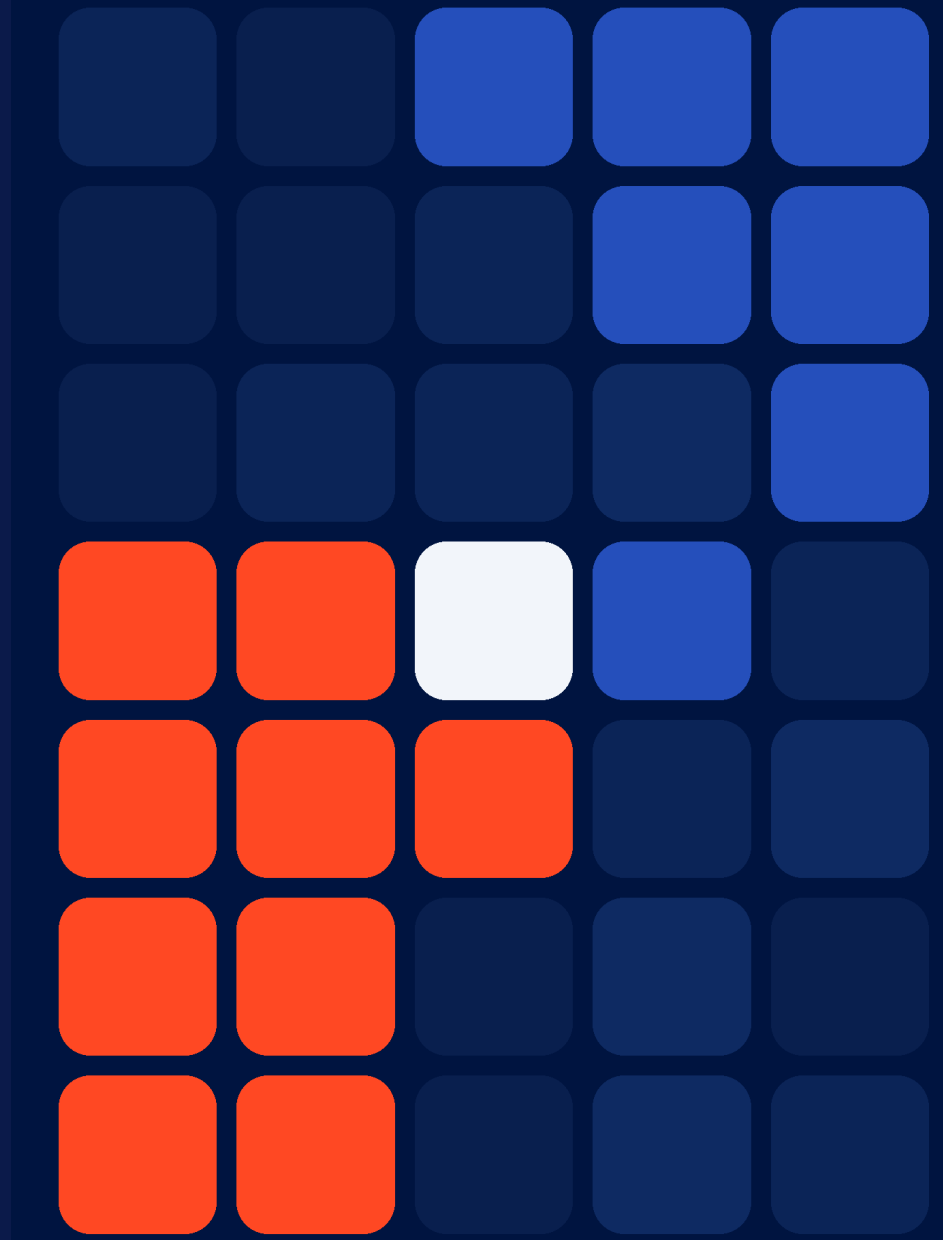
Soltein

ACADEMIA SOLTEIN · INTRODUCCIÓN A LA AE, AS Y LPS

Guía de estudio: patrones GRASP

General Responsibility Assignment Software Patterns.



Lección: Guía de estudio sobre los patrones GRASP · Elaborado por MsC René Lazo (2024-10-10)

Odoo ERP · Suite Soltein

Contenido de la guía

01 Por qué son importantes

02 Information Expert · Creator · Controller

03 Low Coupling · High Cohesion

04 Polymorphism · Pure Fabrication

05 Indirection · Protected Variations

06 Conclusiones

¿Qué son los patrones GRASP?

“Principios conceptuales que ayudan a determinar qué clase u objeto, en un diseño orientado a objetos, debe asumir qué responsabilidades.”

Introducidos por Craig Larman

GRASP frente a los patrones GoF

GRASP

- Lineamientos que guían las decisiones tempranas de diseño.
- Conducen a sistemas flexibles, mantenibles y escalables.

