Unidad 6

Elaboración de diagramas de clases

Entornos de desarrollo

Índice

Introducción

En este tema estudiaremos los diversos conceptos que forman parte de los diagramas UML, los cuales permiten la construcción de los programas orientados a objetos, POO.

La OO, orientación a objetos, pretende eliminar la conciencia anterior de buscar una funcionalidad total, y en su lugar basarse en el empleo de la representación de objetos con los que designar las acciones a realizar y las características de cada una de ellas.

Este tipo de programas permite el empleo de diversas funciones mediante el uso de una sola clase de objetos o a partir de la asociación de diversos objetos, permitiendo la creación de una acción más compleja.

De este modo el empleo de este tipo de programas permite una programación con una clara visualización de los distintos elementos que se emplearán para realizar la acción o que forman parte del proceso.

Al finalizar esta unidad

- Conoceremos los distintos elementos que componen los diagramas de clases.
- Estudiaremos las distintas asociaciones que se pueden realizar entre los elementos del diagrama.
- Describiremos los distintos tipos de relaciones dentro del diagrama que no sean asociaciones.
- Definiremos los distintos conceptos que permiten el empleo de atributos y métodos de forma correcta.

6.1. Notación de los diagramas de clases

El UML, Unified Modeling Language o lenguaje unificado de modelado, es la notación empleada para la representación de diferentes diagramas de clases u objetos, ya que este es un conjunto de herramientas empleadas para la construcción y elaboración de los programas orientados a objetos.

El uso de UML como estándar se establece con el fin de evitar grandes variaciones en los modos de construcción, permitiendo tener un único sistema con el que todos puedan trabajar y que todos los dispositivos puedan interpretar.

6.1.1. Clases

Clases es la denominación que reciben los diferentes tipos de objetos. La representación de las clases se estructura en un conjunto de tres celdas, generalmente en columna, donde se pueden apreciar tres elementos:

- En la primera celda podremos encontrar el nombre de la clase.
- En la segunda celda se encontrarán los atributos de la clase, el número de atributos no es fijo.
- En la tercera celda se incluyen las funciones de la clase, es decir, las acciones que realiza.

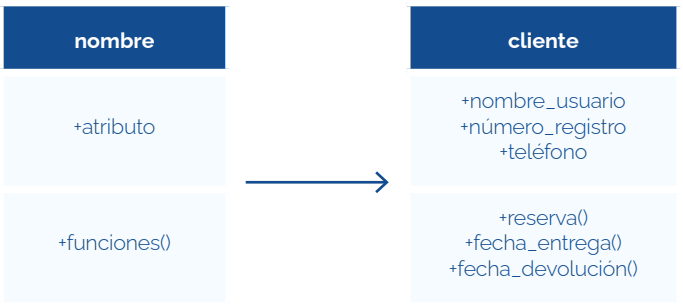


Imagen 1. Ejemplo de clase

La clase es un modelo que nos permitirá identificar elementos de la vida real.

Es posible en el caso de las palabras compuestas escribirlas empleando la notación camelCase, donde en lugar de separase con un "_" ambas palabras se separarían con la mayúscula de la segunda palabra, la primera permanece en minúscula, ejemplo, nombreUsuario, númeroRegistro, etc.

6.1.2. Atributos

Los atributos emplean un tipo de UML para definir su valor, los posibles formatos de tipo empleados son:

- **String o cadena de caracteres:** Empleo de la escritura para la designación del atributo.
- **Integer/int o número entero:** Empleo de un número, positivo o negativo, como valor, pero siempre entero.
- **Float o número con coma flotante**: Empleo de un número con decimales como valor.
- **Boolean o Booleano:** Empleo de una designación bipartita de verdadero o falso.

