación orientado a objetos:

- Una **clase** es la unidad básica que encapsula toda la información de un objeto a través de la cuál podemos modelar el entorno.
- Una **clase** describe un conjunto de objetos que comparten los mismos atributos y las operaciones que representan las acciones que puede hacer.
- Los **atributos** son características propias de cada objeto: nombre, edad, lenguaje de programación favorito...
- Los **métodos** son las acciones que podemos llegar a realizar: caminar, hablar, aprender, programar…

- Un objeto es una instanciación de una clase.
- La **instanciación** es la acción de crear una instancia de una clase.



Por ejemplo (típico) del coche. Un coche tiene como atributos específicos, su marca, la matrícula, el color, la potencia, etc. Además el coche puede realizar acciones como acelerar, frenar, girar a la derecha, girar a la izquierda, etc.

- La **creación de una instancia** de una clase se refiere a hacer una llamada al método constructor de una clase en tiempo de ejecución de un software.
- Un **objeto** se puede definir, como una unidad de memoria relacionada que, en tiempo de ejecución, lleva a cabo acciones dentro un software. Es algo que se puede distinguir y que tiene una existencia propia, ya sea de forma conceptual o de forma física.
- Un **objeto** puede pertenecer a más de una clase, pero sólo de una de ellas se pueden crear objetos directamente: el resto de clases representan sólo papeles que puede desempeñar el objeto.
- **Los objetos tienen identidad propia**, lo que significa que dos objetos creados por separado siempre se consideran diferentes, aunque tengan los mismos valores en sus características estructurales.
- Dos instancias con los mismos valores en las características estructurales se consideran como una misma instancia.

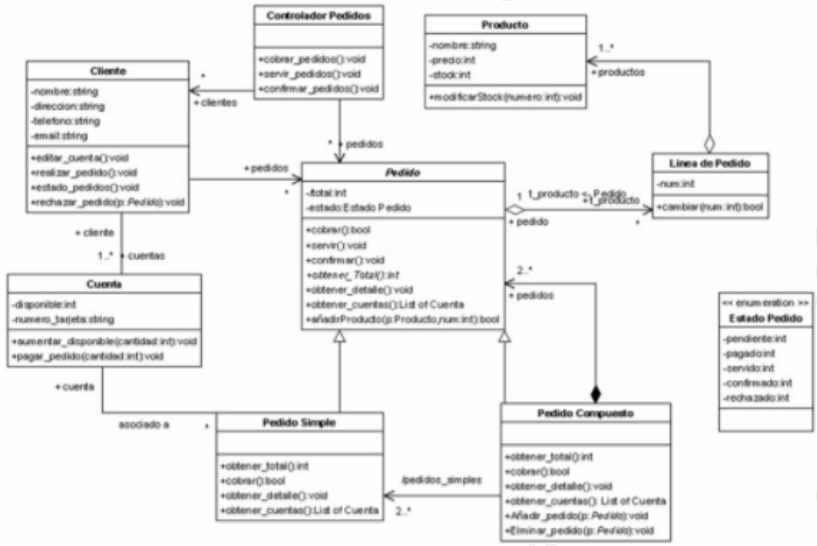
6.3. Diagramas de clases en UML

6.3.1. Diagramas de clases

¿Qué es un diagrama de clases?

Es un diagrama que relaciona las clases y sus diferentes operaciones entre sí.

Tienen esta pinta:



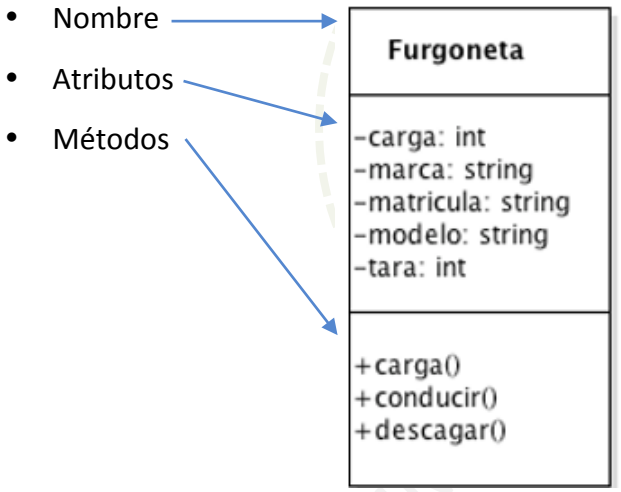
Esta imagen es para que veas que pinta tiene un diagrama de clases.

