**28. Cuál de los siguientes tipos de asociación implica un menor grado de acoplamiento**

a) Uso \*\*

b) Herencia

c) Implementación de interfaces

**¿Qué patrón de lógica de negocios permite agrupar cada funcionalidad del sistema en un único método?**

a) Domain Model

b) Table Data Gateway \*\*

c) Transaption Script

**En una arquitectura en capas, ¿por qué la capa de lógica de negocio se sitúa por encima de la capa de acceso de datos?**

a) No es necesario, al ofrecer normalmente las mismas funcionalidades se puede intercambiar su orden\*\*\*

b) Para evitar que la capa de acceso a datos acceda directamente a la capa de servicios

c) Para desacoplar a las capas superiores de los detalles de acceso a la base de datos\*\*\*

**¿Qué patrón de capa de presentación es más adecuado cuando una sola interfaz presenta información de muchos objetos distintos?**

a) Model View Controller

b) Model View Presenter

c) Model View ViewModel

**Queremos desarrollar una aplicación de escritorio para gestionar colecciones de música, con la posibilidad de que otras personas puedan extender sus funcionalidades una vez publicada ¿Qué patrón arquitectural sería el más adecuado?**

a) Arquitectura orientada a eventos

b) Tuberías y filtros

c) Microkernel

**En una aplicación web, ¿en qué objetos de la capa de presentación se deben hacer transformaciones sobre los datos de entrada que afecten a varias funcionalidades(rutas) distintas?**

a) Middleware

b) Router

c) Application Controller

**¿Que patrón de lógica de negocios permite implementar funcionalidades que involucran a varias entidades distintas (tablas en la base de datos) en los métodos de una misma clase?**

a) Table Module

b) Domain model

c) Transaction Script

¿Qué arquitectura divide los sistemas en pequeños componentes independientes para favorecer la escalabilidad?

a) Microservicios

b) Arquitectura en capas

c) Tuberías y filtros

¿Qué patrón ORM sirve para evitar tener objetos duplicados en memoria al recuperar datos de la base de datos?

a) Unit of work

b) Identity map

c) Lazy load

¿Cuál es el principal propósito de los sistemas de rejillas para diseño responsive?

a) Favorecer el uso de etiquetas semánticas.

b) Facilitar el posicionamiento y organización del contenido de las páginas.

c) Evitar el uso de *media queries* para el posicionamiento

¿Cuándo una clase necesita una instancia de otra, ¿qué nombre recibe la técnica que consiste en pasarle esa instancia desde fuera en ligar de crearla dentro de la case?

a) Inversión de control

b) Inversión de dependencias

c) Inyección de dependencias

En una arquitectura en capas, ¿con qué capa no debería comunicarse nunca la capa de presentación?

a) Acceso a datos

b) Servicios

c) Lógica de negocio

¿Qué patron ORM para mapear la herencia implica que hay que realizar un join de varias tablas para recuperar todos los datos de un objeto?

a) Single table inheritance

b) Concrete table inheritance

c) Class table inheritance

¿Cuál de los siguientes tipos de acoplamiento es más débil?

a) Cuando hay una jerarquía de herencia

b) Cuando una clase implementa una interfaz

c) Cuando una clase recibe una lista de instancias de otra clase como parámetro en un método

Al usar un objeto *ServiceLocator* para inyección de dependencias…

a) El cliente solicita directamente las dependencias al ServiceLocator

b) El cliente solicita las dependencias al objeto Assembler, que las inyecta al ServiceLocator

c) El ServiceLocator inyecta las dependencias en el cliente a través de su constructor

¿Qué patrón junta en los mismos objetos la lógica de negocio con la lógica de acceso a datos?

a) Active Record

b) Data Access Gateway

c) Domain Model

¿Qué mecanismos de inyección de dependencias se pueden usar de forma conjunta?

a) Service Locator e inyección en el constructor.

b) Service Locator e inyección con métodos setter.

c) Inyección en el constructor y con método setter.

¿Qué patrones de lógica de negocio necesitan combinarse normalmente con una capa de servicios para implementar funcionalidades complejas?

a) Table Module y Domain Model

b) Transaction Script y Table Module

c) Transaction Script y Domain Model

¿Qué responsabilidad NO corresponde a la capa de presentación?

a) Mostrar el resultado de ejecutar la lógica de negocio

b) Activar funcionalidades de capas inferiores

c) Alojar funcionalidades de alto nivel que no encajan en ninguna clase del modelo de dominio

¿Cuál de las siguientes afirmaciones sobre la arquitectura en capas es FALSA?

a) La separación en capas favorece un diseño de capas individuales con menor cohesión

b) La comunicación entre capas disminuye el rendimiento de la aplicación

c) La separación den capa poco acopladas facilita el diseño de pruebas automatizadas.

GRASP es el acrónimo de patrones generales de software para la asignación de responsabilidades. Con respecto a dicha responsabilidades, ¿Cuáles de las siguientes afirmaciones es falsa?

a) Los métodos se implementan para cubrir responsabilidades.

b) Cada responsabilidad se traduce en un método que se debe asignar alguna clase software.

c) Los métodos pueden colaborar con otros métodos u objetos para cubrir una determinada responsabilidad.\*\*\*\*

¿Cuál de las siguientes afirmaciones relacionadas con la herencia es verdadera?

a) La herencia escala bien cuando aumenta el número de variaciones del sistema. \*\*\*\*\*

b) Siempre que tenemos una jerarquía de herencia, es posible sustituirla por una agregación o composición (delegación), lo cual aumenta la flexibilidad del sistema.

c) La herencia mejora el acoplamiento del sistema.

¿Cuál de los siguientes efectos se puede deber al alto acoplamiento entre clases?

a) Mayor cohesión de las clases fuertemente acopladas.

b) Mayor facilidad de reuso.

c) Alta probabilidad de propagación inadvertida de errores ante modificaciones en el código.\*\*\*

51. ¿Con qué patrón GOF se relaciona el patrón GRASP controlador?

a) Fachada \*\*\*\*

b) Fabricación pura

c) Factory Method.

¿Cuáles de estas clases es más probable que requieran la aplicación de un patrón singletón?

a) Los comandos de un patrón command.

b) Los componentes de un patrón composite.

c) Las factorías completas de un patrón abstract factory. \*\*\*

Hemos implementado un sistema al que queremos añadir ahora la posibilidad de leer una serie de datos de configuración desde distintas fuentes, como por ejemplo ficheros XML, base de datos, etc. ¿Qué patrón GOFT se adapta mejora este problema?

a) Command.

b) Strategy.

c) Façade. \*\*\*\*

55. Nos han pedido diseñar un sistema de aprendizaje... En el, vamos a tener distintos objetos de aprendizaje que se deben instanciar en función del nivel de experiencia del alumno (De momento tres: K12, Adulto, Profesional). ¿Qué patrón aplicarías para solucionar este problema de manera flexible y extensible a nuevos niveles de experiencia? (e.g alumnos universitarios).

a) Builder.

b) Abstract Factory.

c) Proxy

56. En el sistema de aprendizaje uno de los requisitos es la construcción de guías docentes. Estas guías están compuestas de un contexto, una serie de objetivos, contenidos, una descripción de la metodología docente, un cronograma, y unos criterios de evaluación. ¿Qué patrón GOFT es el más adecuado para diseñar esta construcción?

a) Builder. \*\*\*\*\*\*

b) Factory method.

c) Abstract factory.

57. La clase fachada del patrón fachada.

a) Se limita a transmitir invocaciones desde el exterior hacia los objectos del sistema que oculta, preservando la interfaz de estos últimos.

b) Realiza la adaptación entre su interfaz y la interfaz de los objetos del sistema que este oculta. \*\*\*\*

c) Es una clase abstracta de la que heredan las fachadas concretas de cada componente interno del sistema.

58. El patrón builder se utiliza para...

a) Permitir la variación de la representación interna de un producto.

b) Simplificar el código cliente que crea objetos complejos.

c) Todas son ciertas. \*\*\*\*

59. En patrón OBSERVER ...

a) La clase SUBJEST debe tener acceso de algún modo a los datos del OBSERVER.

b) La clase OBSERVER debe tener acceso de algún modo a los datos del SUBJEST.

c) No existe ningún acoplamiento entre las clases SUBJEST Y OBSERVER. \*\*\*\*

60. El patrón Abstract Factory contribuye a preservar el principio OPEN-CLOSED de Bertrant Merey (Diseños abiertos a la extensión y cerrados al cambio) cuando...

a) El punto más probable de variación supone la adición de nuevas factorías concretas

b) El punto más probable de variación supone la adición de nuevas familias de productos.

c) Los puntos más probables de variación suponen (a) la adición de nuevas familias de producto y/o (b) la adición de nuevas factorías concretas . \*\*\*\*