Los atributos empleados por el ejemplo anterior usarían los distintos elementos según su tipo de valor:

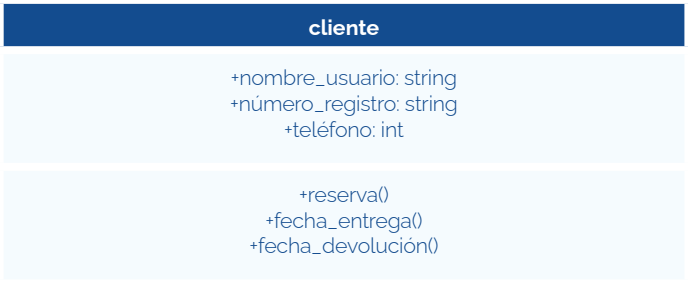


Imagen 2. Ejemplo de atributos

Es posible incluir valores predeterminados a los atributos, ya sea porque sean fijos o porque sean los predeterminados antes de su cambio.

- En el caso de los valores fijos se emplearía un igual y el valor entre comillas, ejemplo: estado_usuario: string = "Registrado"
- En el caso de los valores predeterminados se emplearía solamente un igual con el valor sin ninguna marca, ejemplo, en este caso con el valor "nulo" hasta ser empleado por primera vez: últimoRegistro: int = null
- Es posible también mediante corchetes estipular valores predeterminados a elegir, ejemplo: estadoUsuario: string = conectado {estadoUsuario=conectado o desconectado}.

6.1.3. Notas adjuntas

Es posible insertar información extra mediante el empleo de notas adjuntas, las cuales pueden llegar a contener incluso imágenes, su escritura no afecta al objeto, tan solo sirven como información añadida a la hora de registrar el código, generalmente a modo de recordatorio.

Imagen 3. Ejemplo de nota adjunta

6.1.4. Métodos

El método u operación de una clase es la acción que puede realizar, esta acción puede ser tanto una realización física como una entrega de datos, en este último caso es necesario incluir el tipo de datos entregado como un atributo.

La devolución de datos generalmente se hace mediante números, ya sea con el valor exacto si este debe expresarse en números o empleándolos como código, normalmente usando el "0" como un valor positivo y el resto de números para otros valores.

6.1.5. Objetos: instanciación

La instancia de una clase es la determinación de un objeto real concreto de las clases, por lo que podemos decir que la clase es un modelo y cada una de las instancias son los resultados específicos extraídos a partir de él.

Usando nuestro ejemplo como muestra la clase sería el modelo usado para registrar a los usuarios de una biblioteca, mientras que las instancias son cada uno de los usuarios registrados con él. Un modelo de usuario, con la clase anterior sería este.

Imagen 4. Ejemplo de instancia

6.1.6. Relaciones: asociaciones

Las clases no suelen ser elementos aislados, sino que forman relaciones con otras clases con el fin de desarrollar su labor, por ejemplo, podemos decir que en una biblioteca existe una relación entre la clase "Usuario" y la clase "Libro".

Generalmente en las asociaciones la relación se da en una dirección mediante un rol.

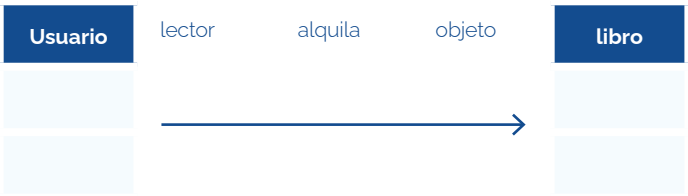


Imagen 5. Ejemplo asociación

La asociación no tiene por que ser bipartita, en este caso un mismo usuario podría adquirir varios libros simultáneamente, por lo que podemos encontrar una asociación formada por diversos elementos unidos a uno solo, en este caso diversas instancias de Libro asociadas a un único Usuario.

Es posible establecer otro tipo de asociaciones en función a diversas circunstancias:

- El caso de que una instancia solo pueda asociarse con un grupo de instancias entre las que debe elegir, se debe marcar con líneas discontinuas y la cláusula {or}.