6.3.2. Notación de los diagramas de clases

6.3.2.1. Clases

Una clase se dibuja como un rectángulo compuesto de tres partes:

- En la primera se indica el nombre de la clase.
- En la segunda indican a los atributos que son las características de la clase. Habrá que indicar su visibilidad.
- En la tercera se indican a los métodos o operaciones con su declaración y visibilidad.



6.3.2.2. Visibilidad

La visibilidad se asigna a los atributos o a las operaciones. Definen el ámbito desde el que podrán ser utilizados estos elementos.

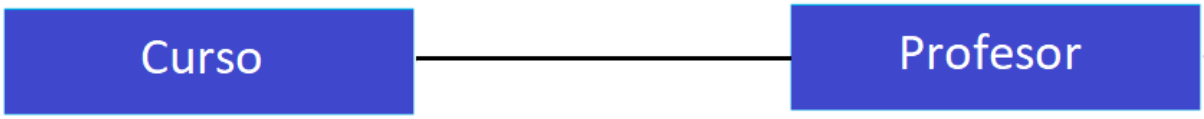
Los atributos y los métodos pueden tener estas visibilidades:

- **+ público** (accessible por todas clases)
- **- privado** (accessible sólo desde la propia clase)
- **# protected** (accessible por la subclases y la propia clase)

6.3.2.3. **Relaciones entre clases**

Asociación

Define las conexiones entre dos o más objetos, lo que permite asociar objetos que instancian clases que colaboran entre sí.



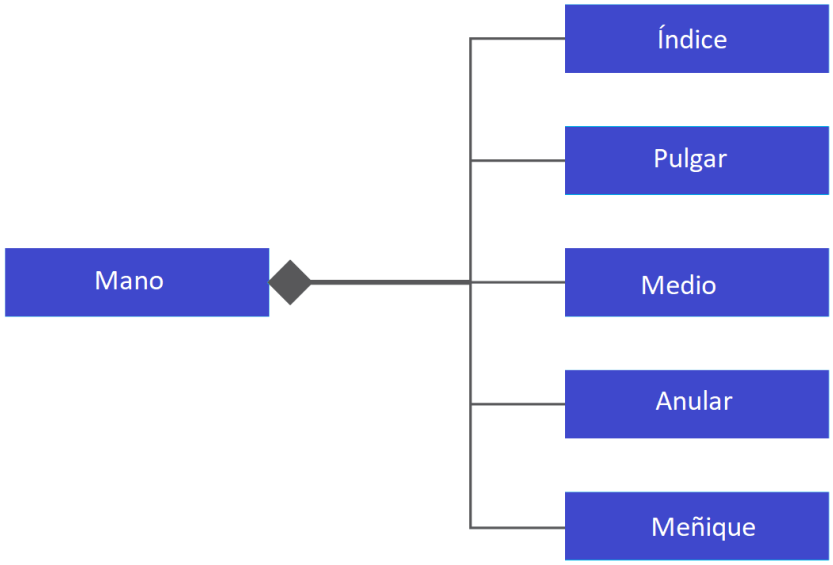
La asociación es la más básica de las relaciones, no tiene un tipo definido y puede ser tanto una composición como una agregación.

Se representan mediante una flecha simple que también puede tener una cardinalidad.

- **Asociación de Composición**

La composición define los componentes de los que se compone otra clase (**asociación fuerte**).

Define además que la clase que contiene la composición no tiene sentido de existencia si la(s) agregada(s) desaparece(n).

