Diagramas de Clase

Pedro Jose Garcia Correa

Universidad de Cundinamarca

Ingeniería de Software

William Alexander Matallana Porras

10 de febrero del 2026

Introducción

La programación orientada a objetos permite tener un orden más definido en el código, pues se organiza todo en "objetos" que representan cosas reales. Cada objeto cuenta con sus propias características y puede hacer ciertas acciones.

Los diagramas de clase son los planos de diagramación de estos objetos. Cómo, por ejemplo, construir una casa: primero miras el diagrama que muestra todos los materiales y cómo se conectan. Eso es un diagrama de clases, un esquema que te dice qué clases se necesita y cómo se relacionan entre sí antes de la programación.

La Programación Orientada a Objetos: Lo Primordial.

Conceptos clave en POO:

Encapsulación

Principio de programación que consiste en agrupar datos y comportamientos relacionados en una unidad llamada clase mientras se bloquea el acceso directo a algunos componentes internos. La encapsulación facilita el mantenimiento del código al encontrar cambios en componentes específicos sin afectar diferentes partes del sistema.

Herencia

Permite crear nuevas clases basadas en clases ya existentes, donde la nueva clase adquiere los atributos y métodos de la clase base. Esto ayuda a la reutilización de código y la creación de jerarquías que representan relaciones de generalización-especialización.

Polimorfismo

Capacidad de objetos de diferentes clases para responder un mismo mensaje o llamada de método de formas distintas. Lo que permite que una interfaz única pueda ser utilizada para diferentes tipos de datos. El polimorfismo se implementa mediante la sobreescritura de métodos

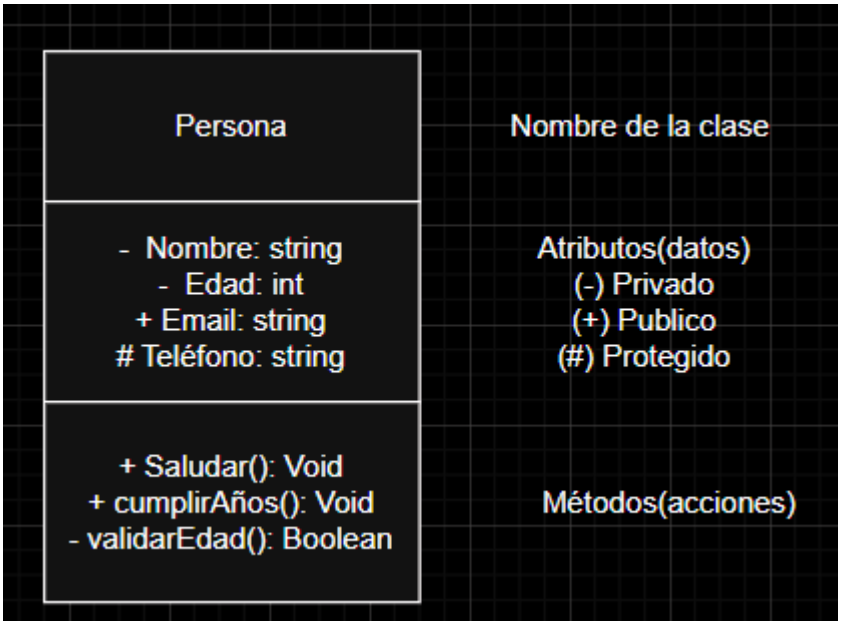
en jerarquías de herencia o interfaces que varias clases pueden implementar de manera diferente.

abstracción

Proceso de identificar las características esenciales de un objeto en el mundo real y representarlas en un modelo computacional. La abstracción se logra mediante clases abstractas e interfaces que definen estructuras y comportamientos sin extenderse en su funcionamiento; solo se muestra lo necesario, mas no lo complejo. (Oracle, s.f.)

Diagramas de clase

Son diseños técnicos que usan símbolos estándar para mostrar cómo se organizarán las clases en el programa.