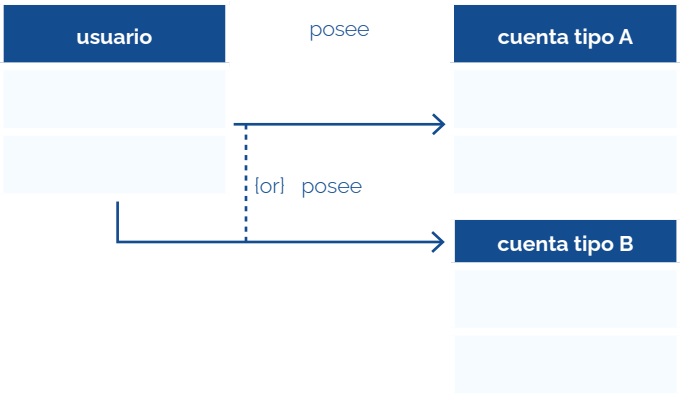


Imagen 6. Ejemplo asociación de múltiple elección con restricción

- El caso de que una asociación funcione como una clase, y por tanto se asocie a su vez con otra clase, en este caso la asociación Posee se asocia con la instancia Acuerdo.

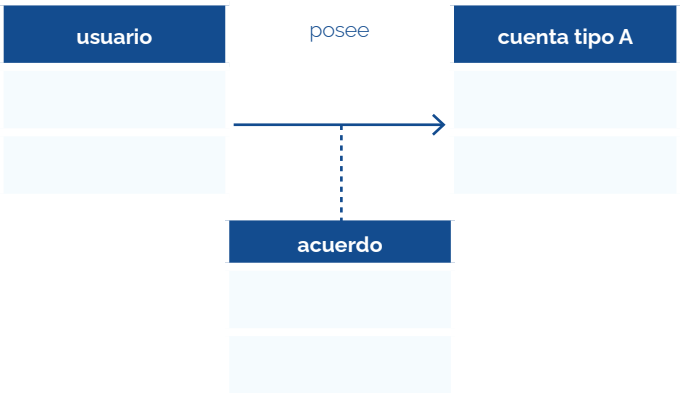


Imagen 7. Ejemplo de asociación de múltiples niveles

- El caso de que se puedan especificar las instancias durante la asociación, en este caso se demarcaría el número exacto o el rango posible, en el caso de no existir un máximo se marcaría con "n" o con "*", como ejemplo se muestra un número exacto, un número claramente demarcado y un número sin máximo determinado: 1, 1 .. 5, 1 .. n, etc. Los elementos que demarcan un número no especificado entre una gama de números se deben intercalar con dos puntos, si se dan opciones a elegir deben separarse por comas: 1,5,6, en este caso solo se podría escoger uno de estos tres números.

Imagen 8. Ejemplo de asociaciones con restricciones de cantidad

- El caso de las asociaciones reflexivas, las cuales conectan una instancia consigo misma o dos instancias de una misma clase, siendo obligatorio el empleo de un rol.

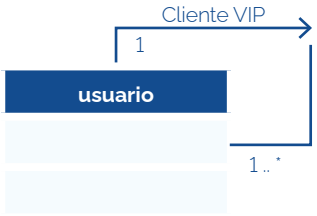


Imagen 9. Ejemplo de asociación reflexiva

6.1.7. Relaciones: herencia

La herencia marca un conjunto de elementos considerados como subclases o clases secundarias que poseen una relación con un elemento clase principal o superclase que las engloba. Estas relaciones pueden darse en diversos niveles, permitiendo que una subclase sea a la ver una superclase para otro elemento inferior la herencia se marca con un triángulo en blanco.