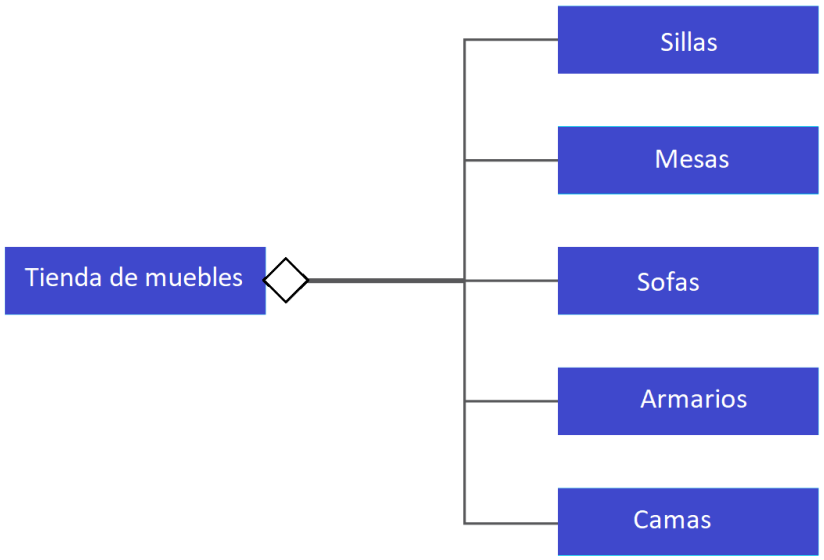


Este diagrama considera que la mano no estaría completa si no estuviera compuesta por los cinco dedos.

- **Asociación de Agregación**

Tipo de asociación que indica que una clase es parte de otra clase **(composición débil)**.

Los componentes pueden ser compartidos por varios compuestos (de la misma asociación de agregación o de varias asociaciones de agregación distintas).

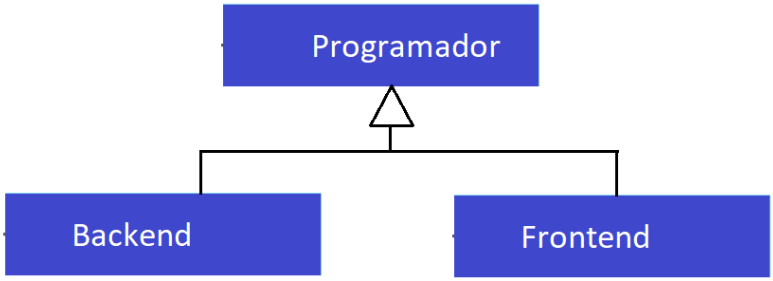


Este diagrama considera que aunque una tienda de muebles se quedara sin existencias de cualquiera de los componentes, podría seguir existiendo como tienda.

La destrucción del compuesto no conlleva la destrucción de los componentes. Habitualmente se da con mayor frecuencia que la composición.

- **Herencia**

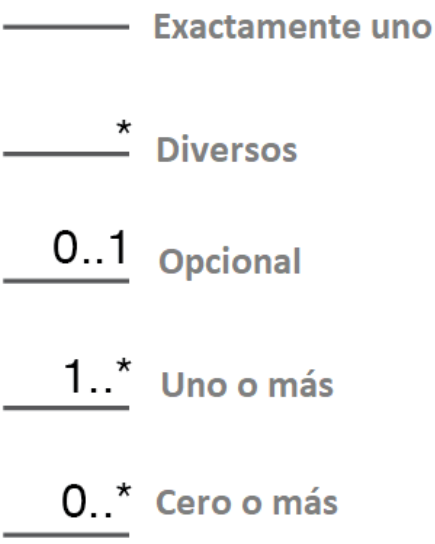
La herencia es un modo de representar clases y subclases, es decir, **clases más específicas de una general**, se representan mediante una flecha con una punta triangular vacía.



- **Cardinalidad**

La cardinalidad es un número que representa el número de elementos de cada clase en cada relación.

