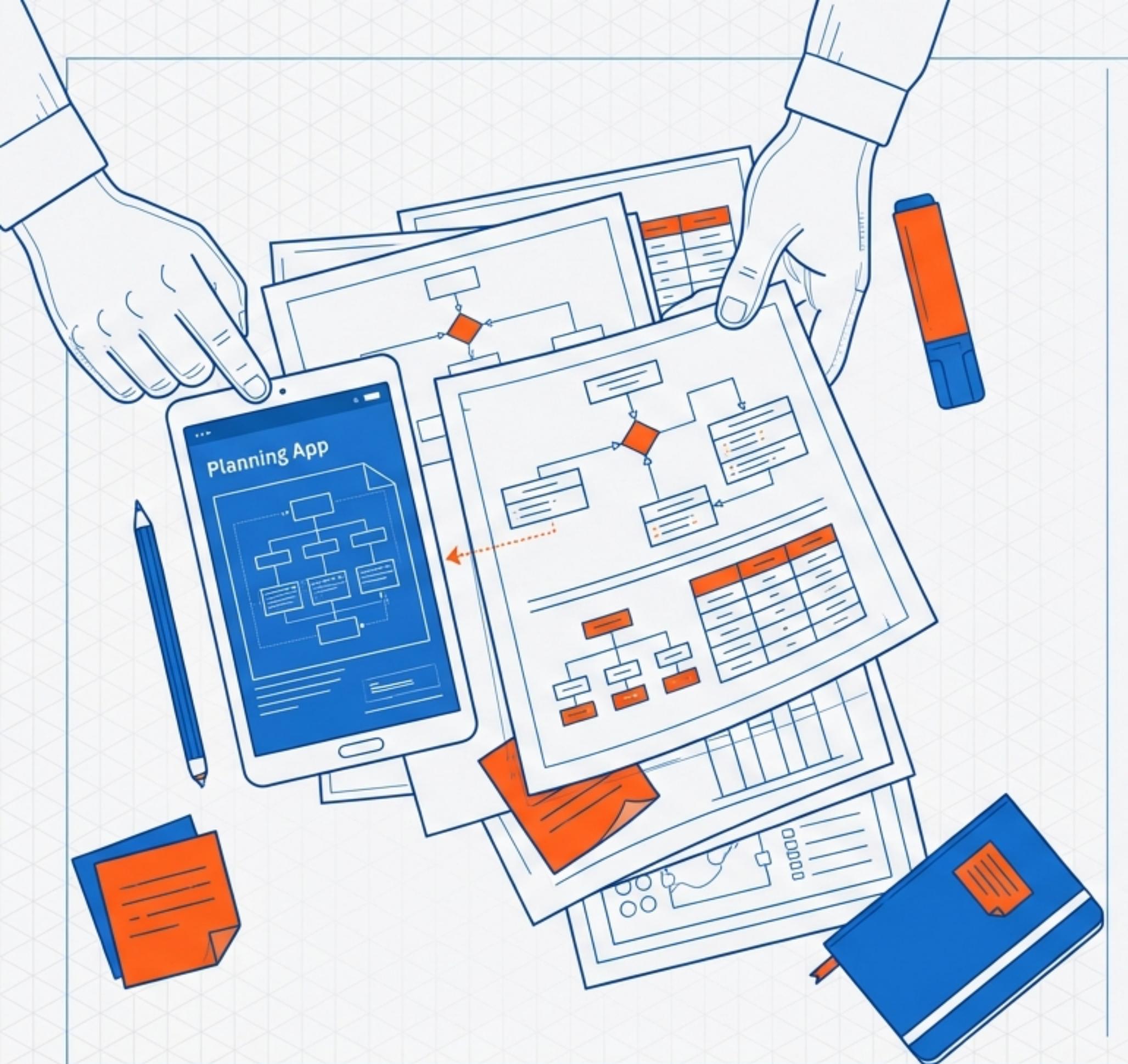


# Diseño y Modelado de Software: El Diagrama de Clases

De la abstracción al código:  
El plano del arquitecto de software.

Basado en el Tema 7: Entornos de Desarrollo



# La Necesidad del Diseño Previo

## El Problema

Desarrollar software sin un plano es como construir sin arquitectura. Las empresas a menudo saltan esta fase, generando "deuda técnica".

