Actividad 1 Ingenieria de Software

Mónica Yulihet Poveda Carrasco

Profesor

William Alexander Matallana Porras

Universidad de Cundinamarca, Extensión Chía

Faculta de Ingenieria

Ingenieria de software I

2025

Introducción

El diagrama de clases es una herramienta fundamental de modelado en UML 2.5 (Unified Modeling Language, versión 2.5) que permite representar la estructura estática de un sistema mediante la visualización de clases, atributos, métodos y sus relaciones. Según Booch, Rumbaugh y Jacobson (2017), "una clase representa un conjunto de objetos similares que se encuentran en el sistema. Las clases poseen atributos y métodos. Un atributo permite almacenar información conocida para todas las instancias, pero que tiene valores específicos diferentes para cada instancia".

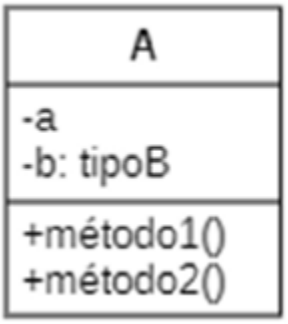
El ejercicio realizado implementa un sistema de gestión de compras basado en un diagrama de clases previamente diseñado, aplicando los principios fundamentales del diseño orientado a objetos. El sistema modela un escenario de comercio electrónico donde los clientes pueden adquirir diferentes tipos de productos, tanto físicos como digitales. Para su construcción, se siguieron los pasos clave del modelado UML: primero se identificaron las clases principales del sistema (Producto, ProductoFisico, ProductoDigital y Cliente); luego se distinguieron las relaciones entre estas clases, reconociendo puntos en común ; y finalmente se creó la estructura, prestando especial atención a las relaciones de herencia.

1. **Diagrama de clases**

El diagrama de clases es una herramienta de modelado en UML 2.5 (Unified Modeling Language, versión 2.5)que muestra la estructura estática de un sistema. Se utiliza para representar las clases, atributos, métodos y sus relaciones, ayudando a visualizar cómo se organiza el sistema antes de su implementación.

> “Una clase representa un conjunto de objetos similares que se encuentran en el sistema. Las clases poseen atributos y métodos. Un atributo permite almacenar información conocida para todas las instancias, pero que tiene valores específicos diferentes para cada instancia. Por su parte, los métodos determinan un comportamiento específico en objetos concretos” (Booch, Rumbaugh y Jacobson, 2017).

Las clases se representan por medio de rectángulos con tres compartimientos como se muestra a continuación.



El primer compartimiento es para el nombre de la clase; el segundo, para el listado de atributos de la clase y el tercero, para el listado de métodos de la clase.

1.1 ¿Cómo se construye un diagrama de clases?

Los diagramas de clase están estrechamente relacionados con el diseño orientado a objetos. Por lo tanto, entender los conceptos básicos de este enfoque es fundamental para crear diagramas de clase efectivos.

Este tipo de diagramas se utiliza cuando se desea mostrar la vista estática de un sistema o sus funcionalidades. Algunos pasos clave para construir diagramas de clase son :

1. Identificar los nombres de las clases

El primer paso es reconocer los objetos principales del sistema. Las clases suelen corresponder a sustantivos dentro del contexto del problema.

2. Distinguir las relaciones

El siguiente paso es identificar cómo se relacionan entre sí las clases u objetos. Busca puntos en común y abstracciones que te permitan agruparlos cuando construyas el diagrama de clases.

3. Crear la estructura

Comienza agregando los nombres de las clases y conéctalos adecuadamente, prestando especial atención a la cardinalidad y las relaciones de herencia. Los atributos y métodos pueden añadirse después, una vez que la estructura básica del diagrama esté definida.

