

DISEÑO DE SISTEMAS.

① Cuál es el objetivo principal del patrón de diseño Bridge?

- A. Permitir crear objetos complejos paso a paso
- B. Separar una interfaz de su implementación para que ambas puedan evolucionar de forma independiente.
- C. Modificar el comportamiento de un objeto en tiempo de ejecución
- D. Garantizar que una clase tenga solo una instancia.

Justificación:

El patrón Bridge es un patrón estructural cuyo objetivo es evitar el enganche fuerte entre la abstracción y su implementación, permitiendo que ambas cambien o evolucionen de forma independiente.

② Cuál de los siguientes es un patrón de comportamiento?

- A. Abstract Factory
- B. Prototype.
- C. Interpreter
- D. Decorator.

Justificación:

El patrón Interpreter es un patrón de comportamiento que define una gramática para un lenguaje y un intérprete que usa dicha gramática para interpretar oraciones del lenguaje.

③ Cuál patrón se utiliza para simplificar el uso de un sistema complejo, proporcionando una interfaz unificada?

- A. Adapter
- B. Mediator
- C. Facade
- D. Composite.

Justificación:

El patrón Facade es un patrón estructural que proporciona una interfaz unificada y simple para un conjunto de interfaces en un subsistema complejo. Su propósito es ocultar la complejidad interna de un sistema y facilitar su uso para los clientes.

④ Cuál patrón permite añadir funcionalidad a un objeto en tiempo de ejecución sin alterar su clase?

- A. Observer
- B. Strategy
- C. Decorator
- D. Proxy.

Justificación:

El patrón decorador es un patrón estructural que permite añadir funcionalidades adicionales a un objeto en tiempo de ejecución de manera flexible, sin modificar su clase original ni usar herencia múltiple.

⑤ En el patrón Flyweight, ¿qué tipo de información se comparte entre objetos para optimizar recursos?

- A. Comportamientos dinámicos
- B. estados externos.
- C. estados internos inmutable (intrínsecos)
- D. Funcionalidad encapsulada.

Justificación:

El patrón Flyweight es un patrón estructural cuyo objetivo es reducir el uso de memoria compartiendo datos comunes entre muchos objetos, en lugar de duplicar esos datos en cada instancia.

⑥ ¿Cuál es la diferencia fundamental entre los patrones Adapter y Decorator?

- A. Adapter modifica la estructura, Decorator modifica la interfaz.
- B. Adapter convierte interfaces incompatibles, Decorator agrega responsabilidades.
- C. Adapter se usa solo en tiempo de ejecución
- D. Decorator requiere herencia múltiple

Justificación:

Adapter se usa para que dos clases con interfaces distintas puedan trabajar juntas y decorator se usa para agregar funcionalidades a un objeto sin modificar su clase.

⑦ ¿Para qué se usa el patrón Mediator?

- A. Para evitar la creación de múltiples instancias.
- B. Para centralizar la comunicación entre objetos y reducir el cumplimiento.
- C. Para crear objetos de manera flexible.
- D. Para encapsular el estado de un objeto.

Justificación:

El patrón Mediator se usa para que los objetos no se comuniquen directamente entre sí, sino a través de un objeto mediador central.

⑧ ¿Qué patrón encapsula una familia de algoritmos para que puedan ser intercambiables sin modificar el cliente?

- A. Visitor
- B. Strategy
- C. State
- D. Builder

Justificación:

El patrón Strategy encapsula una familia de algoritmos y permite intercambiarlos dinámicamente sin que el cliente tiene que modificarse.

① ¿Qué problema resuelve el patrón Strategy?

- A. Permite clavar objetos en carpetas en clase.
- B. Guarda el estado interno de un objeto sin romper el encapsulamiento.
- C. Establece comunicación entre múltiples objetos.
- D. Asigna responsabilidades entre objetos relacionados.

Justificación:

El patrón Strategy permite guardar y restaurar el estado interno de un objeto sin violar principio de encapsulamiento.

② ¿Qué características define el patrón Chain of Responsibility?

- A. Ejecuta operaciones en paralelo.
- B. Permite que múltiples objetos puedan procesar una solicitud sin que el emisor sepa cuál lo hará.
- C. Crea la cantidad de instancias.
- D. Crea familia de objetos relacionados.

Justificación:

El patrón Chain of Responsibility permite descomponer el emisor de una solicitud receptor.

③ ¿Cuál es el propósito del patrón Command?

- A. Encapsular una petición como un objeto para permitir su almacenamiento, ejecutar y deshacer.
- B. Delegar el control a clases hijas.
- C. Reducir el complejismo entre cliente y servicio.
- D. Sincronizar objetos en una orden.

Justificación:

El patrón Command encapsula una acción (la petición) como un objeto lo que permite ejecutar la acción, deshacerla, repetirla y guardarla.

④ ¿Qué patrón se emplea para construir un objeto complejo peso = peso + peso?

- A. Prototype
- B. Factory Method
- C. Builder
- D. Singleton

Justificación:

El patrón Builder se utiliza para construir objetos complejos peso = peso, separando la construcción del objeto de su representación final.

(3) ¿Por qué se utiliza el patrón Abstract Factory?

- A. Para crear instancias únicas.
- B. Para crear objetos complejos.
- C. Para crear familias de objetos relacionados sin especificar sus clases concretas.
- D. Para establecer dependencias entre clases.

Justificación:

El patrón Abstract Factory es un patrón creacional que proporciona una interfaz para crear familias de objetos relacionados sin especificar las clases concretas que se van a instanciar.

(4) ¿Qué patrón permite cambiar el comportamiento de un objeto sin modificar su código fuente ni usar herencia?

- A. Strategy
- B. Adapter
- C. Singleton
- D. Builder

Justificación:

El patrón Strategy permite cambiar el comportamiento de un objeto en tiempo de ejecución, ~~enfocándose en los algoritmos~~.

(5) ¿Cuál es una consecuencia negativa del uso excesivo del patrón Singleton?

- A. Aumentar la complejidad del sistema.
- B. Reducir la escalabilidad.
- C. Introducezcomplimiento global y dificulta las pruebas.
- D. Reduce el uso de memoria.

Justificación:

El patrón Singleton cuando se usa en exceso crea una dependencia global difícil de controlar, lo que complica la realización de pruebas unitarias.

(6) ¿Qué es la diferencia entre los patrones Prototype y Builder?

- A. Prototype crea objetos paso a paso; Builder los abraza.
- B. Prototype abre objetos existentes; Builder los construye paso a paso.
- C. Builder y Prototype son equivalentes.
- D. Builder solo se aplica a estructuras recursivas.

Justificación:

Prototype se utiliza para crear nuevos objetos a través de otros objetos ya existentes y Builder se usa para construir objetos complejos paso a paso.

(17) En el patrón Visitor, ¿Qué hace el visitante?

- A. Invoca métodos de comunicación entre clases.
- B. Representa una acción que se aplica a elementos de una estructura.
- C. Controla el acceso a una clase.
- D. encapsula una petición como objeto.

Justificación:

En el patrón Visitor, el visitante es un objeto que define operaciones que se aplican a elementos de una estructura de objetos.

(18) ¿Qué acción permite tratar de forma uniforme a objetos individuales y compuestos en una jerarquía?

- A. Proxy
- B. Composite
- C. Facade
- D. Bridge.

Justificación:

Composite permite manejar objetos simples y compuestos de la misma forma, facilitando operaciones o estructuras jerárquicas.

(19) ¿Qué patrón se utiliza cuando se necesita un objeto intermedio que controle el acceso a otro objeto real?

- A. Proxy
- B. Adapter
- C. Factory
- D. observer.

Justificación:

El patrón proxy crea un objeto intermedio que controla el acceso a otro objeto real.

(20) ¿Qué función cumple el patrón observer?

- A. Permite acceder objetos sin acoplarse a sus clases concretas.
- B. Define una relación de dependencia de uno a muchos para notificar cambios automáticos.
- C. Controla el acceso a un recurso.
- D. Almacena y recupera el estado de un objeto.

Justificación:

El patrón observer establece que cuando un objeto cambia su estado, todos sus observadores son notificados y actualizados.