Símbolos de visibilidad:

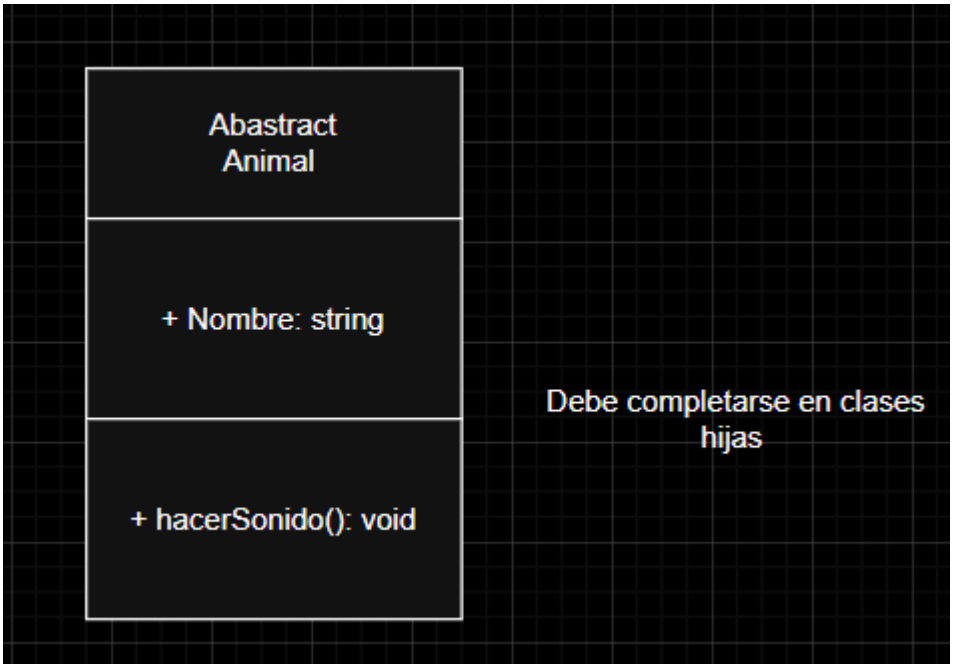
+ Público: Cualquier parte del programa puede acceder.

- Privado: Solo la propia clase puede acceder.

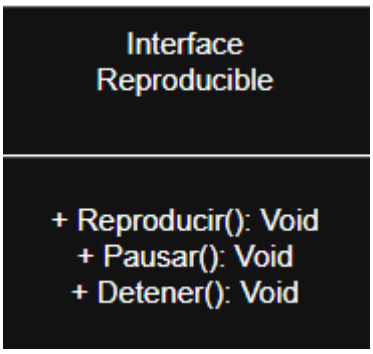
Protegido: La clase y sus "hijos" pueden acceder.

Tipos especiales de clases:

Clases abstractas: Son clases incompletas que no permiten crear objetos directamente. Su función es proporcionar una base común y obligatoria para una jerarquía. Utilizando la herencia en sus clases hijas para completar los detalles específicos.



Interfaces: Es un contrato que define un conjunto de comportamientos obligatorios sin especificar el cómo realizarlos. Cuando se implementa una clase, se compromete a cumplir con ciertos métodos, permitiendo que objetos de distintos tipos puedan interactuar de forma estandarizada bajo una estructura común.



Tomado de Visual Paradigm (Paradigm, s.f.)