GoF (Banda de los Cuatro)

- Patrones de diseño para implementación directa.
- GRASP no son de este tipo.

Asignar bien las responsabilidades evita diseños frágiles, rígidos o demasiado acoplados.

1. Information Expert (Experto en información)

Soltein

1

Idea

Asignar la tarea al objeto que posee la información necesaria.

2

Justificación

Opera sin pedir datos a otros objetos: menos complejidad y acoplamiento.

3

Ejemplo

La clase Orden, que conoce ítems, precios y cantidades, calcula su propio subtotal.

2. Creator (Creador)

Soltein

1

Idea

Determinar qué clase debe instanciar ciertos objetos.

2

Justificación

Quien necesita, tiene la información o guarda datos del objeto es buen creador; evita dispersar la creación.

3

Ejemplo

En reservas de hotel, la clase Hotel crea las Reservas porque conoce la disponibilidad.

3. Controller (Controlador)

1

Idea

Manejar los eventos del sistema con un objeto separado de la interfaz de usuario.

2

Justificación

No mezclar lógica de negocio con presentación; coordina objetos de dominio.

3

Ejemplo

ControladorPago valida la operación, actualiza la orden y registra la transacción.

4. Low Coupling (Bajo acoplamiento)

Soltein

1

Idea

Mantener las dependencias entre clases al mínimo.

2

Justificación

Un cambio en una clase afecta menos a las demás; facilita pruebas y mantenimiento.

3

Ejemplo

Orden usa una interfaz Almacenamiento (MySQL, PostgreSQL) y no conoce la base concreta.

5. High Cohesion (Alta cohesión)

Soltein

1

Idea

Mantener las responsabilidades de una clase estrechamente relacionadas.

2

Justificación

Clases que hacen cosas muy distintas se vuelven confusas y propensas a errores.

3

Ejemplo

CalculadoraImpuestos solo hace cálculos fiscales, sin datos de clientes ni pagos.

6. Polymorphism (Polimorfismo)

1

Idea

Delegar a la subclase o implementación concreta el comportamiento específico.

2

Justificación

Evita if-else y switch por tipos; extiende sin modificar código existente.

3

Ejemplo

MetodoEnvio con calcularCosto(): EnvioTerrestre, EnvioAereo y EnvioMaritimo.

7. Pure Fabrication (Fabricación pura)

Soltein

1

Idea

Crear clases artificiales que no representan algo del dominio.

2

Justificación

Mantienen alta cohesión y bajo acoplamiento cuando nada del dominio encaja.

3

Ejemplo

Logger registra eventos sin contaminar las clases de negocio.

8. Indirection (Indirecta)

1

Idea

Introducir un intermediario entre dos componentes para desacoplarlos.

2

Justificación

Un mediador, adaptador o fachada protege al sistema de cambios en un componente.

3

Ejemplo

AdaptadorServicioPagos absorbe el cambio en la interfaz del servicio de pagos externo.

9. Protected Variations (Variaciones protegidas)

Soltein

1

Idea

Identificar lo que probablemente cambie y encapsularlo tras interfaces estables.

2

Justificación

Si cambia la tecnología o el proveedor, el resto del sistema apenas se impacta.

3

Ejemplo

Interfaz ProcesadorDePago con PayPalProcessor y StripeProcessor.

Conclusiones: lo que aportan los GRASP

- Orientan la asignación de responsabilidades desde el diseño orientado a objetos.
- Fomentan bajo acoplamiento y alta cohesión, claves de la mantenibilidad.
- Promueven claridad, extensibilidad y capacidad de responder al cambio.
- Son marco conceptual al inicio del diseño, antes de los GoF o de los detalles técnicos.

Criterio, no recetas

Son criterios de buen juicio, no recetas exactas; reducen costos de mantenimiento y suben la calidad.

Lo que debes recordar

Responsabilidades

GRASP guía qué clase asume cada responsabilidad.

Nueve patrones

Expert, Creator, Controller, Low Coupling, High Cohesion, Polymorphism, Pure Fabrication, Indirection y Protected Variations.

Bajo acoplamiento

Con alta cohesión es la base de la mantenibilidad.

Antes de GoF

Marco conceptual para el inicio del diseño.

Soltein · Una empresa que evoluciona · soltein.mx · cursosodoo.soltein.net

Fuentes y pendientes por validar

FUENTES

· Guía de estudio doc-guia-estudio-patrones-graps-2024-12-v1.0 (Drive), MsC René Lazo.
· Craig Larman (autor de los patrones GRASP, citado en el original).

REQUIERE VALIDACIÓN

· El índice y el título de sección enumeran 8 patrones, pero el cuerpo desarrolla 9 (se añade Protected Variations).
· Los ejemplos de código (pseudo-código Java) del original no se incluyen.
· Fecha 2024-12 en el nombre del archivo y 2024-10-10 en la portada.