Patrones de Software para la Asignación de Responsabilidades

Unidad 2

Patrones de Diseño de Software

Patrones

- Un patrón es una descripción de un problema y cómo resolverlo
  - Descripción
  - Escenario
  - Solución
  - Consecuencias

Patrones de Responsabilidad

- GRASP
  - *General Responsability Assignment Software Patterns*
  - Patrones de Diseño de Asignación General de Responsabilidades
  - Asigna una responsabilidad a cada clase

GRASP y Patrones de Diseño

- GRASP no es una competencia a otros patrones existentes
- Sirven para ayudar a elegir qué patrón de diseño es el más adecuado

Patrones GRASP

- Los patrones que ofrece GRASP:
  - Alta Cohesión
  - Bajo Acoplamiento
  - Experto en información
  - Creador
  - Controlador
  - Polimorfismo
  - Fabricación Pura
  - Indirección
  - Variaciones Protegidas

Cohesión

- Es la medida en que un componente de un sistema realiza la o las tareas para las que fue creado y delega otras tareas a otros componentes
  - Una clase es responsable de hacer lo que respecta a su entidad y no a otras clases
- Se busca una **Alta Cohesión**

Alta Cohesión

- Entre más específico sea el propósito de una clase, más alta será su cohesión
- Es importante que el contenido de una clase está relacionado con su nombre
- A nivel de clase, los métodos de una clase trabajan con datos similares
- A nivel de paquetes, las clases que lo forman, están relacionadas entre ellas

Acoplamiento

- Indica la fuerza con la que está relacionado un componente con otros, y cómo impactan los cambios que se le realicen
  - Si se modifica una clase, se tienen que modificar los atributos de otra clase
- Se busca tener un **Bajo Acoplamiento**

Bajo Acoplamiento

- Entre menos sepa una clase A de una clase B, menor será el Acoplamiento
- Entre menos sepa A de cómo se implementa B, menor será su Acoplamiento
- La comunicación debe ser a través de los métodos get() y set()

Experto en Información

- Asigna responsabilidades a un objeto
- Se le asignará una responsabilidad a una clase dependiendo la información de la que dispone
- Los objetos deben hacer cosas relacionadas con la información que poseen
- Responsabilidades:
  - Implementación de un método
  - Creación de un objeto

Creador

- Se refiere a la responsabilidad de crear objetos de otra clase
- Una clase A creará objetos de B cuando:
  - A es una composición de B
  - A almacena a B
  - A tiene los datos de inicialización de A
  - A usa a B

El Patrón Creador

Patrón Creador
Problema: Determinar quién debe crear instancias de alguna clase
Solución: A creará objetos de una clase B si: •A es una composición de B •A almacena a B •A tiene los datos de inicialización de A •A usa a B

El Patrón Experto

Patrón Experto
Problema: Determinar las responsabilidades a asignar a cada clase
Solución: Asignar la responsabilidad a la clase que tiene la información necesaria

Controlador

- Responsable de manejar los evento de un sistema
- Evento: Evento generado por una acción externa del sistema

El Patrón Controlador

- Se asignará el control del flujo del programa a clases muy específicas
- Se delegan las actividades a otras clases
- Esta asignación debe promover la Alta Cohesión
- Un solo controlador genera un Alto Acoplamiento con otras clases
- Sugiere dividir el número de eventos en la mayor cantidad de controladores

Polimorfismo

- Cuando una actividad (funcionalidad) dependa del tipo de objeto, se usará el polimorfismo
- Dar el mismo nombre a métodos diferentes de cada objeto

El Patrón Fabricación Pura

Patrón Fabricación Pura
Problema: Asignar responsabilidades cuando se violan los principios de Alta Cohesión y Bajo Acoplamiento
Solución: Asignar la responsabilidad a un objeto de una clase que no represente un objeto del modelo de dominio del problema

Fabricación Pura

- Son clases que no representan un objeto del dominio del problema
- Su objetivo es reducir el acoplamiento y facilitar la reutilización de código
- Surge cuando hay clases con baja cohesión y se busca separar los métodos que no tienen mucha relación entre ellos
- El abuso en la Fabricación pura, puede llevar a clases con un solo método

El Patrón Indirección

Patrón Indirección
Problema: Reducir el acoplamiento y mantener la reutilización
Solución: Asignar la responsabilidad a un objeto intermedio que sirva de mediador entre otros componentes o servicios

El Patrón Variaciones Protegidas

Patrón Variaciones Protegidas
Problema: Cómo asignar responsabilidades de tal forma que que futuras modificaciones tengan un menor impacto
Solución: Asignar responsabilidades para crear interfaces estables

Variaciones Protegidas

- Todo lo que pueda sufrir modificaciones en un diseño, se debe manejar a través del polimorfismo e interfaces para facilitar nuevas implementaciones

GRASP y Otros Patrones

- Los patrones GRASP en ocasiones no son considerados como Patrones de Diseño
- Algunos los consideran como buenas práticas que ayudan a decidir qué otro tipo de patrones se utilizarán