Diagrama de clases

Lizeth Vanessa Moreno Moreno

Universidad de Cundinamarca, Extensión Chia

Facultad ingeniería, Programa sistemas y Computación

Ingeniería de Software 1

William Alexander Matallana Porras

10 de feb. de 26

Introducción

Los diagramas de clase cumplen la misma función que un plano construido por un arquitecto para una edificación sin embargo los diagramas de clase son empleados para el desarrollo de software, este permite visualizar mejor cada aspecto de un programa a la hora de codificar

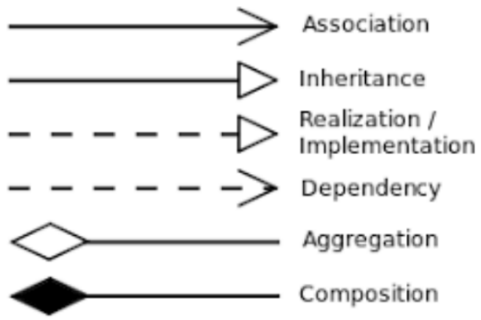
Los diagramas de clases del Lenguaje Unificado de Modelado (UML) son esenciales para la interpretación de cualquier código, desglosare los seis tipos de relaciones de clase existentes en este documento.

Desarrollo

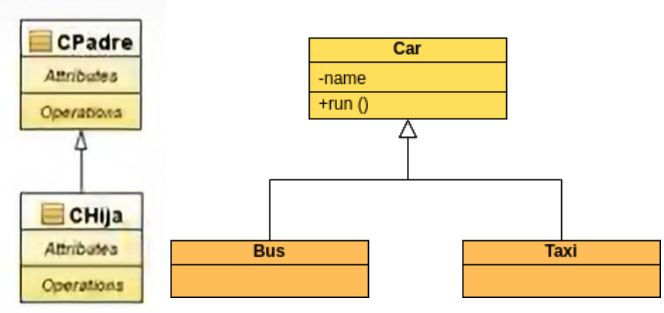
En la programación orientada a objetos en una clase se definen las características y comportamientos, en las clases se decide como deben ser y que pueden hacer los objetos “En un diagrama de clases, los nombres de las clases son los mismos que los de los objetos, porque la finalidad de una clase es definir los atributos” lo que quiere decir que los objetos van ha tomar el mismo nombre de la clase para facilitar la identificación de sus atributos y comportamientos

Un diagrama de clases tiene como finalidad mostrar y analizar las relaciones entre las clases, esto nos ayuda a comprender como funciona el sistema, como se comporta y como se relaciona, estos diagramas muestran en un rectángulo de arriba abajo el nombre de la clase, sus atributos y sus métodos,

Hay seis tipos principales de relaciones entre clases y estas se muestran mediante líneas y puntas de flecha que tienen significados diferentes

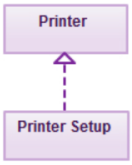


1. **Herencia**
   Es un tipo de relación en la que una clase es hija de otra y obtiene las mismas funciones de la clase padre, para mostrar la herencia en un diagrama se dibuja una línea solida desde la clase hija hasta la clase padre utilizando una punta de flecha sin rellenar



2. **Realización / Implementación**
   Es una relación que se da cuando una clase cumple un contrato definido por una interfaz o una clase abstracta, la interfaz solo dice qué se debe hacer, pero no cómo y la clase que la implementa es la que realmente escribe el código de esos métodos. Esto permite que diferentes clases compartan un mismo comportamiento general, aunque lo realicen de formas distintas.

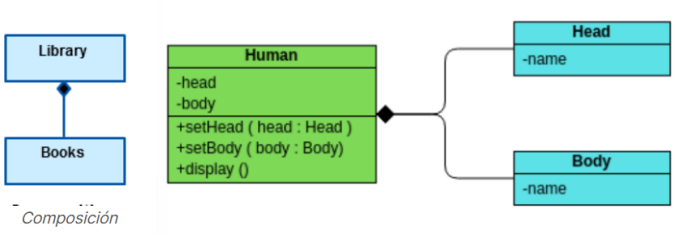
   ejemplo: los automóviles y los barcos son vehículos, y el vehículo es solo un concepto abstracto de una herramienta móvil, y el barco y el vehículo realizan las funciones móviles específicas



Realización

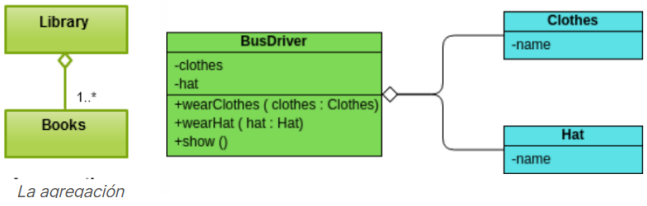
3. **Relación de composición**
   Una clase depende de otra totalmente para existir, esto quiere decir que una cosa está dentro de otra y no puede existir sola, si la clase se principal se elimina la clase que contiene también desaparece. En los diagramas UML esta relación se representa con una línea que une ambas clases un diamante relleno del lado de la clase principal y una flecha que apunta a la clase que depende de ella.

   ejemplo, una persona está compuesta por una cabeza y un cuerpo. Los dos son inseparables y coexisten.