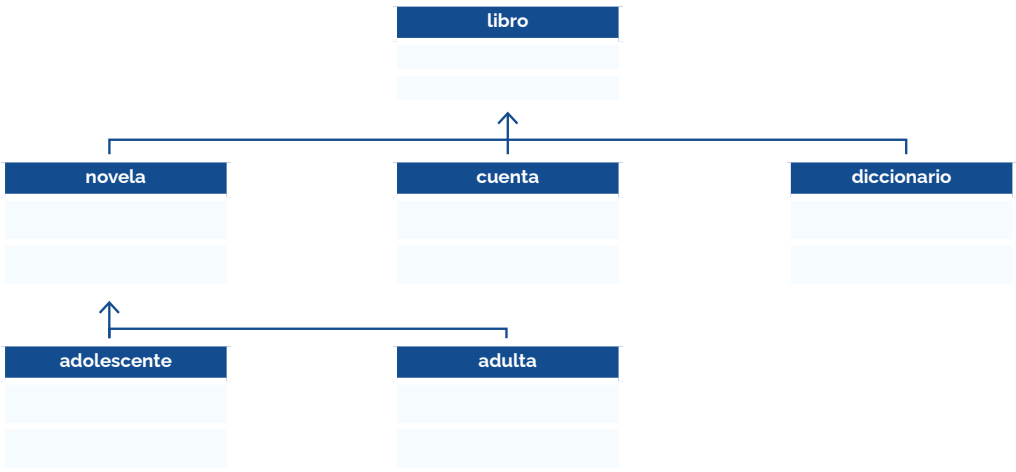


Imagen 10. Ejemplo de clases con relaciones de herencia.

Asociación y herencia

Las subclases, al ser un subtipo, debe poseer los mismos atributos, métodos, etc. que la superclase, en caso de que no se compartan estos sería una asociación en lugar de herencia.

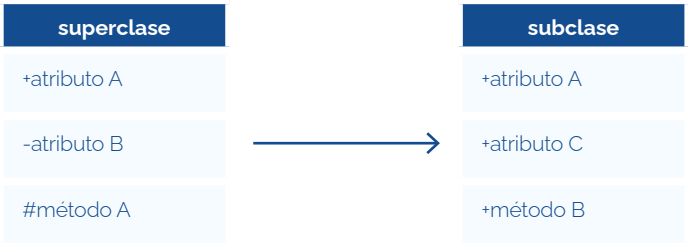
Clases abstractas

Existen clases denominadas abstractas que sirven como modelo superclase, por lo que de ellas nunca se generan instancias, ya que están se crean directamente en su subclase específica. Con el fin de marcar este tipo de clase sin instancias se escribe su nombre en cursiva.

6.1.8. Visibilidad

La subclase permite designar la autorización de visibilidad y uso que una clase posee sobre los atributos y métodos de su superclase. Existen diversas autorizaciones, aunque se emplean tres principalmente:

- **Nivel público:** Se designa con un "+", indica que el elemento puede ser tanto visualizado como heredado por una subclase.
- **Nivel protegido:** Se designa con un "#", indica que el elemento puede ser visualizado, pero no heredado por una subclase.
- **Nivel privado:** Se designa con un "-", indica que el elemento no puede ser visualizado ni heredado por una subclase.



- El atributo A puede ser visto y es heredado por la subclase
- El atributo B no puede ser visto ni heredado por la subclase
- El método A puede ser visto pero no heredado por la subclase

Imagen 11. Ejemplo de visualización y herencia

Los otros dos métodos de señalización son los siguientes:

- **Paquete:** Se indica con un "~", indique que el elemento solo es accesible por las clases del mismo paquete.
- **Estáticas:** Se indica mediante el subrayado, indique que ese elemento es propio de la case, por lo que siempre debe ser visible por sus subclases e incluirse en estas.

Generalmente los valores de los atributos suelen ser privados y permanecer bajo los métodos de setters y getters para lograr los cambios de los atributos de una manera segura, evitando que estos se modifiquen por otros medios o personas ajenas. Al mismo tiempo el uso de los setters permite la comprobación de los valores bajo normas establecidas, de modo que no se incluyan en él valores con un formato no apto.

6.1.9. Relaciones: Composición y agregación

La agregación es la suma de diversas clases con el fin de crear otra que las englobe, de la misma manera que las partes de un coche forman un vehículo completo.

