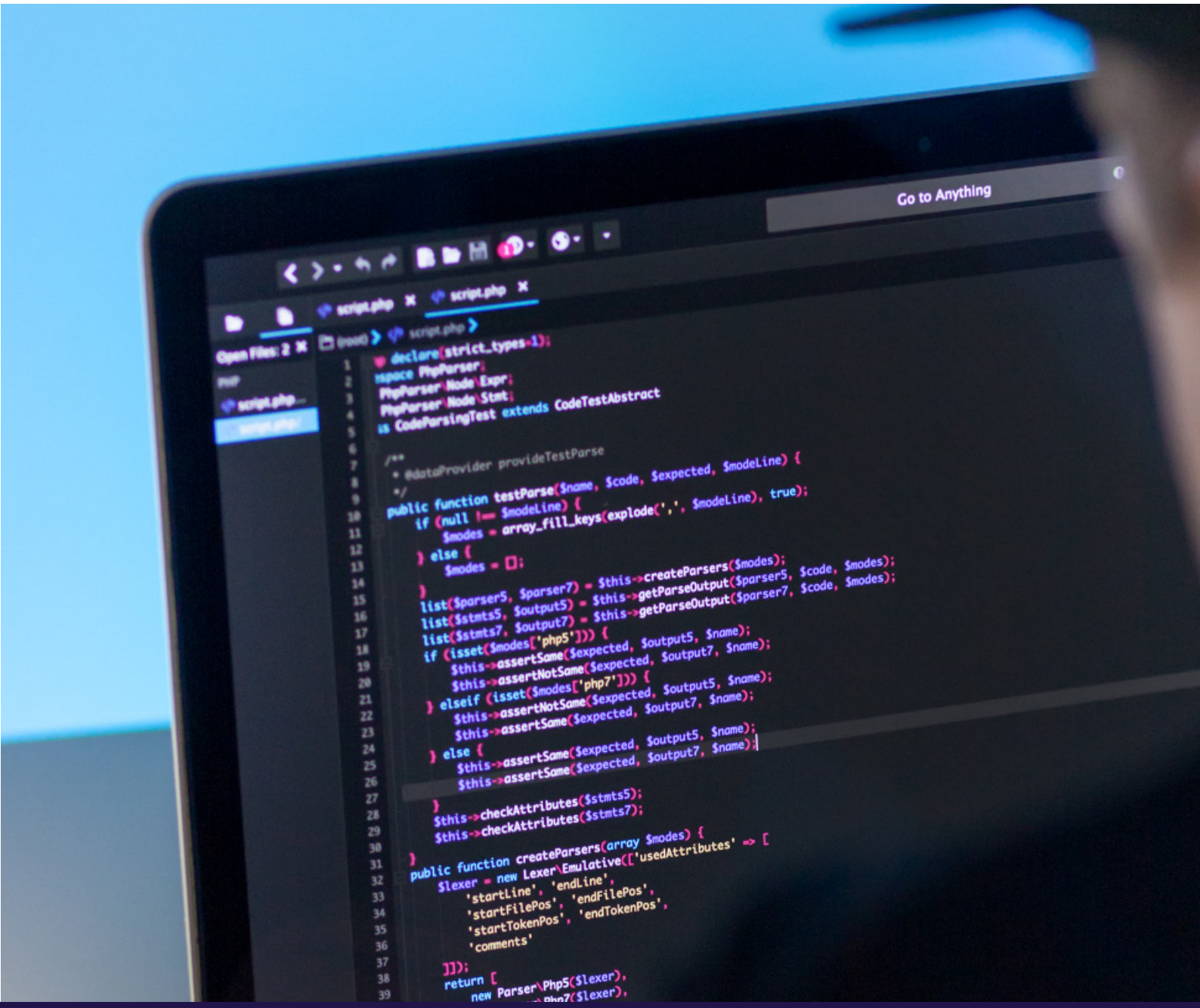
TECNOLÓGICA DEL ORIENTE
INSTITUCIÓN DE EDUCACIÓN SUPERIOR



FUNDAMENTOS DE *SOFTWARE*

RELACIONES ENTRE CLASES EN EL DIAGRAMA DE CLASES

RELACIONES ENTRE CLASES EN EL DIAGRAMA DE CLASES

Las relaciones entre clases, son fundamentales en un diagrama de clases, porque representan cómo interactúan, dependen, o están conectadas entre sí. Estas relaciones permiten modelar la estructura lógica del sistema y determinar las dependencias entre los distintos componentes.

A continuación, se describen en detalle, los tipos principales de relaciones entre clases, su representación, características y ejemplos.

1. Asociación

La asociación representa una relación estructural entre dos clases, donde una clase está **relacionada con otra, de alguna manera.**

Características:

- Bidireccional o unidireccional:

  o Puede ser de dos sentidos (bidireccional), donde ambas clases son conscientes de la relación, o de un solo sentido (unidireccional), donde solo una clase conoce la relación.

  o Representada gráficamente con una línea sólida entre las clases.

