Exploración de los Patrones de Diseño en el Desarrollo de Software

Índice

1. ¿Cómo definirías un patrón de diseño en el contexto del desarrollo de software?

Son moldes que ya existen de modelos de desarrollo y que ya te hagas una idea, como ya están echos de antes, por esa parte es fácil que ya encajen buenamente con cualquier esquema general cara cualquier proyecto y te adelanten tiempo.

2. ¿Cuáles son algunas características clave de los patrones de diseño y por qué son importantes?

- **Reutilización de soluciones**: Te escupen y recomiendan posibles soluciones a un problema que puedas tener.
- **Mantenibilidad y escalabilidad**: Son fáciles de leer, lo que hace fácil mantenerlos con el tiempo.
- **Comunicación efectiva**: Tienen un tipo de gerga común, conceptos básicos que sin importar quien seas, se dicen igual en todos sitios.
- **Estandarización**: Los diseños vienen con una serie de estándares básicos de desarrollo de software, así ya se va de serie bien.
- **Flexibilidad y adaptabilidad**: Los patrones te permiten hacer cambios sin dañar colateralmente, el resto de cosas si modificas algo en otro sitio.

3. ¿Cuáles son los principales tipos de patrones de diseño y qué los diferencia?

Estos se dividen en 3 tipos, los creacionales, estructurales y de comportamiento aquí van:

- **Creacionales**: Son patrones centrados en crear objetos de manera flexible entre si.
- **Estructurales**: Se centrar en la estructura de los objetos y las clases para poder buildearlas.
- **Comportamiento**: Se centran en como se comportaran los objetos y interactuaran entre si, que tipo de responsabilidades obtendrá cada uno.

4. ¿Cuál es la importancia de los patrones de diseño en el desarrollo de software?

Entiendo que, agilizar y facilitar la vida de los desarrolladores software, que no estén tanto comiéndose la cabeza conceptualmente en lo que están haciendo y tengan un punto del partida por el que guiarse y hacer las cosas.

5. ¿Cómo se selecciona un patrón de diseño apropiado para una situación particular?

Estudiando lo que quieres hacer y plasmándolo, a groso modo.

6. ¿Cuáles son algunas consideraciones importantes al aplicar un patrón de diseño en un proyecto?

No sabría decir la verdad, pero asumo que, que no se salga dentro de lo que cabe de los moldes que te ofrece en patrón, en el aspecto de, romperlo y eso.

7. ¿Puedes proporcionar ejemplos de patrones de diseño creacionales y explicar cómo se utilizan?

- **Singleton**: Hace que una clase tenga solo una instancia y establece el punto de acceso de esa misma instancia.

- **Factory Method**: Te ofrece una interfaz con la que crear objetos y crear instancias subclases.
- **Abstract Factory**: Te ofrece una interfaz para crear múltiples objetos sin definir clases, así en el vacío.

8. ¿Cuáles son algunos ejemplos de patrones de diseño estructurales y en qué situaciones se aplican?

- **Adapter**: Permite que dos interfaces incompatibles trabajen juntas.
- **Composite**: Trata objetos individualmente y compuestos de igual manera.
- **Decorator**: Da funciones dinámicamente a los objetos.
- **Proxy**: Te ofrece control de objetos a través de representaciones (seré sincero, no entiendo del todo este)

9. ¿Qué tipo de problemas resuelven los patrones de diseño de comportamiento y cómo se implementan?

- **Observer**: Cambia o pone la dependencia de un objeto a uno a muchos.
- **Strategy**: Define una familia de algoritmos, los encapsula individualmente y los hace intercambiables. También te deja cambiar el algoritmo utilizado en tiempo de ejecución (Seré sincero, este tampoco lo entiendo del todo).
- **Command**: Te encapsula peticiones como objetos, para diferenciarlas y poder gestionar cosas.

10. ¿Puedes mencionar ejemplos prácticos de patrones de diseño que se utilicen comúnmente en la industria del desarrollo de software?

Mirando, por ejemplo el factory mod se puede usar se puede usar para crear objetos y familias de objetos con el observer por ejemplo agenciar dependencias entre objetos.

Seré sincero, tampoco indague demasiado en esto, pero como concepto se ve bien.