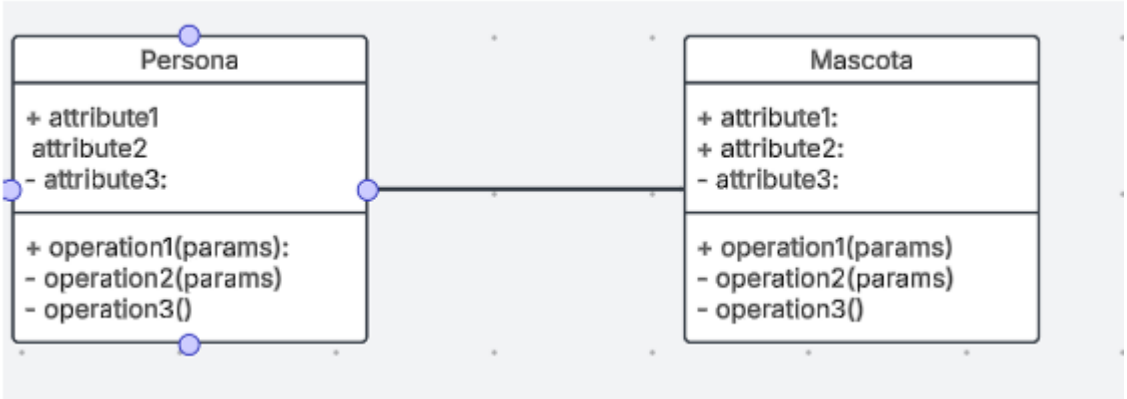
2. Tipos de relaciones del diagrama de clases

En un diagrama de clases, todas las clases están vinculadas entre sí mediante relaciones adecuadas. Estos enlaces ayudan al usuario a comprender a fondo la conexión entre diferentes entidades. Sin embargo, debido a las leves similitudes, muchos a menudo tienen problemas para comprender las diferentes relaciones del diagrama de clases.

Un diagrama de clases incluye los siguientes tipos de relaciones:

2.1 Asociación

Las asociaciones se utilizan para representar los vínculos estáticos entre clases, reflejando la relación estructural entre dos o más clasificadores. Este tipo de relación, que es el más común, se emplea para mostrar la dependencia semántica entre las clases. Se representa mediante una línea continua sólida que conecta las clases, enumerando sus atributos, propiedades y asociaciones.

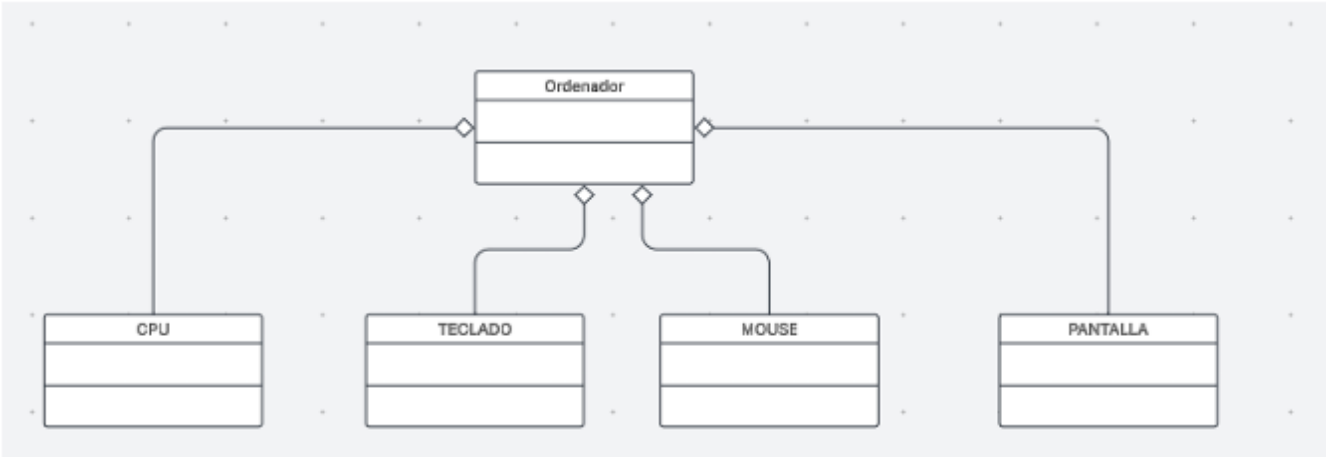


Nota. El diagrama de clases muestra un ejemplo de asociación bidireccional donde una mascota pertenece a una persona.

