Preguntas:

1. ¿Este patrón garantiza que una clase tenga una única instancia y proporciona un punto de acceso global a esa instancia?
2. ¿Proporciona una interfaz para crear familias de objetos relacionados o dependientes sin especificar sus clases concretas?
3. ¿Agrega funcionalidad a un objeto dinámicamente. Proporciona una alternativa flexible a la herencia para extender funcionalidades?
4. ¿Define una dependencia uno a muchos entre objetos, de modo que cuando un objeto cambia de estado, todos sus dependientes son notificados y actualizados automáticamente?
5. ¿Permite que un objeto altere su comportamiento cuando su estado interno cambia. El objeto parecerá cambiar su clase?
6. ¿Permite que un objeto altere su comportamiento cuando su estado interno cambia. El objeto parecerá cambiar su clase?
7. ¿Define una interfaz para crear un objeto, pero permite que las subclases decidan qué clase concreta instanciar?
8. ¿Permite la construcción de un objeto complejo paso a paso, se utiliza cuando el proceso de creación de un objeto es demasiado complejo o hay muchas maneras diferentes de construir un objeto?
9. ¿Permite la creación de nuevos objetos mediante la copia de un objeto existente, evitando así la creación de objetos utilizando una estructura de construcción más compleja?
10. ¿Desacopla una abstracción de su implementación, de modo que ambas puedan variar independientemente?
11. ¿Compone objetos en estructuras de árbol para representar jerarquías de parte-todo, permite a los clientes tratar objetos individuales y composiciones de objetos de manera uniforme?
12. ¿Agrega funcionalidad a un objeto dinámicamente. Proporciona una alternativa flexible a la herencia para extender funcionalidades?
13. ¿Representan los recursos de hardware, como servidores, computadoras o dispositivos móviles, en los que se ejecutan los componentes del sistema?
14. ¿Muestran las relaciones y las comunicaciones entre los nodos, como conexiones de red, enlaces físicos o protocolos de comunicación?
15. ¿Representan los archivos de software, bibliotecas, archivos de configuración u otros recursos que se despliegan en un nodo?
16. ¿Representan unidades lógicas y funcionales de software dentro del sistema. Estos pueden ser módulos, bibliotecas, clases, paquetes, etc?
17. ¿Representan las relaciones de dependencia entre los componentes. Una dependencia indica que un componente utiliza los servicios proporcionados por otro componente?
18. ¿Estas relaciones implica que un componente está compuesto por otros componentes y es responsable de su ciclo de vida, mientras que la otra implica que un componente está formado por otros componentes, pero no es responsable de su ciclo de vida?
19. ¿Qué es una red LAN y cuál es su propósito principal?
20. ¿Qué es DNS y cuál es su función en Internet?
21. ¿Qué es TCP/IP y por qué es importante en las comunicaciones de red?
22. ¿El cliente, que solicita recursos o servicios, y el servidor, que proporciona esos recursos o servicios?
23. ¿Como MVC (Modelo-Vista-Controlador), MVVM (Modelo-Vista-VistaModelo), y otros, ayudan a estructurar y organizar el código de la aplicación de manera modular y reutilizable?
24. ¿La arquitectura de software y web implica la gestión de datos utilizando sistemas de gestión de bases de datos (SGBD), que pueden ser relacionales, NoSQL, o en memoria, dependiendo de los requisitos de la aplicación?
25. Respuestas:
26. Singleton
27. Abstract Factory/Abstract
28. Decorator/Decorador
29. Observer/Observador
30. State
31. Visitor/Visitante
32. Factory Method/Factory
33. Builder
34. Prototype/Prototipo
35. Birdge
36. Composite
37. Decorator
38. Nodos
39. Conexiones/conexion
40. Artefactos/artefacto
41. Componentes
42. Dependencias/Dependencia
43. Metodo de composición y agragación/composición y agrgación
44. Conexión
45. Traducción
46. Protocolo
47. Modelo Cliente-Servidor/Ciente-Servidor
48. Patrones de Diseño/Diseño
49. Base de Datos

Juego

<https://es.educaplay.com/recursos-educativos/17833316-el_juego_del_singleton.html>