1. ¿Qué es un patrón de diseño?

Es una solución probada y reutilizable a problemas comunes en el diseño de software.

2. ¿Cuál es su clasificación?*

Se clasifican en tres tipos: creacionales, estructurales y comportamentales.

3.¿Cuáles son sus ventajas?*

Fomentan la reutilización de código, mejoran la flexibilidad y facilitan el mantenimiento.

4. ¿Quién inventó los patrones y cuándo?

Los patrones de diseño fueron popularizados por el "Gang of Four" (Gamma, Helm, Johnson, Vlissides) en 1994.

5. ¿Se puede usar fuera del software?

Sí, en arquitectura, negocios y otros campos donde se busca resolver problemas comunes de manera estructurada.

6. ¿Qué es un patrón creacional y cómo se clasifica?

Se encarga de la creación de objetos. Se clasifica en: Factory Method, Abstract Factory, Builder, Prototype, Singleton.

7. Patrón Factory Method:

Crea objetos sin especificar la clase exacta, delegando a subclases la decisión de qué instancia devolver.

8. Patrón Abstract Factory:

Proporciona una interfaz para crear familias de objetos relacionados sin especificar las clases concretas.

9. Patrón Builder:*

Separa la construcción de un objeto complejo de su representación, permitiendo construir objetos paso a paso.

10. Patrón Singleton:

Garantiza que una clase solo tenga una instancia y proporciona un punto de acceso global a esa instancia.

11. ¿Qué es un patrón estructural y cómo se clasifica?

Define cómo se relacionan los objetos. Se clasifica en: Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy.

12. Patrón Adapter:

Convierte la interfaz de una clase en otra que el cliente espera, facilitando la compatibilidad.

13. Patrón Bridge:

Desacopla una abstracción de su implementación, permitiendo variar ambas independientemente.

14. Patrón Composite:

Permite tratar objetos individuales y compuestos de manera uniforme.

15. Patrón Decorator:

Añade responsabilidades a los objetos de manera dinámica sin modificar su estructura.

16. *Patrón Facade:

Proporciona una interfaz simple para un conjunto de interfaces en un sistema complejo.

17. *Patrón Flyweight:

Reduce el uso de memoria compartiendo lo máximo posible entre objetos similares.

18. Patrón Proxy:

Proporciona un sustituto o marcador de posición para controlar el acceso a otro objeto.

19. ¿Qué es un patrón comportamental y cómo se clasifica?

Define la comunicación entre objetos. Se clasifica en: Chain of Responsibility, Command, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, Visitor.

20. Patrón Chain of Responsibility:

Permite que varios objetos tengan la oportunidad de manejar una solicitud, pasándola a lo largo de una cadena.

21. Patrón Command:

Encapsula una solicitud como un objeto, lo que permite parametrizar objetos para diferentes solicitudes.

22. Patrón Iterator:

Proporciona una forma de acceder a los elementos de una colección secuencialmente sin exponer su estructura interna.

23. Patrón Mediator

Define un objeto que encapsula la forma en que interactúan un conjunto de objetos, promoviendo un acoplamiento débil.

24. Patrón Memento:

Permite capturar y restaurar el estado interno de un objeto sin violar la encapsulación.

25. Patrón Observer:

Permite que un objeto notifique a otros objetos cuando su estado cambie.

26. Patrón State:

Permite que un objeto cambie su comportamiento cuando cambia su estado interno.

27. Patrón Strategy:

Permite definir una familia de algoritmos y hacer que sean intercambiables.

28. Patrón Template Method:

Define el esqueleto de un algoritmo en una operación, dejando algunos pasos a las subclases.

29. Patrón Visitor:

Permite definir nuevas operaciones sin cambiar las clases de los elementos en los que opera.

30. Patrón MVC (Modelo-Vista-Controlador):

Separa la representación de la información (Modelo) de la interfaz de usuario (Vista) y el flujo de control (Controlador).

31. Patrón DAO (Data Access Object):

Proporciona una abstracción para la persistencia de datos, separando la lógica de negocio del acceso a la base de datos.