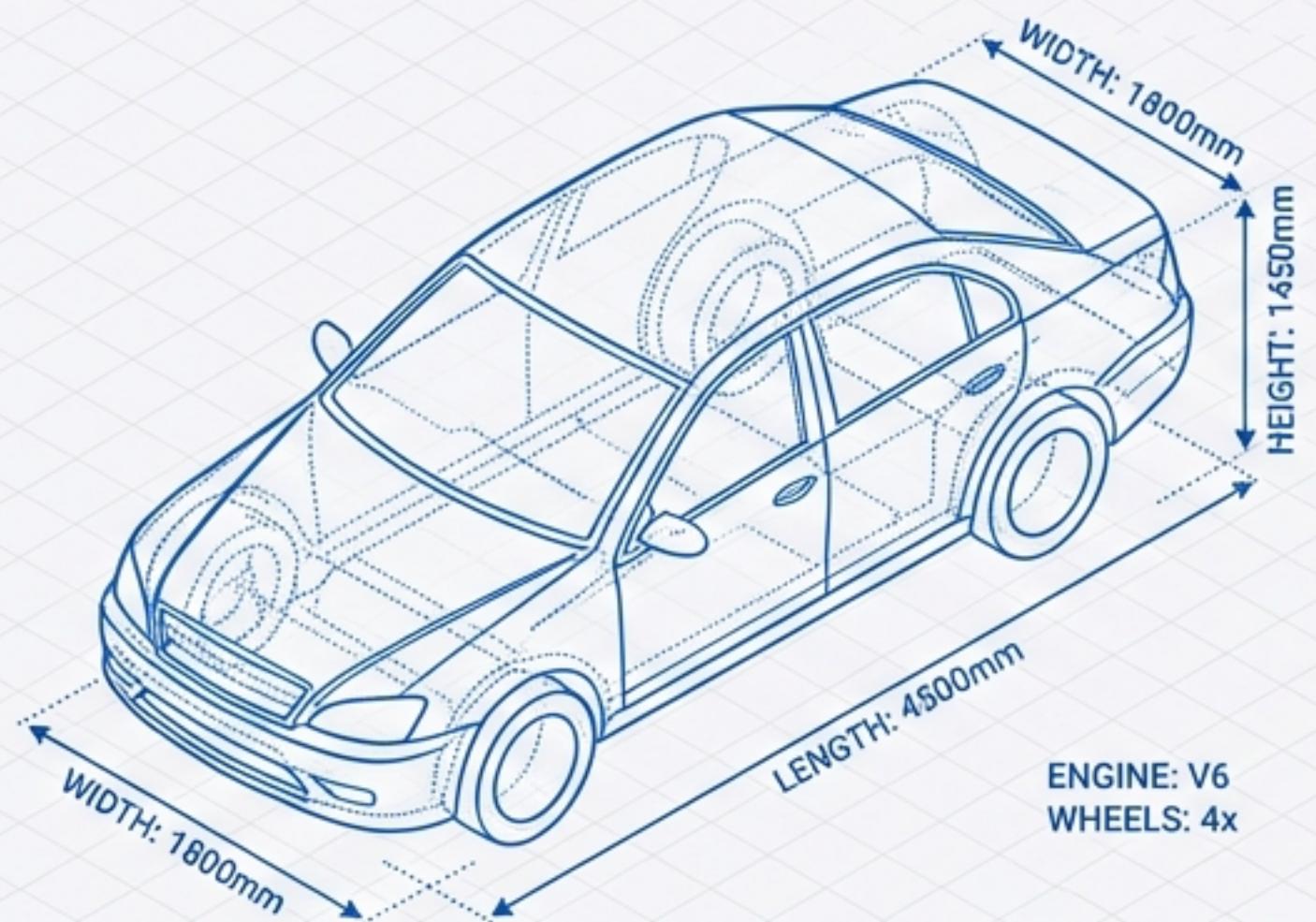
## La Solución: El Diagrama de Clases

Es la herramienta principal para definir la estructura estática del sistema antes de escribir una sola línea de código.

## Beneficios Clave

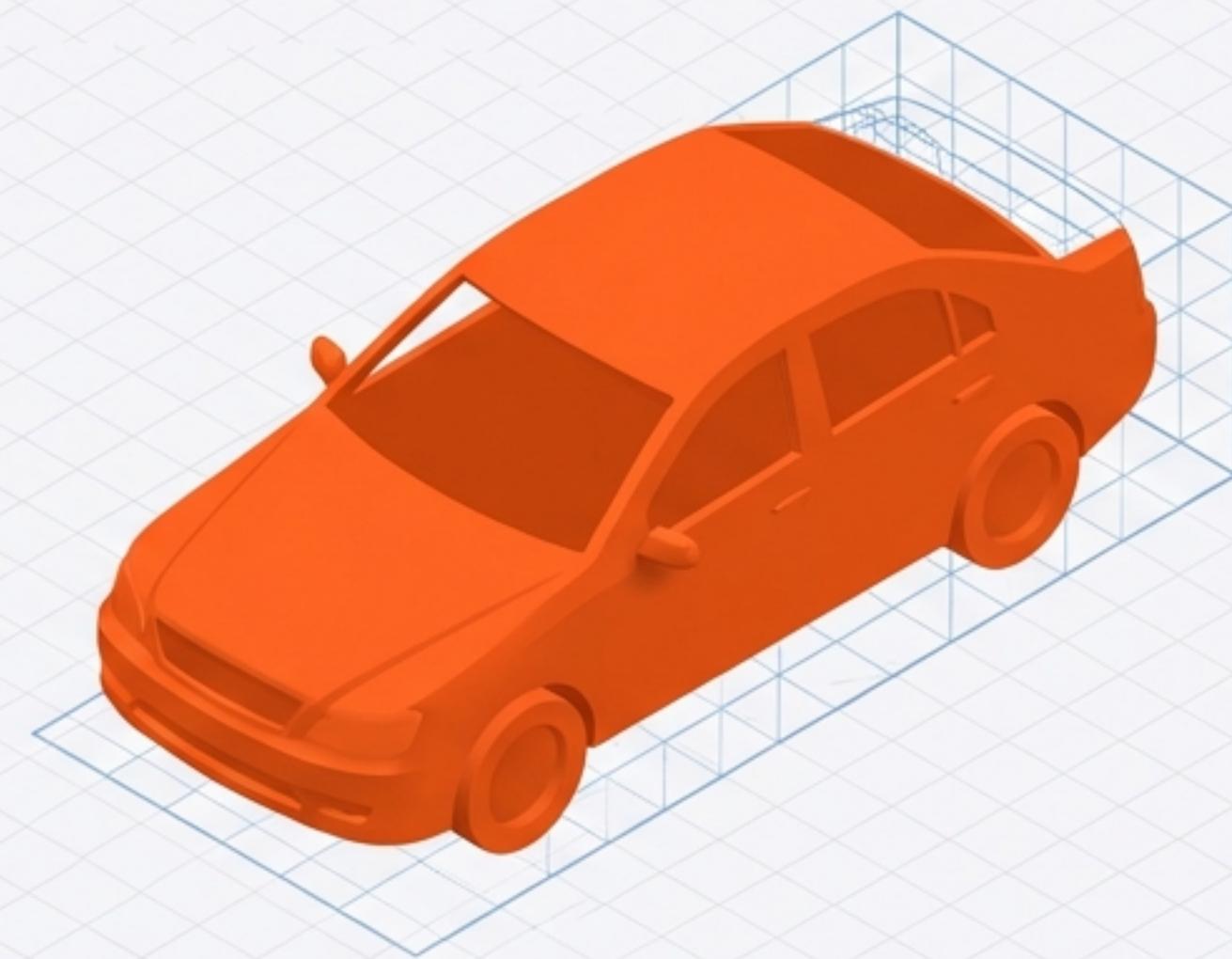
- Visualización clara de la estructura del sistema.
- Facilitación de la incorporación de nuevos programadores.
- Posibilidad de generación automática de código.