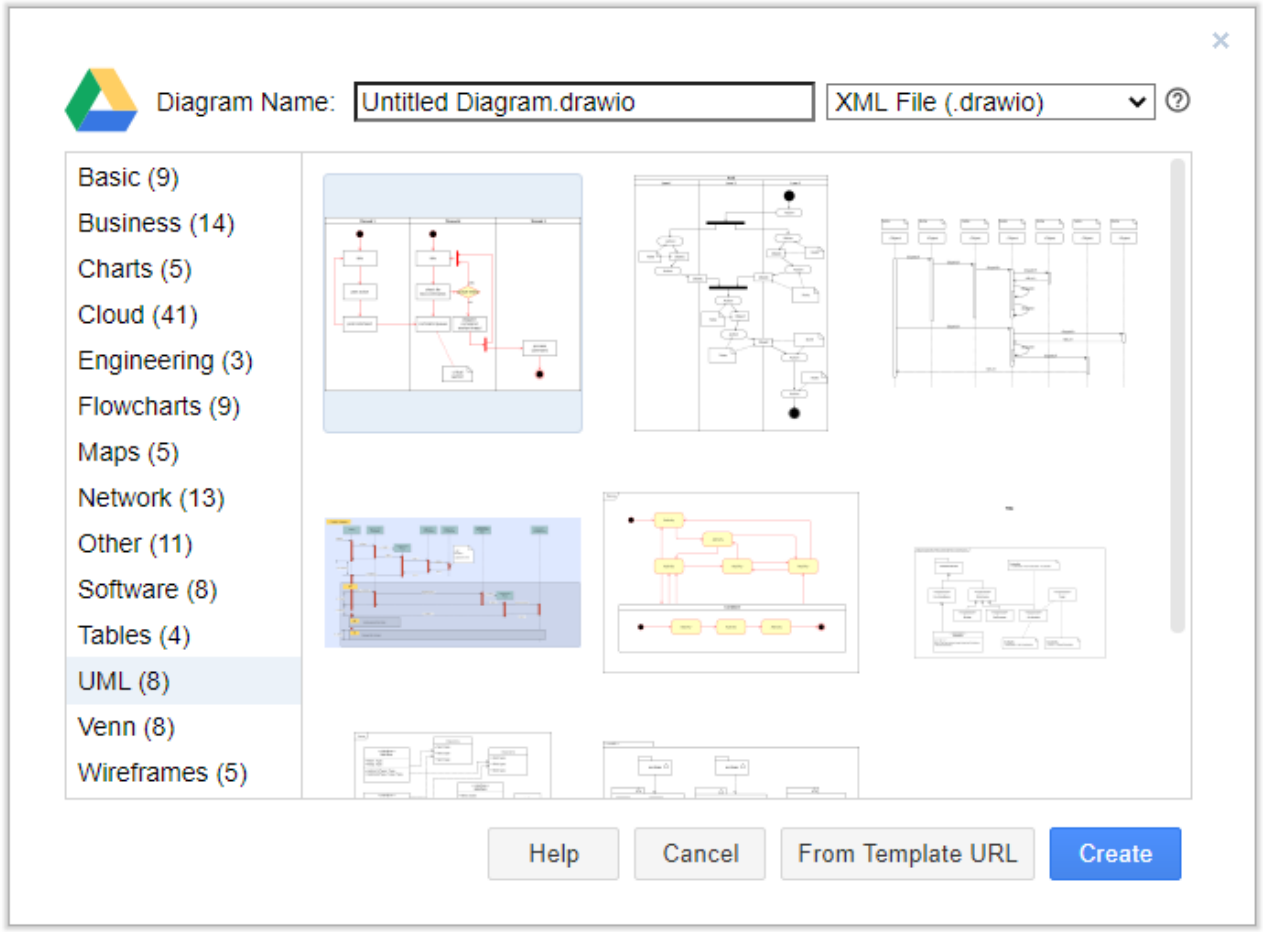
El carácter * es un comodín para definir un número indeterminados de clases.