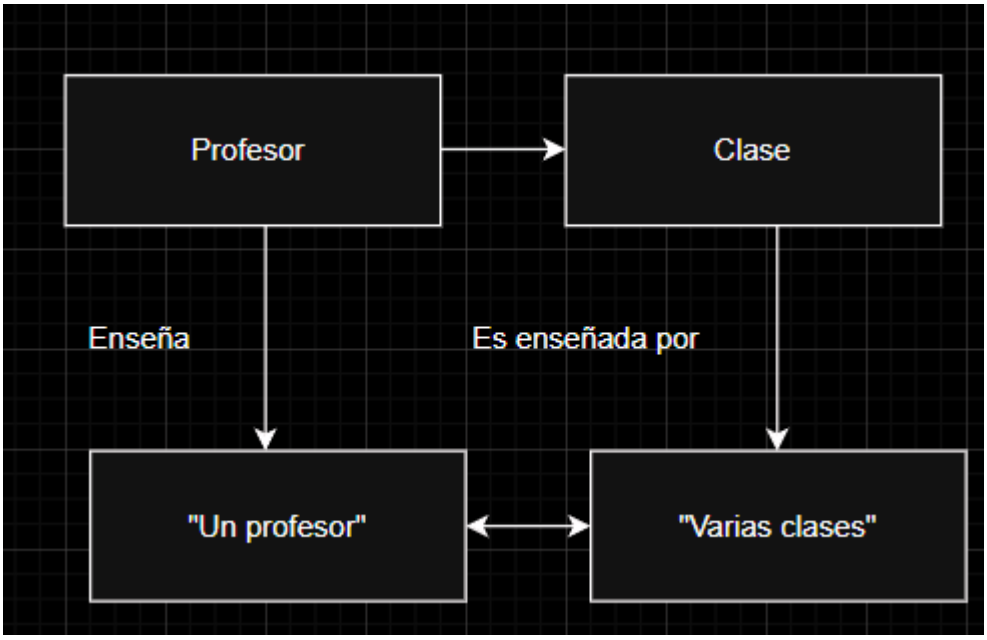
Tipos de Relaciones entre Clases

Asociación – “Se relaciona con”.

Significa que dos clases se encuentran conectadas de alguna manera.

Características clave

- Puede ser unidireccional (A conoce a B, pero B no conoce a A).
- O bidireccional (A y B se conocen mutuamente).
- No implica una posesión o control.



Multiplicidad Común:

- **1:1** - Uno a uno
- **1: *** - Uno a muchos
- ***: *** - Muchos a muchos
- **0..1: *** - Cero o uno a muchos

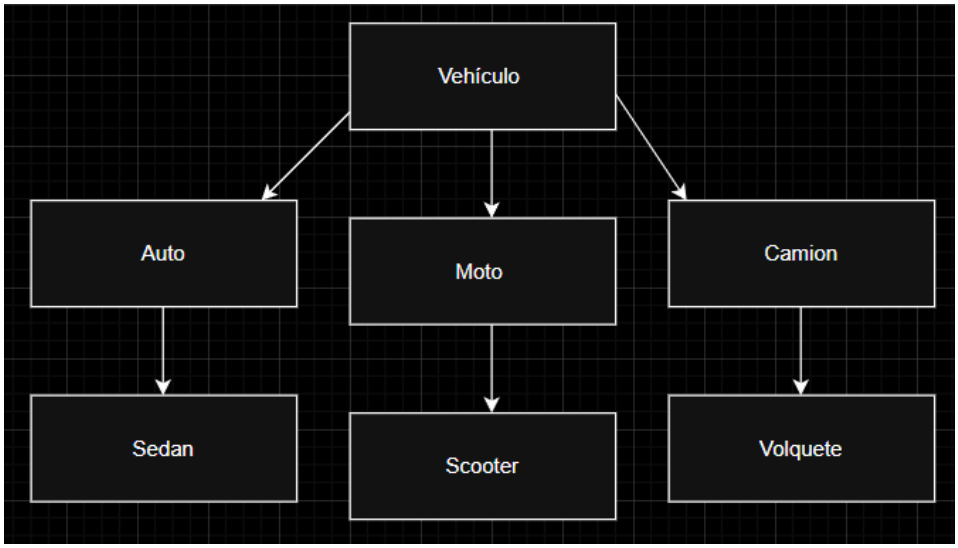
Herencia o Generalización

Cuando una clase es considerada un tipo especial de otra clase. Es la relación "padre-hijo" en programación.

Reglas importantes:

La prueba "es-un" sirve para validar la jerarquía de clases. Si se puede afirmar que una subclase es una versión específica de su superclase, la herencia es correcta.