Cardinalidad:

La cardinalidad define cuántas instancias de una clase, están asociadas con cuantas instancias, de la otra clase.

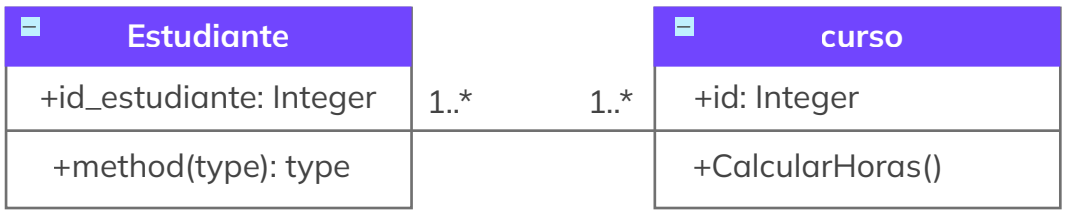
Se representa con notaciones como:

- 1: Una instancia.
- *: Muchas instancias.
- 0..1: Ninguna o ninguna instancia.
- 1..*: Al menos una instancia.

Ejemplo:

Un “Estudiante” se asocia con “Cursos”. Un estudiante puede estar inscrito en varios cursos, y un curso puede tener muchos estudiantes.

Figura 1: Relación en diagrama de clases



2. Agregación

La agregación es una forma especial de asociación que describe una relación "todo-parte". Una clase contiene a otra, pero las partes pueden existir independientemente del todo.

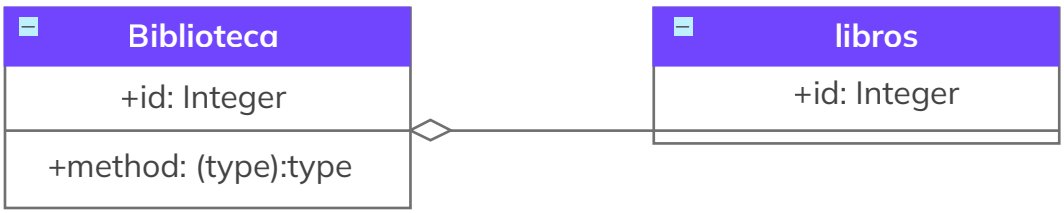
Características:

- Representada gráficamente con una línea sólida con un rombo vacío en el extremo del "todo".
- La existencia de las partes no depende del todo.

Ejemplo:

Una "Biblioteca" contiene varios "Libros". Aunque se elimine la biblioteca, los libros pueden seguir existiendo de manera independiente.

Figura 2: Relación de Agregación



3. Composición

La composición es una relación "todo-parte" más fuerte que la agregación, donde las partes no pueden existir sin el todo. Es decir, si se destruye el todo, las partes también son eliminadas.

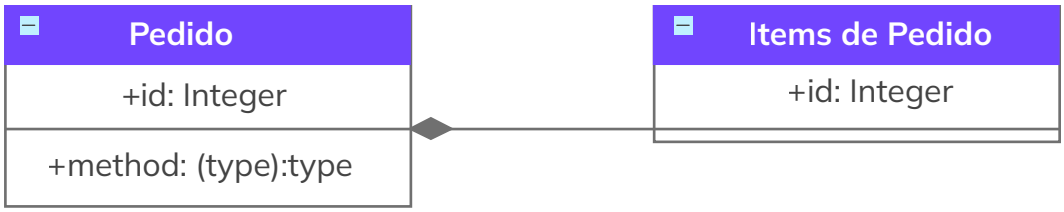
Características:

- Representada gráficamente con una línea sólida con un rombo relleno en el extremo del "todo".
- Las partes dependen completamente del todo para su existencia.

Ejemplo:

Un “Pedido” contiene “Ítems de pedido”. Si se elimina el pedido, también se eliminan los artículos asociados.

Figura 3: Relación de composición



4. Generalización (herencia)

La generalización (o herencia) es una relación jerárquica donde una clase hija hereda atributos y métodos de una clase padre. Representa una relación “es-un”.

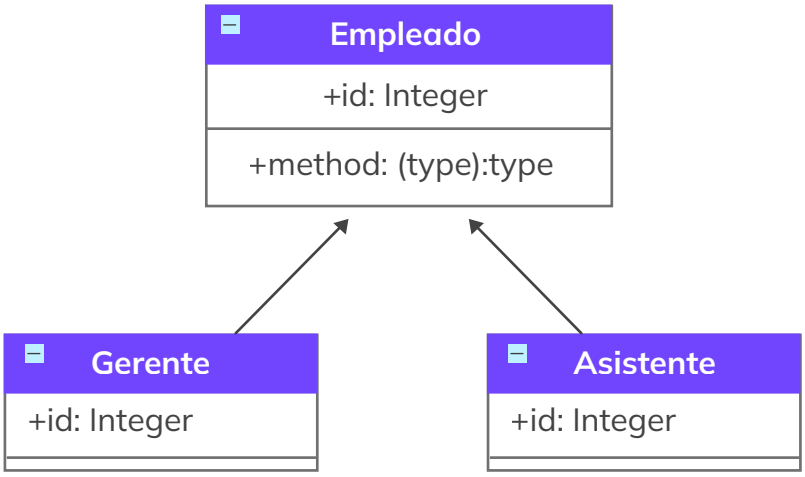
Características:

- Representada gráficamente con una línea sólida con un triángulo vacío, apuntando hacia la clase padre.
- Las clases hijas pueden extender o sobrescribir el comportamiento de la clase padre.

Ejemplo:

Una clase “Empleado” puede tener subclases como “Gerente” y “Asistente”, que heredan sus propiedades básicas, pero también pueden tener atributos o métodos específicos.

Figura 4: Relación de generalización



5. Dependencia

La dependencia es una relación débil y temporal entre dos clases, donde una clase utiliza otra para realizar una acción específica.

Características:

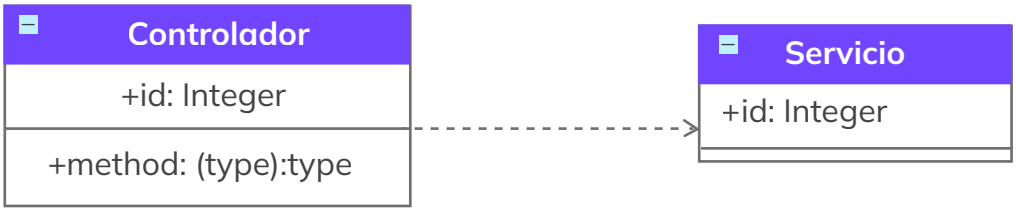
• Representada gráficamente con una línea punteada con una flecha que apunta hacia la clase utilizada.

• Indica una relación de “usa-un” o “depende de”.

Ejemplo:

Un “Controlador” depende de un “Servicio” para realizar operaciones.

Figura 5: Relación de dependencia



6. Realización

La realización describe la relación entre una interfaz y las clases que implementan esa interfaz.

Características:

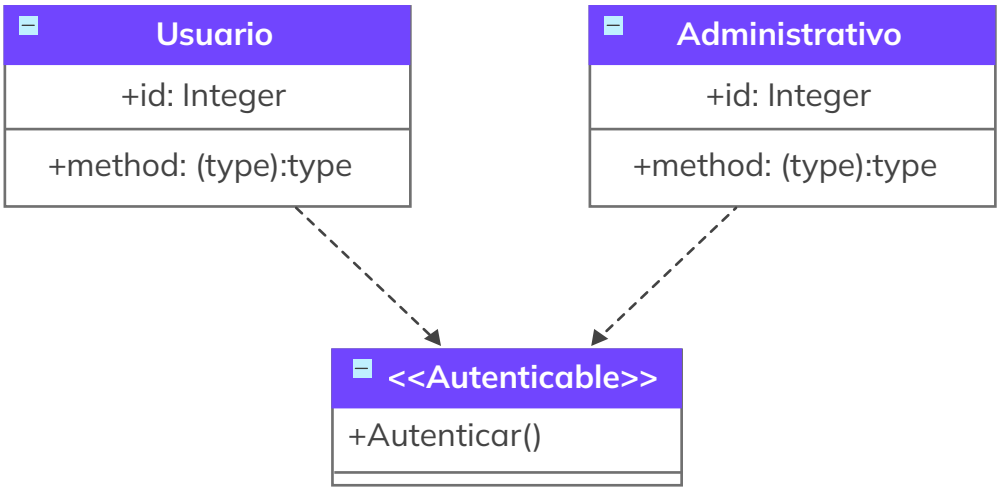
• Representada gráficamente con una línea punteada, con un triángulo vacío apuntando hacia la interfaz.

• Las clases concretas deben proporcionar la implementación de los métodos definidos en la interfaz.

Ejemplo:

Una interfaz “Autenticable” tiene un método autenticar (). Las clases “Usuario” y “Administrador” implementan esta interfaz.

Figura 6: Diagrama de clases



Ventajas de usar relaciones en el diseño de software:

- **Claridad del modelo:** las relaciones ofrecen una representación precisa y comprensible de cómo interactúan los componentes de un sistema.
- **Reutilización:** herencia y composición fomentan la reutilización del código, disminuyendo la redundancia.
- **Mantenimiento:** representar dependencias y asociaciones facilita identificar y modificar partes específicas del sistema.
- **Modularidad:** relaciones como agregación y composición, contribuyen a un diseño modular, promoviendo la extensibilidad.
- **Facilita la documentación:** relacionar componentes de manera explícita en un diagrama, permite comprender y comunicar mejor el diseño a los interesados.