2.2. Agregación

Es una representación jerárquica que indica a un objeto y las partes que componen ese objeto. Es decir, representa relaciones en las que un objeto es parte de otro, pero aun así debe tener existencia en sí mismo. La forma de representar una relación de agregación en UML es mediante una línea que sale de la clase que está siendo agregada (clase "parte") hacia la clase resultante (clase "todo"), terminada en un rombo vacío en la clase que definimos por agregación Por ejemplo, los tornillos y las tablas forman parte de una mesa. Como ves, el tornillo podría formar parte de más objetos, por lo que interesa especialmente su abstracción en otra clase.

Un ejemplo de esta relación podría ser la de un ordenador.

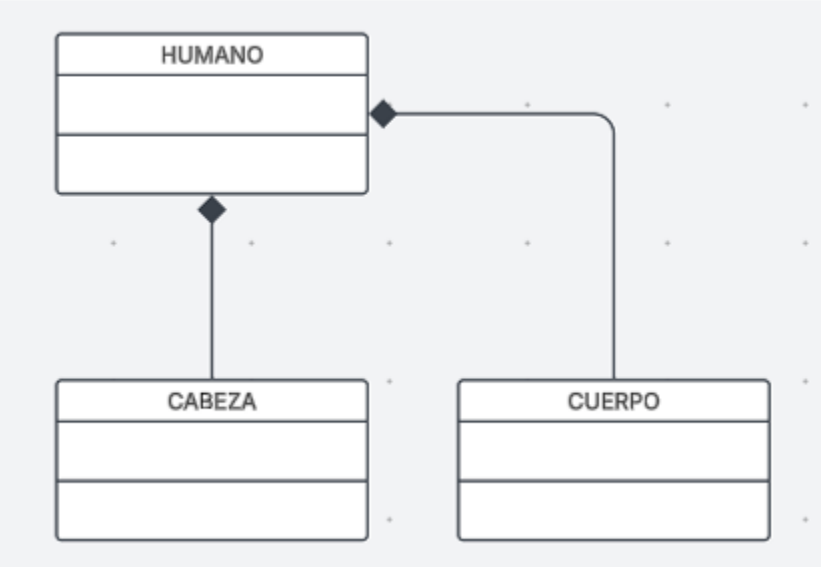


Nota. El diagrama de clase muestra como un ordenador es el resultado de “agregar” diferentes componentes como la “CPU”, “TECLADO” , “RATÓN” y una “PANTALLA” (Como lo pueden ser otros adicionales). Después podemos aplicar el “test” de un objeto de la case de la siguiente manera: “ORDENADOR” tiene -una “CPU” y tiene -un “TECLADO”, y tiene -un “RATÓN”, y tiene -una “PANTALLA”.

2.3 Composición

La composición es una relación jerárquica más fuerte que la agregación. Al igual que la agregación, describe cómo un objeto está compuesto por partes, pero en este caso, las partes no pueden existir sin el objeto principal. Si el objeto que las contiene desaparece, todas las partes también deben desaparecer, ya que no tienen sentido por sí mismas. En otras palabras, las partes dependen completamente del objeto que las compone y no pueden existir de manera independiente.

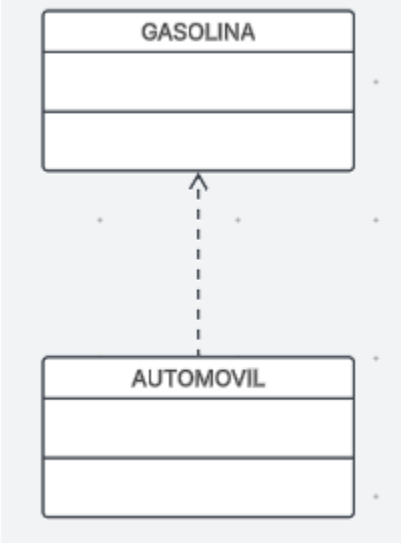
Además, en una relación de composición, los componentes tienen el mismo ciclo de vida que el objeto principal. Esto significa que cuando el objeto principal se elimina, sus partes también lo hacen, Una diferencia clave entre composición y agregación es que, en la composición, los componentes no se comparten entre varios objetos. Cada objeto compuesto tiene sus propias partes, mientras que, en la agregación, un componente puede formar parte de varios objetos al mismo tiempo. En un diagrama de clases, la composición se representa mediante una línea continua con un rombo relleno negro en el extremo que apunta hacia la clase que contiene a los demás un claro ejemplo sería el cuerpo humano.



