Introducción a los Patrones de Diseño de Software

Unidad 1

Patrones de Diseño de Software

Introducción

Diseño Orientado a Objetos

- El objetivo del Diseño Orientado a Objetos es resolver un problema
- No se debe resolver cada problema partiendo desde cero
- Es necesario reutilizar soluciones que ya han demostrado resolver un problema

Definición

Un patrón de diseño es una descripción de clases y objetos que se comunican entre sí para resolver un problema de diseño general en un contexto particular

Elementos de un Patrón

De manera general se encuentra:

- Nombre del Patrón
- El problema que resuelve
- La solución
- Consecuencias

Problema

- Se presenta el tipo de problema que se resuelve
- Explica el problema y su contexto
- Puede incluir las condiciones que deben darse para aplicar el patrón

Solución

- Describe que elemento forman parte del diseño
- La manera en que se relacionan
- Sus responsabilidades y cómo colaboran

Consecuencias

- Resultados a obtener tras aplicar el patrón
- Ventajas y desventajas de aplicarlo
  - Tiempo de desarrollo
  - Portabilidad
  - Flexibilidad del código

El Patrón MVC

- Modelo / Vista / Controlador
- Aplicado en el desarrollo de sistemas con interfaces gráficas
- Formado por tres tipos de objetos:
  - Modelo. Es el modelo de la aplicación
  - Vista. Es la parte gráfica
  - Controlador. Cómo reacciona la interfaz a la entrada

Gang of Four (GoF)

- Se trata de cuatro autores que iniciaron con el concepto de patrones de diseño en el desarrollo de software
- Principios en los que se basan los patrones de diseño:
  - Programar para una interfaz, no una implementación
  - Preferir composiciones de objeto sobre herencia
- Este grupo desarrolló 23 patrones estándar, conocidos como los patrones GoF

Catálogo de Patrones de Diseño

- Los patrones se clasifican en base a dos criterios:
  - Propósito. ¿Qué hace un patrón?
  - Ámbito. Afecta a clases u objetos
- Propósito:
  - Creación. Creación de objetos
  - Estructural. Composición de clases y objetos
  - Comportamiento. Responsabilidades

Los más utilizados

- Los patrones más utilizados son:
  - Abstract Factory
  - Adapter
  - Composite
  - Decorator
  - Factory Method
  - Observer
  - Strategy
  - Template Method

Diseñar para el Cambio

El Paradigma Orientado a Objetos

- Una aplicación en el paradigma OO, está formada por objetos
- Un objeto encapsula datos y procedimientos (métodos)
- Un objeto recibe una petición y realiza una operación
- Con una operación, se cambian los datos de un objeto
- Lo más complejo es descomponer un sistema en objetos
- Se programa para un problema no en términos de re utilización

Diseñar para el Cambio

- El objetivo de los patrones es facilitar la re utilización de código
- Para poder reutilizarlo, se debe pensar en los nuevos requisitos
  - Re definición de funciones
  - Re implementación de clases
- Los patrones de diseño ayudan a asegurar que el sistema cambie de forma concreta

Importancia de los Patrones

- Los patrones ayudan a identificar objetos que no proceden del análisis
- Ayudan a identificar la granularidad de los objetos
- Ayudan a especificar las interfaces de los objetos
  - Una interfaz ayuda a saber lo que hace un objeto

Ventaja de Utilizar Patrones

- Permite que una solución sea sencilla de reutilizar, extender y mantener
- Aplicar patrones significa programar para resolver problemas y que automáticamente se convierten en buenas técnicas de diseño

Ejemplo Sencillo

- Considerar una aplicación web en dónde hay navegación entre todos sus elementos
- La funcionalidad para saltar entre cada página y cómo mostrar la siguiente sería muy repetitiva

Solución con Patrones

- Existe un patrón que indica la creación de un elemento intermedio que se encargue de todas las peticiones.

Cambio de Objetos

- Suponer que una aplicación a trabajado con cierto tipo de objetos
- En una actualización, ese tipo ha cambiado y ahora debe trabajar con nuevos
- Sería muy tardado modificar todo el código para que se adapte a este nuevo requerimiento

¿Qué propone un patrón?

- ¿Y si un patrón propusiera un bloque que adaptara el objeto viejo a uno nuevo?

Patrones y Orientación y Objetos

Patrones y Objetos

- La re utilización de ideas es el objetivo principal de los Patrones de Diseño
- En particular en el diseño de aplicaciones Orientadas a Objetos

Patrones y Objetos

- Dados los cuatro aspectos básicos de la POO
  - Abstracción
  - Encapsulado
  - Polimorfismo
  - Herencia
- Los patrones de diseño están involucrados con cada uno de ellos

Abstracción

- La abstracción se relaciona en como se entiende un problema antes del diseño OO
- El entender como se va dividiendo un problema en segmentos, es parte vital para una representación en objetos
- Los patrones presentan como atacar un problema muchas veces a nivel abstracto

Encapsulado

- Es cuando se “agrupan” (encapsulan) los datos y operaciones de un objeto
- Los patrones de diseño hacen mucha énfasis en el encapsulado de acciones o datos

Polimorfismo

- Desarrollar código que pueda trabajar con distintos tipos de objetos en tiempo de ejecución
- En muchos de los patrones de diseño, el polimorfismo juega un papel de gran importancia

Herencia

- Una clase puede heredar sus datos y operaciones a otras clases
- Los patrones de diseño en general tienden a usar Composición en vez de Herencia

Principios de Diseño

Principio de Diseño 1

- Se deben identificar los elementos de una aplicación que varían y separarlos de los que permanecen igual
- Esto ayuda a que las partes que varían se modifiquen sin alterar a las que no cambian

Principio de Diseño 2

- Programar orientado a una interfaz, no hacia una implementación
- Los objetos realizarán una implementación concreta o específica a partir de un super tipo, preferentemente una interfaz o clase abstracta

Principio de Diseño 3

- Considerar la Composición en lugar de la Herencia
- Una Composición “tiene un” es preferible a una Herencia “es un” ya que permite cambiar el comportamiento al momento de la ejecución

Principio de Diseño 4

- Las clases deben ser extendidas y no modificadas
- A una clase se le agregará funcionalidad en base a extensiones y no a modificaciones de su código

Principio de Diseño 5

- Se debe buscar el bajo acoplamiento entre objetos
- Se buscará reducir el uso de relaciones que aumenten la relación de un componente con otros
  - Herencia
  - Composición
  - Asociación

Principio de Diseño 6

- Depender de abstracciones, no de clases concretas
- Se debe preferenciar el uso de clases abstractas o interfaces que sean implementadas por otras clases