4. **Relación de agregación**

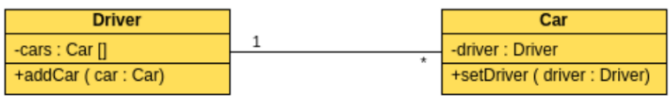
   Una de las clases está formada por otras pero sin que exista una dependencia total entre ellas, la clase principal contiene a las otras como una colección, si la clase principal que agrupa a las demás deja de existir las que están contenidas no dejaran de existir, por ejemplo, una biblioteca contiene libros, pero si la biblioteca se cierra, los libros siguen existiendo. Se representa con una línea que une ambas clases y un diamante vacío colocado junto a la clase principal.



5. **Relaciones de asociación**

   Abarca casi cualquier conexión o relación lógica entre clases, lo que quiere decir que una clase usa a otra, y mantiene la relación, conexión entre dos tipos de objetos sin que uno dependa del ciclo de vida del otro, los objetos pueden existir de manera independiente.

   Ejemplo: Un estudiante puede estar inscrito en varios cursos y un curso puede tener muchos estudiantes, pero ni el estudiante depende del curso para existir ni el curso depende del estudiante.



   Existen dos tipos de asociación:

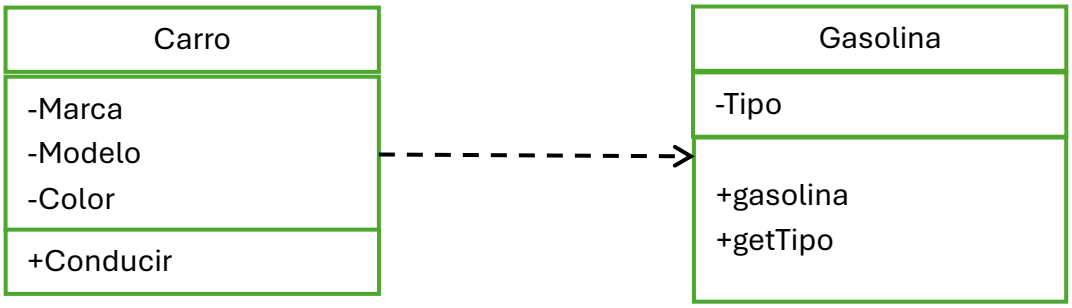
   Asociación dirigida es donde solo una clase conoce o tiene acceso a la otra y no al revés lo que simboliza que la comunicación es en un solo sentido

   Asociación reflexiva esta existe cuando una clase se relaciona consigo misma porque una misma entidad puede cumplir varias funciones en un aeropuerto un miembro del personal pertenece a una sola clase pero puede ser cualquier miembro del personal desempeñando diferentes roles

6. **Dependencias**

Indica que una clase necesita usar a otra para funcionar, aunque no sea una parte permanente, esta relación es débil al no indicar pertenencia ni relación d objetos. Un ejemplo sencillo es el de un auto y la gasolina puesto que el auto necesita la gasolina para funcionar, pero no la contiene como parte de su estructura, sin gasolina el auto no puede andar.

Ejemplo Herencia + dependencia



En el ejemplo diagramado la clase carro cuenta con tiene tres atributos privados, dos constructores: uno vacío y otro con parámetros para inicializar los datos al crear el objeto. La clase Gasolina es sencilla y solo tiene un atributo privado, en el método conducir la clase carro recibe un objeto de tipo Gasolina implementándose una relación de dependencia porque carro usa la clase gasolina solo dentro de ese método para poder funcionar en la clase Main se crean dos objetos de carro y un objeto gasolina y luego se imprimen los datos de los carros usando toString().

Conclusión

Los diagramas de clases son esenciales para la planificación y visualización de la estructura de un sistema antes de escribir el código, sirve como un plano guía donde plasmamos la estructura de forma organizada del programa a realizar, la elección de que diagrama usar depende de la aplicación y la estructura del código que se realice ya sea relación de herencia o una relación combinada entre las diferentes clases.

Hay que tener en cuenta que antes de realizar o inicializar la codificación, empezar a programar es necesaria la planificación estructural del programa por eso los diagramas UML ayudan a un mejor entendimiento a la hora de desarrollar la codificación.

Referencias

https://blog.visual-paradigm.com/es/what-are-the-six-types-of-relationships-in-uml-class-diagrams/

https://miro.com/es/diagrama/que-es-diagrama-clases-uml/

https://creately.com/blog/es/diagramas/relaciones-de-diagrama-de-clases-uml-explicadas-con-ejemplos/#Inheritance

https://programacionymas.com/series/poo-en-java/diagramas-uml-relaciones-entre-clases-e-interfaces