# La Clase (El Molde)



Abstracción: Define atributos y métodos.

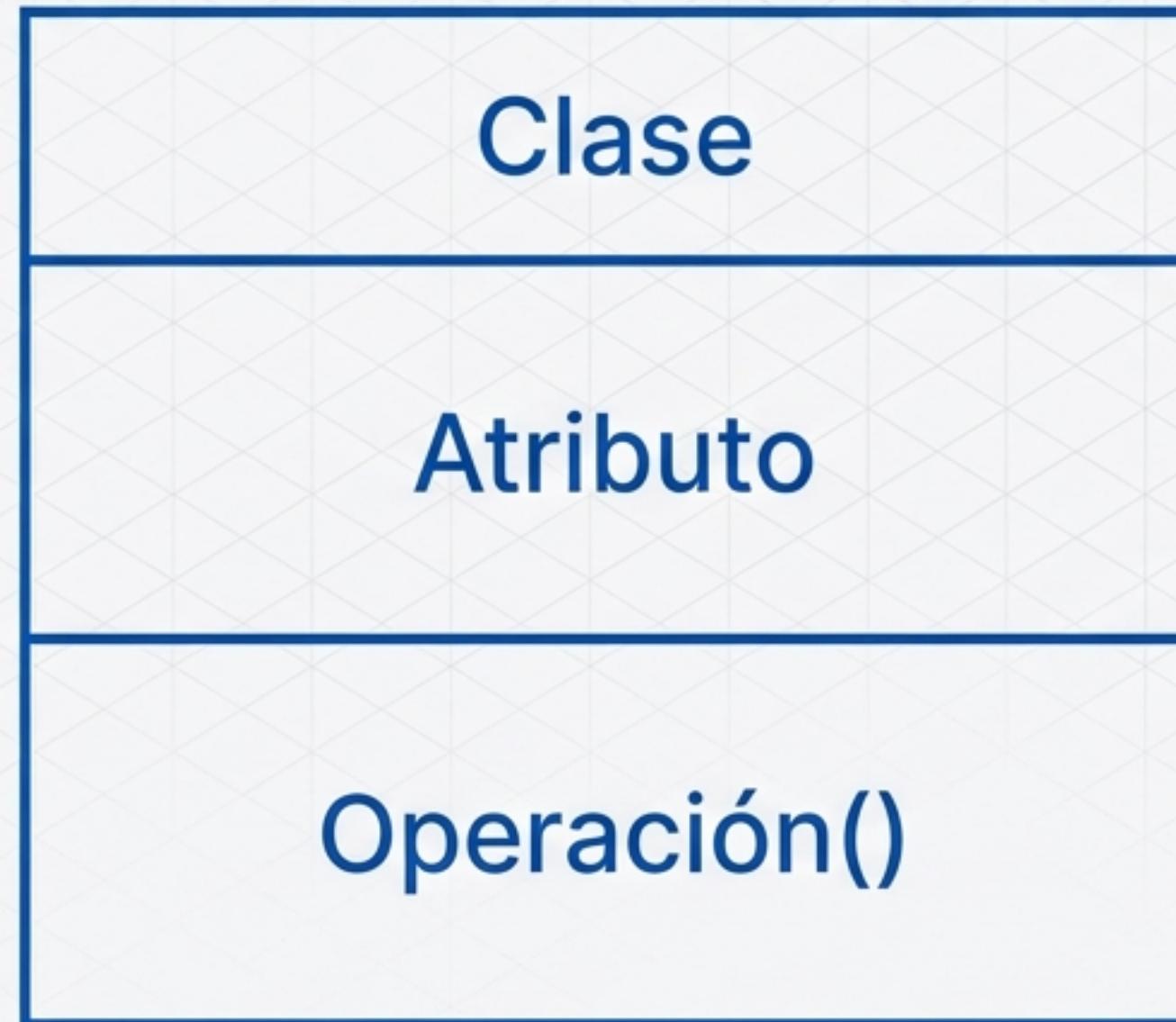
# El Objeto (La Instancia)