Nota. Una persona está formada por una cabeza y un cuerpo, los cuales son inseparables y coexisten. Si falta uno de estos elementos, no existe un ser humano.

2.4 Dependencia

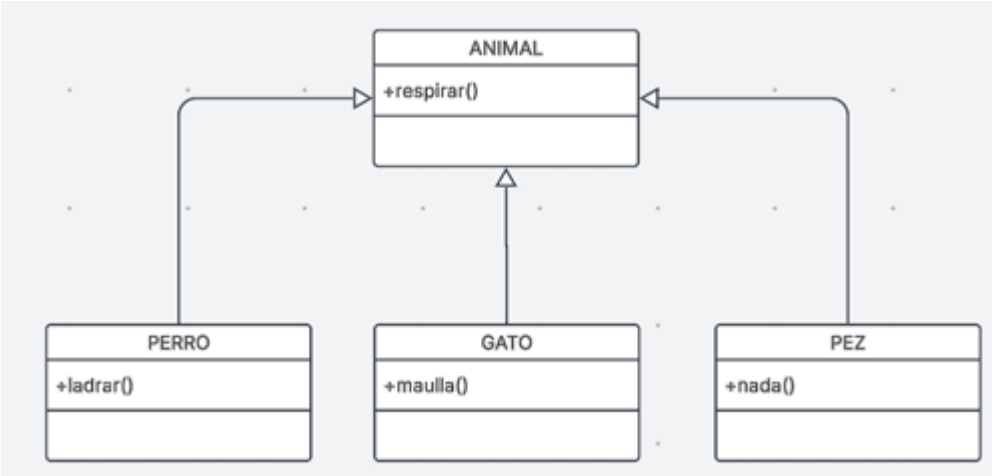
Este es un tipo de relación más débil que se usa cuando un objeto no está contenido en ningún campo. Está representado por una línea discontinua y una punta de flecha que apunta hacia la entidad dependiente. La relación se lee como “una clase A depende de la clase B”, el sentido de la flecha va de A hacia B. Una relación de dependencia indica que una clase depende de otra, es decir, si una clase cambia, la otra también cambia. Por lo tanto, en el diagrama de clases se presenta una cadena de dependencias para hacer la lectura de un archivo de texto. Como se muestra a continuación.



Nota. El auto depende de la gasolina. Si no hay gasolina, el automóvil no podrá conducir.

2.5 Herencia

Un concepto muy importante en diseño orientado a objetos, la herencia, se refiere a la capacidad de una clase (clase hija) para heredar la misma funcionalidad de otra clase (superclase). La herencia indica que una subclase hereda los métodos y atributos especificados por una Super Clase, por ende, la Subclase además de poseer sus propios métodos y atributos, poseerá las características y atributos visibles de la Super Clase. Como se muestra en el siguiente ejemplo.

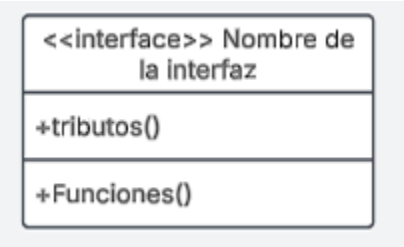


Nota. El diagrama describe como las tres clases (PERRO, GATO,PEZ) podrán utilizar la función respirar, ya que lo heredan de la clase animal, pero solamente la clase Pez podrá nadar, la clase Perro ladrar y la clase Gato maullar.