- "Perro es un Animal"
- "Círculo es una Figura"
- "Auto es un Vehículo"



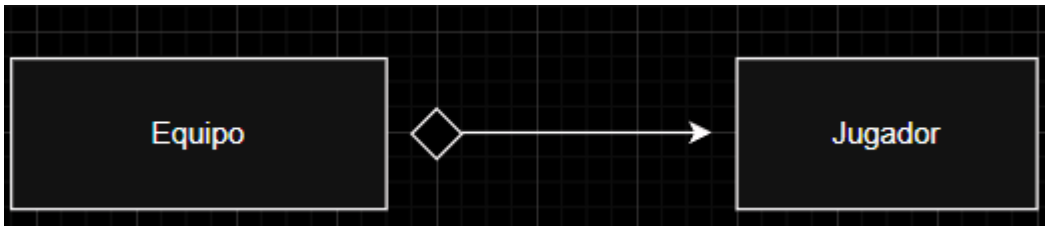
Agregación – “Tiene Un”

Es cuando una clase tiene objetos de otra clase, pero estos objetos pueden existir independientemente.

Ejemplo con Analogía:

Un equipo de fútbol tiene jugadores.

- Si el equipo se disuelve, los jugadores siguen existiendo.
- Los jugadores pueden cambiar de equipo.
- El equipo no creó a los jugadores.



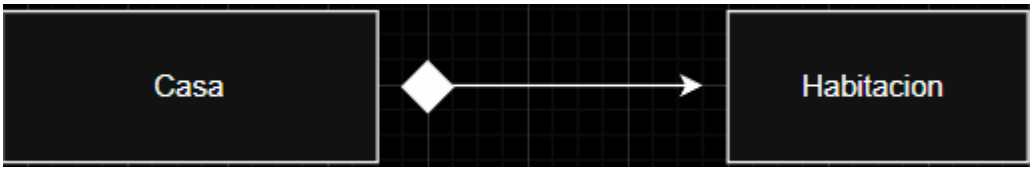
Composición – “Parte de”

Representa una relación donde una clase contiene a otra de forma dependiente. A diferencia de la herencia, si el objeto principal desaparece, sus partes también dejan de existir.

Ejemplo con Analogía:

Una casa tiene habitaciones.

- Si la casa se destruye, las habitaciones también
- Las habitaciones no existen sin la casa
- La casa crea sus propias habitaciones



Dependencia – “Usa temporalmente”

Es un vínculo de uso temporal. Ocurre cuando una clase necesita a otra únicamente para realizar una acción específica.

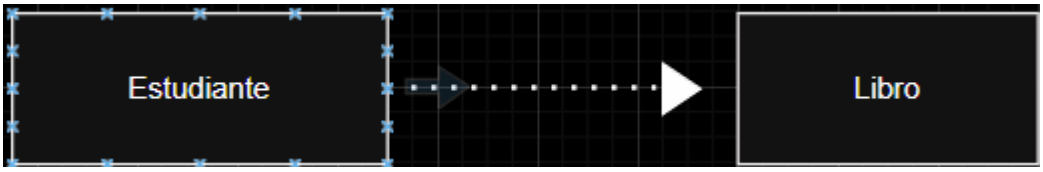
Situaciones Comunes:

- Parámetro de método.
- Variable local en un método.
- Llamada a método estático.

Ejemplo:

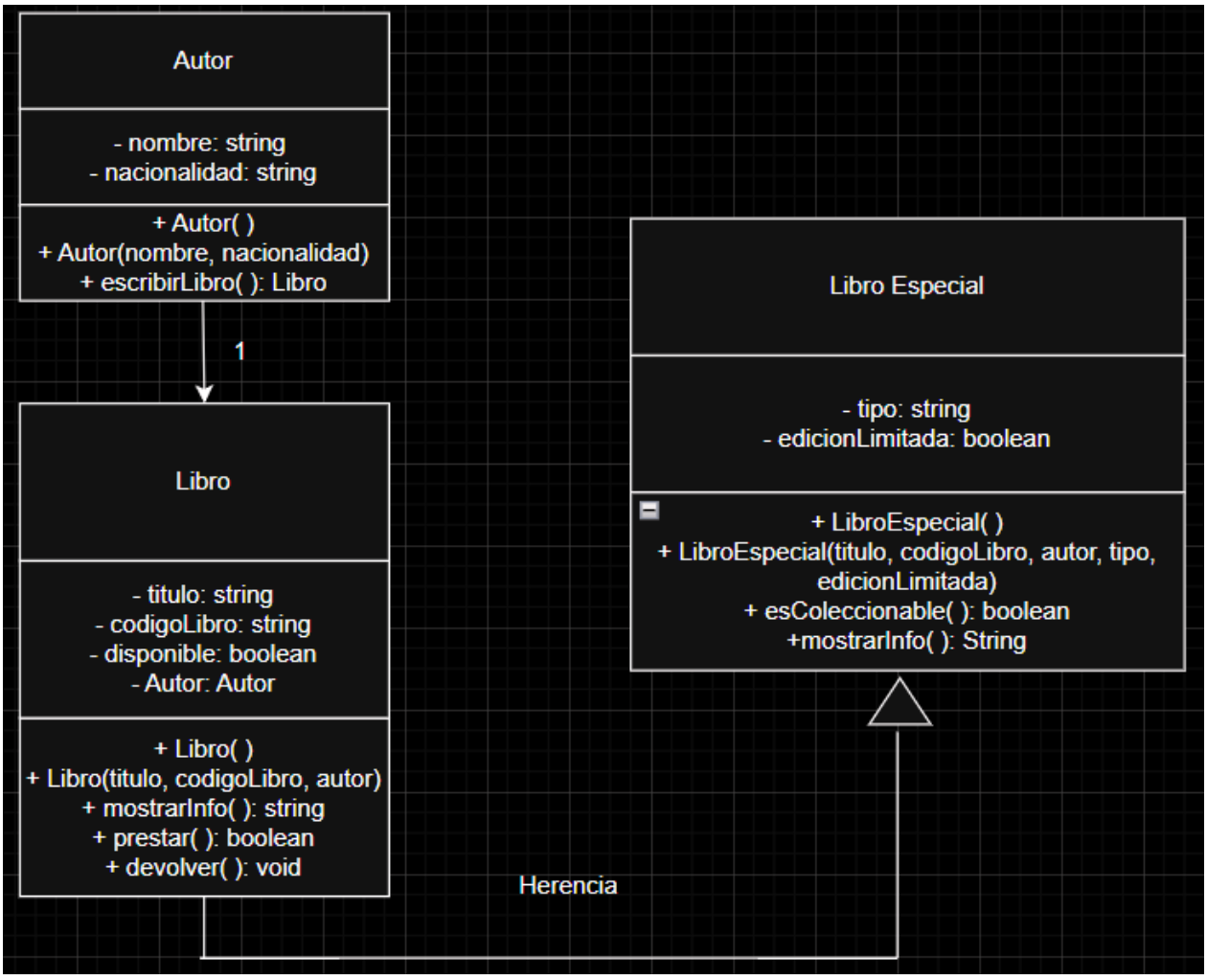
Un Estudiante usa un Libro para estudiar.

- El estudiante no es dueño del libro.
- Solo lo usa durante el estudio.
- Después lo devuelve.



Tomado de GeeksforGeeks (GeeksforGeeks, 2026)

Diagrama de Clase



Conclusiones

Los diagramas de clase son herramientas fundamentales para una programación organizada y profesional. Los diagramas no son simples gráficos, sino los planos de construcción que guían el proceso paso a paso. Dominar esta habilidad marca la diferencia en la práctica de la creación de software.

Entender los diferentes tipos de relaciones entre clases permite diseñar sistemas más flexibles y realistas. Como hemos visto, cada relación tiene un propósito en específico y consecuencias distintas. Saber cuándo usar cada una permite crear programas que sean más versátiles y longevos sin necesidad de reescribir todo de nuevo. Esta comprensión profunda de las relaciones mejora las prácticas al momento de programar y diseñar software, convirtiéndolo en algo fundamental.

Bibliografía

GeeksforGeeks. (21 de Enero de 2026). Obtenido de https://www.geeksforgeeks.org/system-design/unified-modeling-language-uml-class-diagrams/

Oracle. (s.f.). Obtenido de https://docs.oracle.com/javase/tutorial/java/concepts/

Paradigm, V. (s.f.). Obtenido de https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/