Ejecución: Elemento vivo en tiempo real.

```
↓ Clase  
Coche miCoche = new Coche();  
↑  
    ↓ Objeto
```

# Anatomía de una Clase y Encapsulamiento



## Principio de Ocultamiento

El programador debe poder usar la clase sin conocer los detalles complejos de su implementación interna.

## Niveles de Visibilidad

- + **Público**: Visible para todos.
- # **Protegido**: Visible para clases derivadas.
- **Privado**: Visible solo dentro de la clase (Encapsulación).

# La Matriz de Relaciones



## Asociación

Relación estructural básica.



## Composición

Relación Todo-Parte fuerte. Una clase compone a otra.



## Dependencia

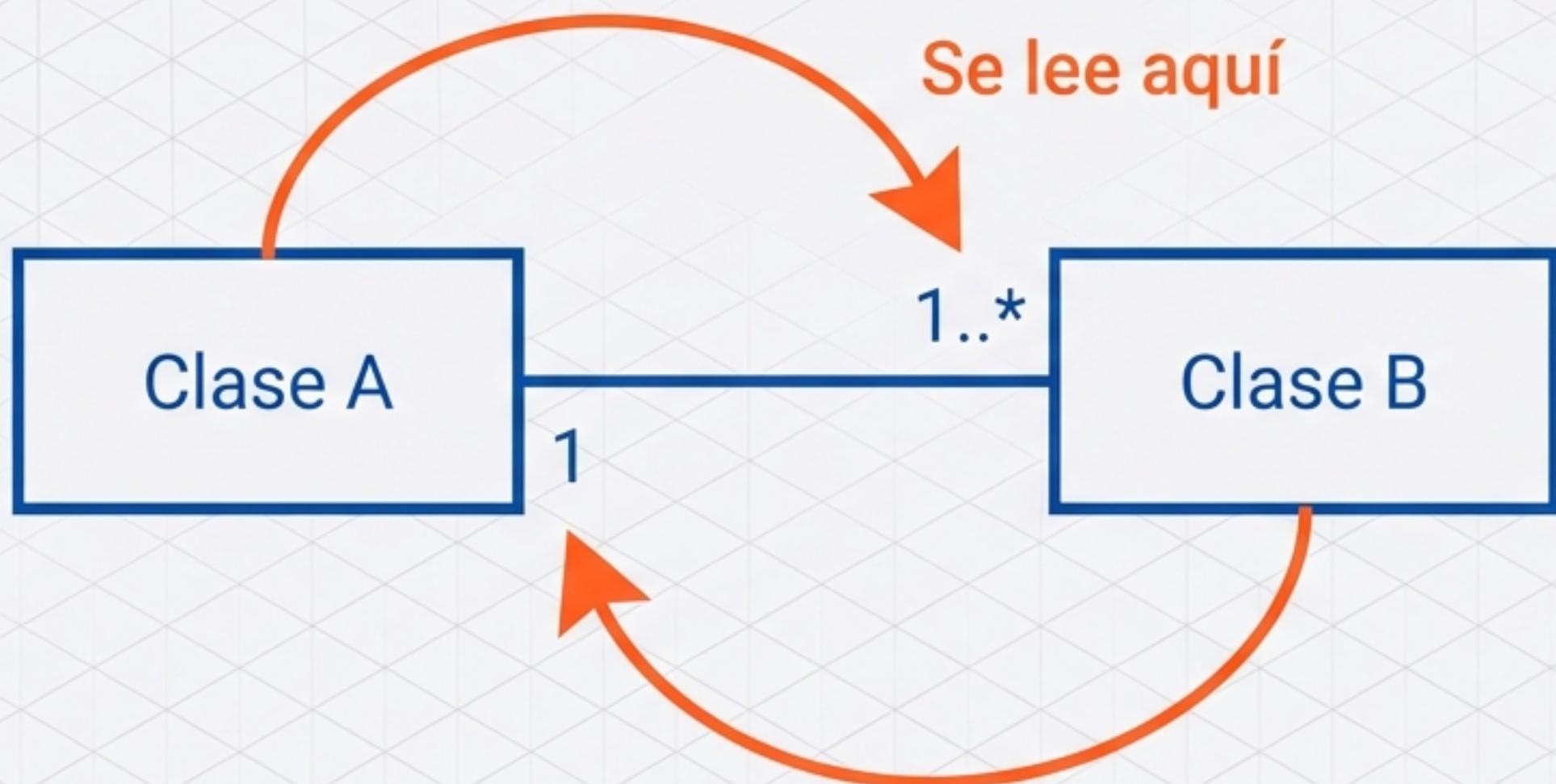
Relación "Usa a". Temporal y frágil.



## Herencia

Jerarquía y especialización.