La agregación se marca con un rombo vacío, también se pueden incluir los números de las diversas estancias como ya se ha visto antes diversos elementos.

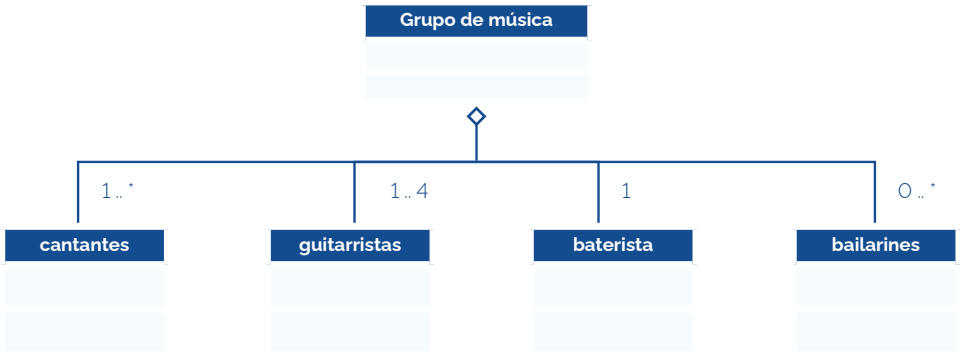


Imagen 12. Ejemplo de agregación

Diferente a la agregación tenemos a la composición, donde la diferencia radica en el empleo de estancias que no se pueden usar en otras agregaciones y por lo tanto son indivisibles, es decir, mientras que el cantante de la agregación puede cambiar de grupo fácilmente, el motor del automóvil permanecerá siempre con este. Por razones lógicas se excluyen de este término los elementos separados por fuerza mayo, un motor puede separarse del automóvil, pero no sin esfuerzo. Se marca con un rombo negro.

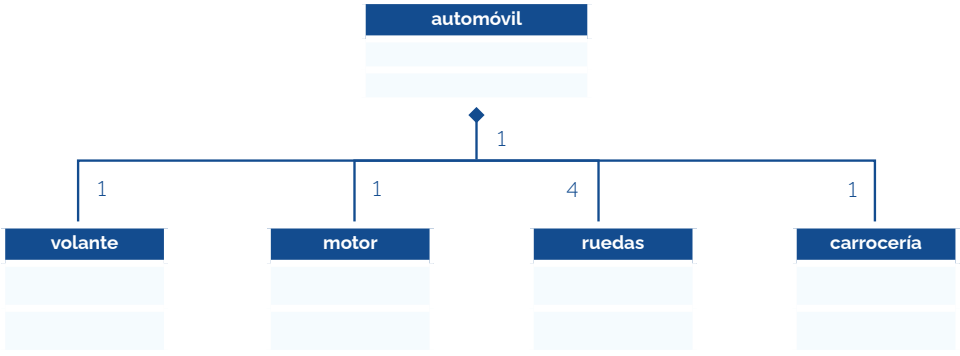


Imagen 13. Ejemplo de composición

6.2. Herramientas para la elaboración de diagramas de clases

Existen programas que simplifican este proceso de creación de diagramas UML como son DIA, el cual posee una licencia GNU, o MagicDraw.

Existe una multitud de estas herramientas para el diseño de diagramas UML en el mercado, con el fin de diferenciar una buena de una mala herramienta es necesario que cumpla con ciertos requisitos, estos requisitos son comunes para todas las herramientas CASE, Computer Aided Software Engineering o Ingeniería de Software Asistida por Computadora, como puede ser:

- Que su uso sea sencillo e intuitivo.
- Que se adapte a las necesidades del usuario como el lenguaje empleado en el código.
- Que permita la inclusión de módulos o plugins.
- Que permita colaboraciones mediante acceso simultaneo a un mismo archivo.
- Que permita la conversión en ambas direcciones:
- De diagrama a código.
- De código a diagrama.

www.universae.com