2 Layered Dependencies

Comportamientos que cambian la arquitectura: Interfaces e Inversión de Dependencias

1. CONNECT

- ¿Cuántas dependencias tiene un objeto ?
- ¿Cuántos dependientes tiene un objeto ?

> Objetivo: entender la necesidad de control de las dependencias.

2. CONCEPT

Principios clave:

- **Interface Segregation (ISP)**: composición de comportamientos.
- **Dependency Inversion (DIP)**: el código de alto nivel no depende de detalles.

Ideas fundamentales:

- El flujo de dependencias va de arriba hacia abajo.
- Depender de abstracciones hace el código más flexible y testeable.

3. CONCRETE PRACTICE

Partimos de un diseño en capas con responsabilidades claras.

Flexibilizar las **dependencias** mediante interfaces y factorías.

- Las capas de infraestructura y negocio ofrecen un comportamiento.
  - [ ] Definir interfaces para los comportamientos.
  - [ ] Implementar los comportamientos.
- Ofrecer factorías para evitar instancias concretas.
  - [ ] Definir factorías para los comportamientos.
  - [ ] Invocar factorías en el código de alto nivel.
  - [ ] Depender de interfaces en el código de alto nivel.
- No es necesario ningún sistema de inyección de dependencias.

> Objetivo: patrón factoría y familiaridad con interfaces.

4. CONCLUSIONS

- Las dependencias genera acoplamiento.
- Las abstracciones permiten flexibilidad.
- ¿Qué dependencia concreta en tu proyecto sería la primera candidata a flexibilizar mañana?