# Sintaxis y Cardinalidad



## Cardinalidad

- 1 : Uno estricto
- 0..1 : Cero o uno
- 1..\* : Uno a muchos
- \* : Indeterminado (Muchos)

## Regla de Lectura Cruzada

Para definir la relación de A hacia B, se lee el número situado junto a B.

# Ingeniería Directa (Forward Engineering)



UML Class  
Diagram file

IDE  
(Computer/Chip)

Java Code  
(Brackets)

## Flujo Ideal

Diseño primero,  
implementación  
después.

## Automatización

Generación de  
esqueletos de código  
(Boilerplate).

## Limitación

Crea la estructura, pero  
**NO la lógica interna** de  
los métodos.

# Ingeniería Inversa

DIN Pro Bold

Código Binario

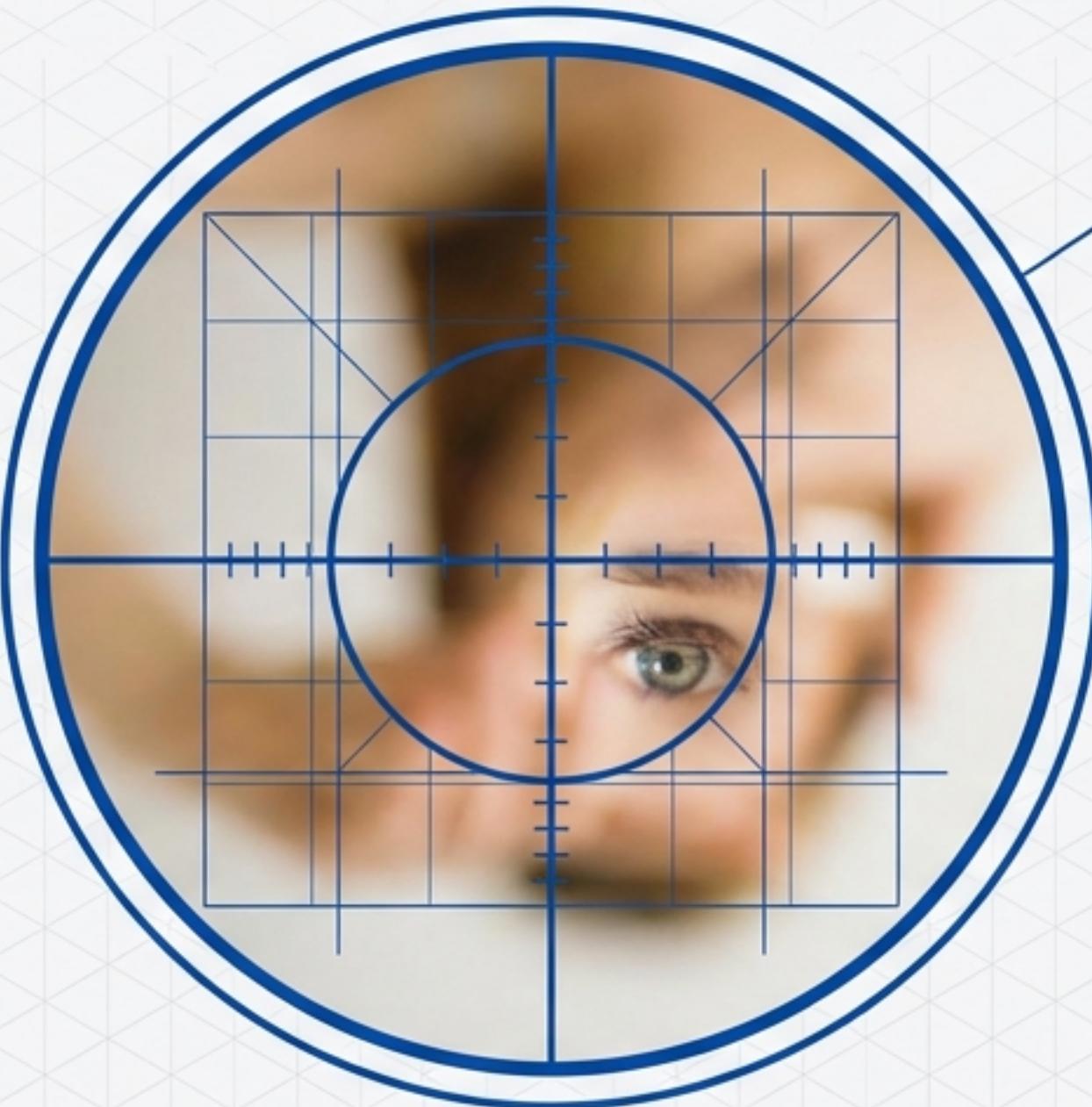
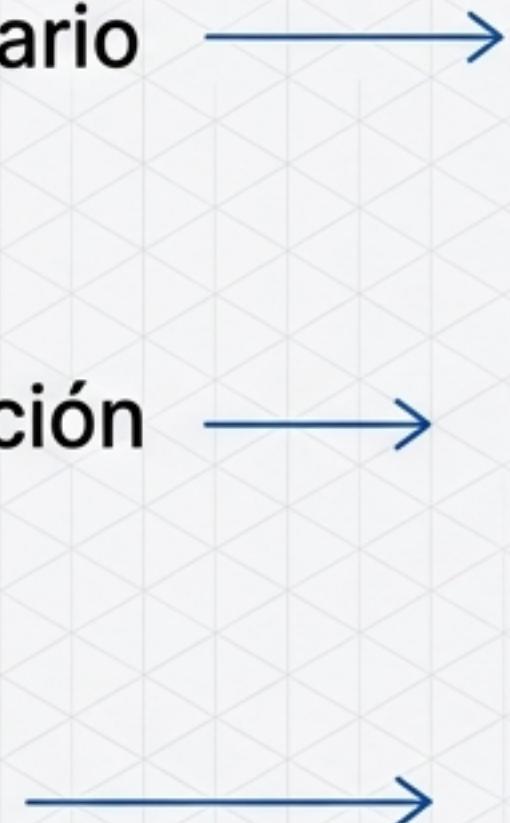
DIN Pro Medium

Decompilación

Inter Regular

Diagrama

Inter Light

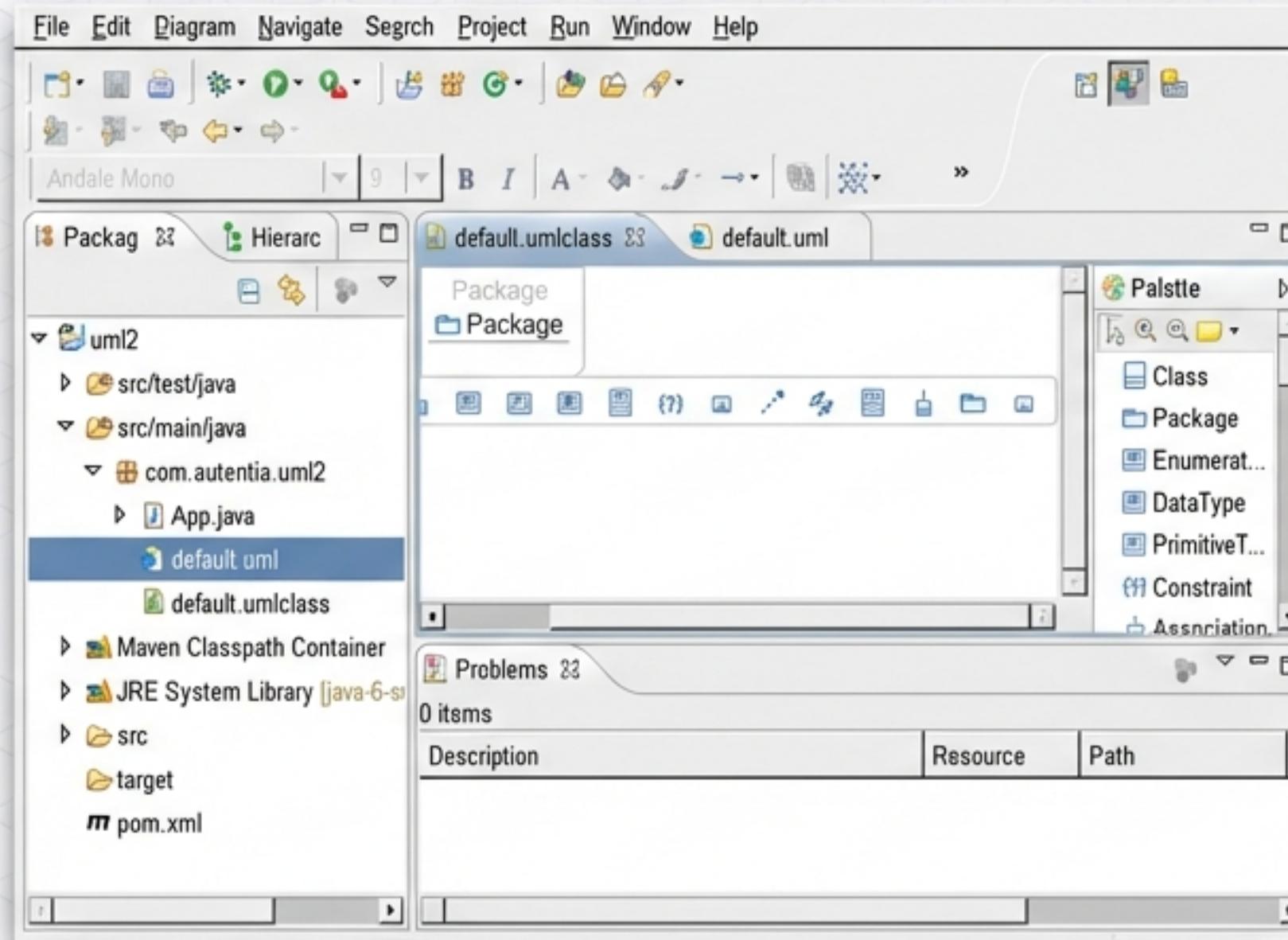


## Uso y Riesgos

DIN Pro Medium

- Uso Legítimo: Recuperación de documentación en sistemas Legacy.
- **Decompilación:** Sencilla en Java (bytecode), compleja en C++.
- **Contramedida:** La Ofuscación de código protege la propiedad intelectual.

# Herramientas CASE



**Computer Aided Software Engineering**

**Microsoft Visio**

Propósito general. Flexible pero no específico de ingeniería.

**Rational Rose**

Estándar profesional. Estricto cumplimiento del Proceso Unificado.

**StarUML / Plugins IDE**

Integración directa en Eclipse/VS. Ideal para ingeniería inversa y generación de código.

# Caso Práctico 1: El Desafío del Código Legacy

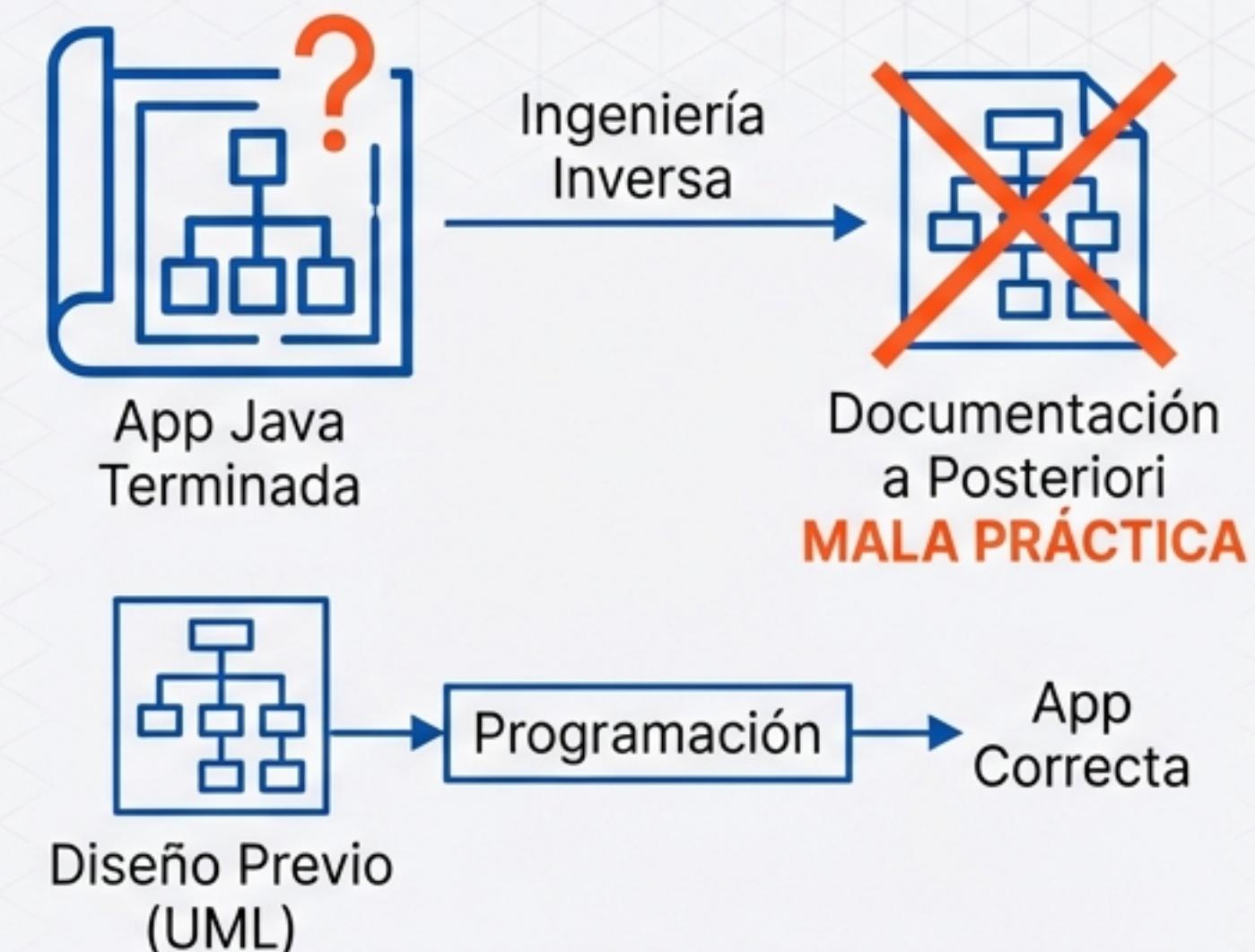


**Situación:** Un cliente exige documentación de una App Java ya terminada.

**El Dilema:** ¿Es válido generar el diagrama a posteriori?

**Veredicto del Experto:** Es técnicamente posible mediante ingeniería inversa, pero es una **MALA PRÁCTICA** como metodología habitual.

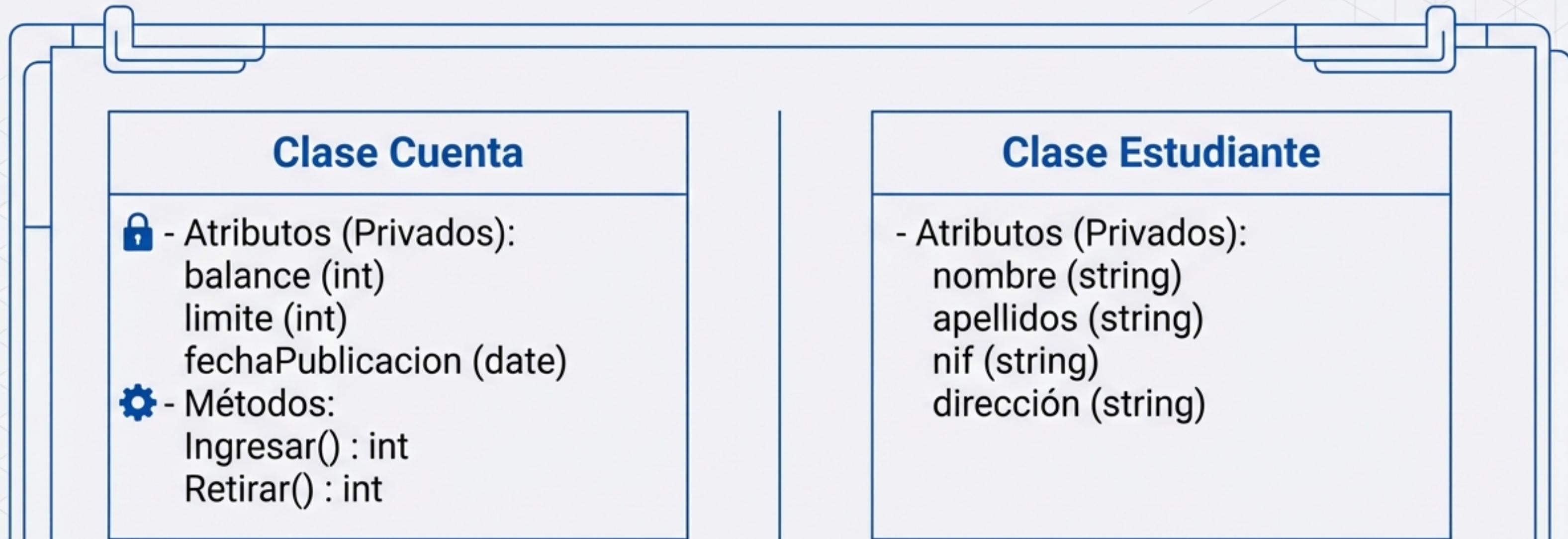
**Lección:** UML es para diseñar y resolver problemas antes de programar. Si solo documentas al final, estás **usando mal la herramienta**.



**Lección:** Diseño antes que Código.  
No después.

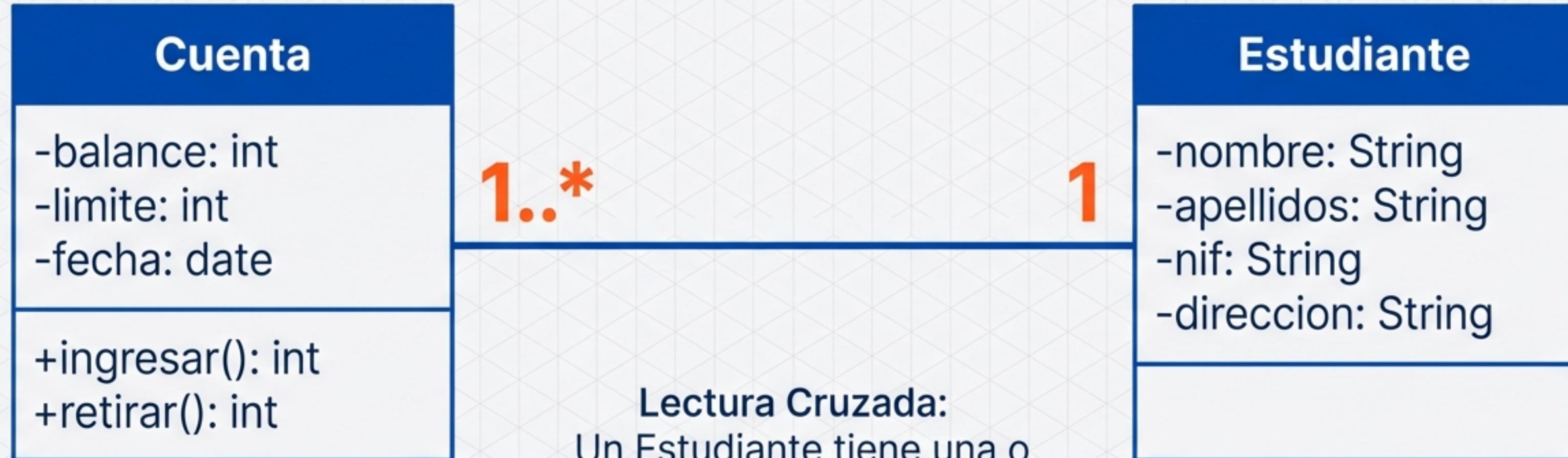
# Caso Práctico 2: Especificación de Requisitos

Objetivo: Modelar relación Estudiante - Cuenta



¿Qué tipo de relación y cardinalidad une a estas entidades?

# Caso Práctico 2: La Solución



**Lectura Cruzada:**  
Un Estudiante tiene una o  
muchas (1..\*) Cuentas.

# Refuerzo Visual: Contexto Universitario



# Conclusiones del Arquitecto



## Visión Global

El diagrama de clases es vital para visualizar la complejidad.



## El Momento Correcto

El diseño precede al código. La ingeniería inversa es solo análisis.



## Independencia

Un buen diseño UML perdura más allá del lenguaje (Java, C++, Python).



## Herramientas

IDEs modernos agilizan el paso del dibujo al esqueleto de código.