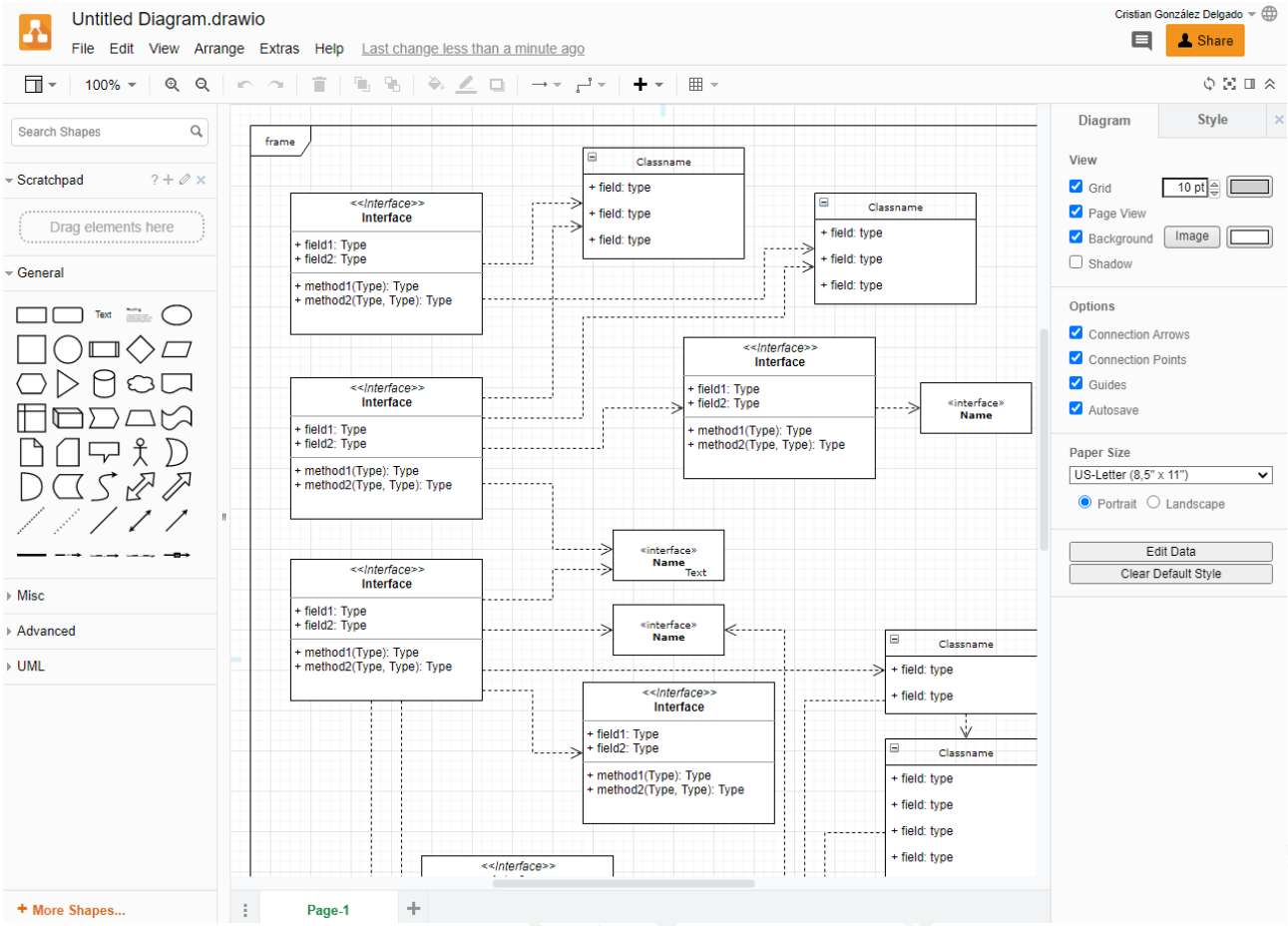


- **Herramientas para construir diagramas UML**

Aunque los diagramas UML se pueden llegar a hacer en un papel y con un bolígrafo. Existen muchas herramientas online con las que poder crear todos los tipos de diagramas UML.

Por ejemplo, **Draw.io** es una herramienta web que te permite crear diferentes tipos de diagramas UML y guardarlos en la nube.





Existen otras muchas herramientas con las que crear diagramas UML.

- dia
- Lucidchart
- Creately
- UMLetino
- Diagramo

6.4. La ingeniería inversa en los diagramas de clase

La ingeniería inversa, en el contexto de los diagramas de clases es un proceso que permite a través de un código fuente ya realizado, construir el diagrama con la finalidad de respresentarlo.

Algunos IDE ofrecen una herramienta para llevar a cabo esta tarea. A partir de un proyecto ya codificado, será posible obtener el diagrama que lo representa.

Recursos y enlaces

- Web oficial de UML (ingles) (inglés)

- Web de diagramas UML (español)

- Herramienta para crear diagramas UML Draw.io

- Herramienta para crear diagramas UML dia

Conceptos clave

- **UML:** Unified Modeling Language.
- **Objeto**: entidades identificables del mundo real. En el modelo UML, los objetos son instancias de una clase.
- **Instancia:** un elemento del conjunto de objetos de la clase.
- **Métodos:** conjuntos de instrucciones que toman valores en la entrada y modifican los valores de los atributos o producen un resultado.
- **Atributos**: caracterísiticas individuales que diferencian un objeto de otro y determinan su apariencia, estado u otras cualidades.

Test de autoevaluación

1. En el diseño de clases en UML...
   a) Las clases representan a nuestros objetos, los atributos definen las propiedades y características del objeto y los métodos especifican las acciones que podemos realizar con dicho objeto.
   b) Las clases representan a nuestros objetos, los métodos definen las propiedades y características del objeto y los atributos especifican las acciones que podemos realizar con dicho objeto.
   c) Los métodos representan a nuestros objetos, los atributos definen las propiedades y características del objeto y las clases especifican las acciones que podemos realizar con dicho objeto.
   d) Los atributos representan a nuestros objetos, las clases definen las propiedades y características del objeto y los métodos especifican las acciones que podemos realizar con dicho objeto.

