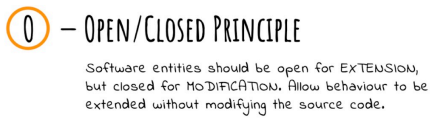
23 patrones

S.O.L.I.D
¤ Single Responsability Principle
¤ Open-Closed Principle
¤ Liskov Substitution Principle
¤ Interface Segregation Principle
¤ Dependency Inversion Principle



Nombres:
-Autoexplicativos
-Evitar desinformación -> no incluir tipos

funciones:
Menos es más, lo más pequeñas posibles.
La menor cantidad posible de argumentos.

Why am I so down on comments? Because they lie. Not always, and not intentionally, but too often. The older a comment is, and the farther away it is from the code it describes, the more likely it is to be just plain wrong. The reason is simple. Programmers can't realis- tically maintain them

Making the name of a function that is the same as the comment

Ideas en vez de comentar código:

- Generar documentación incrustada mediante, por ejemplo javadoc, lo que genera páginas de html que describen las interfaces, constructores, métodos, etc… de forma entendible para el resto del equipo pero aún así fácil de implementar

También el mismo uso de pruebas unitarias o de integración permiten comprender el comportamiento esperado del código al permitir hacer pruebas y uso real de código entendible.

Clases:

A través de funciones que modelen el flujo de código

Paradigma imperativo

Programación estructurada: La programación estructurada es un tipo de programación imperativa donde el flujo de control se define mediante bucles anidados, condicionales y subrutinas, en lugar de a través de GOTO.
Programación procedimental: Este paradigma de programación consiste en basarse en un número muy bajo de expresiones repetidas, englobarlas todas en un procedimiento o función y llamarlo cada vez que tenga que ejecutarse.
Programación modular: consiste en dividir un programa en módulos o subprogramas con el fin de hacerlo más manejable y legible. Se trata de una evolución de la programación estructurada para resolver problemas de programación más complejos.

Se puede lograr algo similar mediante modulos

Paradigma declarativo

Introducción a los patrones
Efectividad -> rapidez y acierto en la diagnosis, identificación y resolución de tales problemas.
Mejor ingeniero -> reutiliza la misma solución matizada.

El Catálogo GOF

Constituido por 23 patrones de Diseño

- Abstract Factory
- Builder
- Factory Method
- Prototype
- Singleton
- Adapter
- Bridge
- Composite
- Decorator
- Facade
- Flyweight
- Proxy
- Chain of Responsibility
- Command
- Interpreter
- Iterator
- Mediator
- Memento
- Observer
- State
- Strategy
- Template Method
- Visitor

Documentación de los patrones:
-> estructuras: descripción gráfica de los comportamientos, acciones y relaciones de los objetos que participan en el patrón. **Diagramas de clases y trazas de eventos.**
-> participantes: diccionario de las partes (clases) que componen los patrones.
-> Colaboraciones: Diccionario de las relaciones e interacciones entre los participantes en un patrón. Cómo colaboraran los participantes.
-> Consecuencias: Detalle de los posibles **beneficios y perjuicios** que pueden derivarse del uso del patrón.
-> implementación: Detalles de las posibles implementaciones y catálogos de las decisiones de diseño en la codificación de soluciones concretas basadas en el patrón, Técnica

-> El libro GoF

Tener cuidado con que los patrones no son aptos para todo, y lo importante es descubrir CUANDO SÍ usarlo. Si los utilizas mal el software queda de poca calidad.