2.6 Interfaces

Una interfaz es una entidad que declara una serie de atributos, funciones y obligaciones. Es una especie de contrato donde toda instancia asociada a una interfaz debe de implementar los servicios que indica aquella interfaz. Dado que únicamente son declaraciones no pueden ser instanciadas. Las

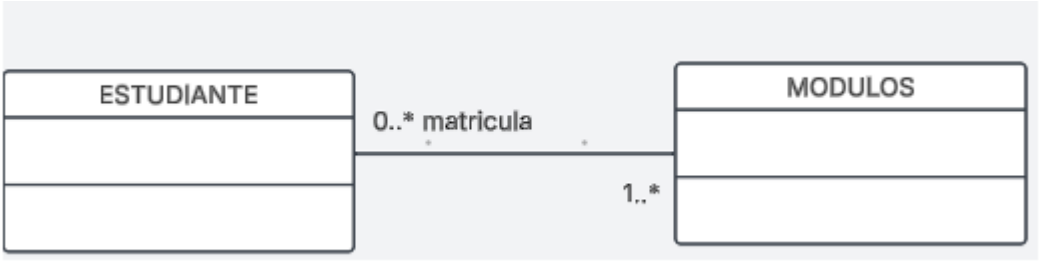
interfaces se asocian a clases. Una asociación entre una clase y una interfaz representa que esa clase cumple con el contrato que indica la interfaz, es decir, incluye aquellas funciones y atributos que indica la interfaz. Su representación es similar a las clases, pero indicando arriba la palabra <<interfaces>>. Como se muestra en el ejemplo a continuación.



2.7 Cardinalidad

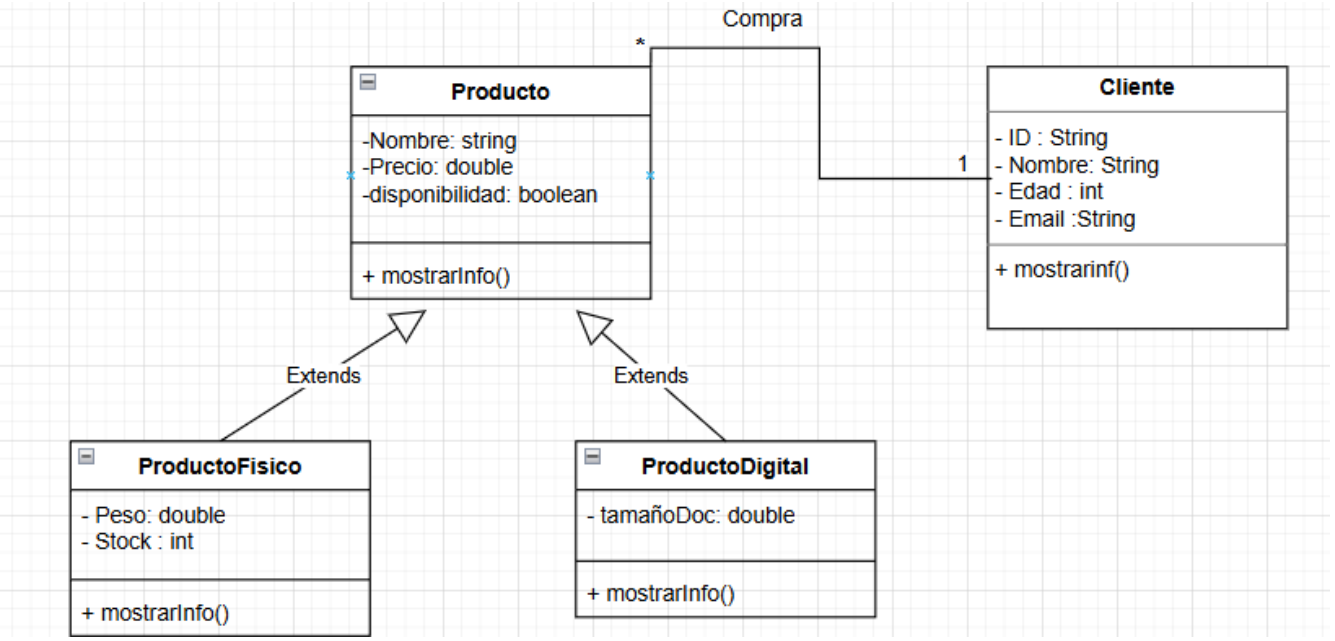
La cardinalidad también conocida como multiplicidad es la asociación lógica activa cuando se está representando la cardinalidad de una clase en relación con otra. En una relación de multiplicidad, puede agregar un número directamente a la línea asociada para indicar el número de objetos en la clase correspondiente.

- 1..1: Sólo uno
- 0..*: cero o más
- 1..*:uno o mas
- 0..1: Ninguno o solo uno
- m..n: al menos m, como máximo n (m<=n)



Nota. Quiere decir que la clase Alumno se relaciona con la clase Módulo debido a que los alumnos se matriculan en diferentes módulos y en un módulo puede estar matriculado alumnos. La cardinalidad indicada quiere decir que todo alumno está matriculado en al menos un módulo y puede estar matriculado en varios y que en un módulo puede haber varios alumnos matriculados y puede ser que en un módulo no haya nadie matriculado.

3 . Ejercicio de diagramas de clase con relaciones



El diagrama representa el modelo de clases de un sistema básico de tienda en línea. La clase padre es Producto, la cual contiene atributos generales como nombre, precio y disponibilidad, además del método mostrarInfo(), que permite presentar la información del producto.

A partir de Producto se establecen relaciones de herencia con dos clases derivadas: ProductoFísico y ProductoDigital. Esto significa que ambas clases comparten las características de Producto, pero añaden atributos específicos. ProductoFísico incluye atributos como peso y stock, mientras que ProductoDigital incorpora el atributo tamañoDoc, relacionado con el tamaño del archivo digital.

Por otra parte, se incluye la clase Cliente, que almacena información básica del usuario como ID, nombre, edad y correo electrónico, junto con el método mostrarInfo() e indicando que existe una relación cliente producto ya que un cliente puede realizar una o varias compras de distintos productos. Esta relación permite representar de manera más precisa el proceso de adquisición dentro del sistema y evita una relación directa simple entre Cliente y Producto.

En conjunto, el diagrama combina herencia para clasificar los tipos de productos y asociación a través de Compra para modelar la interacción entre clientes y productos dentro de la tienda.

Conclusiones

El diagrama de clases elaborado permitió aplicar y comprender los principios fundamentales del modelado orientado a objetos y del lenguaje UML , demostrando su utilidad para representar de manera estructurada la realidad de un sistema de información. A través de la identificación de clases, atributos, métodos y relaciones, fue posible organizar conceptualmente el funcionamiento de una tienda en línea antes de su implementación, evidenciando que el uso de UML facilita la comunicación, el análisis y el diseño de sistemas de software.

La utilización de relaciones como la herencia y la asociación, permitió modelar de forma adecuada tanto la clasificación de los productos como la interacción entre clientes y el sistema. Esto demuestra que un buen diseño de diagrama de clases no solo describe los elementos del sistema, sino que también refleja sus comportamientos y dependencias, contribuyendo a un diseño más claro, modular y fácilmente comprensible para futuros desarrollos o mejoras del sistema.

Referencias

Diagrama de clases. (2018, agosto 21). Diagramasuml.com; admin. https://diagramasuml.com/diagrama-de-clases/

Diagrama de clases: Qué es, cómo hacerlo y ejemplos. (s/f). https://miro.com/. Recuperado el 10 de Febrero de 2026, de https://miro.com/es/diagrama/que-es-diagrama-clases-uml/

Relaciones en diagramas de clases UML. (s/f). Edrawsoft. Recuperado el 10 de Febrero de 2026, de https://www.edrawsoft.com/es/article/classdiagramrelationships.html?srsltid=AfmBOoplcgD0U8m32OQKZ_8KNNT4tJDmGExgdIopJ_jDj7rwUHXlAvFg

3.4.1.- Cardinalidad o multiplicidad de la relación. (s/f). Caib.es. Recuperado el 10 de Febrero de 2026, de https://sarreplec.caib.es/pluginfile.php/11369/mod_resource/content/16/ED05_Contenidos_Web/341_cardinalidad_o_multiplicidad_de_la_relacin.html

¿Cuáles Son Los Seis Tipos De Relaciones En Los Diagramas De Clases UML? - Visual Paradigm Blog Español. (2022, febrero 9). Visual Paradigm Blog Español. https://blog.visualparadigm.com/es/what-are-the-six-types-of-relationships-in-uml-class-diagrams/

Diagrama de clases. (s/f). Dosideas.com. Recuperado el 10 de Febrero de 2026, de https://dosideas.com/wiki/Diagrama_de_clases