2. Los diagramas de clases en UML...
   a) Tienen la particularidad de que no muestran las acciones que se realizan sobre cada objeto.
   b) Tienen la particularidad de que muestran las acciones que se realizan sobre cada objeto.
   c) Tienen la particularidad de que muestran la secuencia de acciones de los objetos.
   d) Tienen la particularidad de que no muestran los métodos ni los atributos de los objetos.

3. ¿Qué es el UML?
   a) Un programa para la creación de modelos de otros programas.
   b) Un conjunto de estándares para la representación gráfica de un programa.
   c) Un conjunto de estándares que definen la metodología para definir diagramas de clases.
   d) Un lenguaje que nos permite modelar de forma unificada las pruebas a realizar sobre un programa.

Ponlo en práctica

Actividad 1

Realizar el diagrama de clases a partir del siguiente enunciado

Se quiere desarrollar una aplicación informática que permita llevar a cabo la gestión de varios proyectos informáticos. Habrá que tener almacenada toda la información referente a cada proyecto que se quiera manejar. Habrá dos tipos de personas que podrán estar involucradas en los proyectos, ya sea para su gestión o para la ejecución de las actividades. Estos dos tipos de personas son el jefe de proyecto y el programador. De cada persona se querrá almacenar su nombre y su DNI.

Un proyecto será liderado por un jefe de proyecto. Pero un jefe de proyecto sólo podrá gestionar tres proyectos. Un proyecto tiene actividades; cada proyecto podrá tener desde una a muchas (de forma indefinida) actividades, pero una actividad sólo podrá pertenecer a un proyecto.

Los programadores desarrollarán las actividades. Cada programador tendrá vinculada una actividad como mínimo, sin un máximo establecido. Igualmente, las actividades podrán ser desarrolladas por un mínimo de un programador, sin haber un máximo establecido.

SOLUCIONARIOS

Test de autoevaluación

1 En el diseño de clases en UML...

a) **Las clases representan a nuestros objetos, los atributos definen las propiedades y características del objeto y los métodos especifican las acciones que podemos realizar con dicho objeto.**

b) Las clases representan a nuestros objetos, los métodos definen las propiedades y características del objeto y los atributos especifican las acciones que podemos realizar con dicho objeto.

c) Los métodos representan a nuestros objetos, los atributos definen las propiedades y características del objeto y las clases especifican las acciones que podemos realizar con dicho objeto.

d) Los atributos representan a nuestros objetos, las clases definen las propiedades y características del objeto y los métodos especifican las acciones que podemos realizar con dicho objeto.

2- Los diagramas de clases en UML...

a) **Tienen la particularidad de que no muestran las acciones que se realizan sobre cada objeto.**

b) Tienen la particularidad de que muestran las acciones que se realizan sobre cada objeto.

c) Tienen la particularidad de que muestran la secuencia de acciones de los objetos.

d) Tienen la particularidad de que no muestran los métodos ni los atributos de los objetos.

3. ¿Qué es el UML?

a) Un programa para la creación de modelos de otros programas.

b) **Un conjunto de estándares para la representación gráfica de un programa.**

c) Un conjunto de estándares que definen la metodología para definir diagramas de clases.

d) Un lenguaje que nos permite modelar de forma unificada las pruebas a realizar sobre un programa.

Ponlo en práctica

Actividad 1

Realizar el diagrama de clases a partir del siguiente enunciado

Se quiere desarrollar una aplicación informática que permita llevar a cabo la gestión de varios proyectos informáticos. Habrá que tener almacenada toda la información referente a cada proyecto que se quiera manejar. Habrá dos tipos de personas que podrán estar involucradas en los proyectos, ya sea para su gestión o para la ejecución de las actividades. Estos dos tipos de personas son el jefe de proyecto y el programador. De cada persona se querrá almacenar su nombre y su DNI.

Un proyecto será liderado por un jefe de proyecto. Pero un jefe de proyecto sólo podrá gestionar tres proyectos. Un proyecto tiene actividades; cada proyecto podrá tener desde una a muchas (de forma indefinida) actividades, pero una actividad sólo podrá pertenecer a un proyecto.

Los programadores desarrollarán las actividades. Cada programador tendrá vinculada una actividad como mínimo, sin un máximo establecido. Igualmente, las actividades podrán ser desarrolladas por un mínimo de un programador, sin haber un máximo establecido.

Solución: