

Guía de actividad



A continuación, le presentamos las orientaciones de las actividades de aprendizaje.

Generalidades

Tiempo total previsto para el desarrollo de la actividad: 4 horas

Indicadores de desempeño:

Evaluar la calidad de las arquitecturas de software mediante la aplicación de técnicas de evaluación y aseguramiento de atributos de calidad, como rendimiento, seguridad y fiabilidad, utilizando herramientas y metodologías avanzadas.

Actividad: Patrones de diseño

Descripción:

En los siguientes tres ejercicios deberá:

- Identificar el tipo de patrón (Estructural, comportamiento, creacional)
- Seleccionar el patrón que considera que es.
- Diseñar el diagrama de clases de la solución.
- Desarrollar el código de los tres ejercicios.

Recuerde que deberá entregar la solución de código en un único lenguaje. Los lenguajes permitidos son Java, Python, C#, JS / TS .

1.

Escenario

Imagina que estás desarrollando una aplicación para una compañía automotriz que permite a los clientes personalizar y ordenar un automóvil. Un objeto Automóvil puede tener muchas configuraciones opcionales: tipo de motor, color, llantas, sistema de sonido, interiores, techo solar, navegación GPS, etc.

Problema

Crear un objeto Automóvil con múltiples configuraciones puede llevar a constructores con muchos parámetros (el infame "constructor telescópico") o a múltiples constructores sobrecargados, lo que dificulta el mantenimiento y legibilidad del código.

Beneficios esperados de la solución:

- Legibilidad y claridad: Facilitar la creación de objetos complejos con muchos parámetros sin necesidad de múltiples constructores o valores por defecto.
- Inmutabilidad: Una vez creado el objeto, sus propiedades no se pueden modificar si el constructor lo define como inmutable.
- Flexibilidad: Poder omitir atributos opcionales sin necesidad de crear subclases o múltiples constructores.
- Separación de construcción y representación: Separar la lógica de construcción del objeto en sí, facilitando modificaciones futuras.

2.**Escenario**

Estás desarrollando una aplicación que gestiona la visualización de notificaciones en diferentes plataformas (por ejemplo: escritorio, móvil, web). Las notificaciones pueden ser de distintos tipos (mensaje, alerta, advertencia, confirmación) y cada tipo puede mostrarse de distintas formas según la plataforma.

Problema

Si usas herencia tradicional, tendrías que crear clases como:

- NotificacionMensajeWeb, NotificacionAlertaWeb, NotificacionMensajeMovil, NotificacionAlertaMovil, etc.

Esto lleva rápidamente a una explosión combinatoria de subclases difíciles de mantener.

Beneficios esperados de la solución:

- Separación de responsabilidades: Separar la lógica de la notificación del medio por el que se presenta.
- Escalabilidad: Poder agregar nuevas plataformas o tipos de notificación sin modificar el resto del sistema.
- Reducción de clases: Evitar la multiplicación de clases para cada combinación.
- Flexibilidad en tiempo de ejecución: Poder cambiar la plataforma dinámicamente si es necesario.

3.

Escenario

Estás desarrollando una aplicación de chat grupal. Los usuarios pueden enviarse mensajes entre sí dentro de una sala de chat. Sin embargo, gestionar las interacciones directas entre cada usuario haría que cada uno deba conocer y comunicarse con todos los demás, lo que resulta en una alta dependencia entre objetos.

Problema

Sin un mediador, cada usuario tendría que mantener referencias directas a todos los demás, lo que genera un sistema difícil de escalar y mantener. Si agregas o eliminas usuarios, debes actualizar muchas relaciones.

Beneficios esperados de la solución:

1. Facilita el mantenimiento: Agregar o eliminar usuarios no debe requerir modificar los demás.
2. Mejor organización: La lógica de comunicación debe estar centralizada, no dispersa en muchos objetos.
3. Reduce la complejidad: Evitar una red enmarañada de interacciones punto a punto.

Recomendaciones para su entrega:

- Recuerde que contará con fechas específicas para su desarrollo, por esto, le recomendamos organizar su tiempo con anterioridad y consultar las fechas habilitadas.
- Recuerde que esta es una actividad que se desarrolla desde el componente grupal.

Rubrica de evaluación

Criterios de evaluación	Indicadores de cumplimiento									
	Excelente		Bueno		Necesita mejorar		Deficiente		No cumple con el criterio o no entrega	
	pts.	Ítem	pts.	ítem	pts.	Ítem	pts.	ítem	pts.	Ítem
Evalúa la correcta identificación del tipo de patrón (estructural, comportamiento , creacional) y la selección adecuada del patrón para cada ejercicio.	5	Identifica con total precisión el tipo de patrón en los 3 ejercicios y selecciona correctamente el patrón más adecuado, con justificación clara.	4	Identifica y selecciona correctamente el patrón en al menos 2 de los 3 ejercicios, con una justificación clara en la mayoría de los casos.	3	Identifica correctamente el patrón en al menos 1 ejercicio, pero muestra errores o poca justificación en los demás.	2	Identifica incorrectamente los patrones en la mayoría de los ejercicios o no explica su selección de forma clara.	0	No identifica correctamente ningún patrón ni seleccióna uno adecuado.
Evalúa la claridad, coherencia y corrección de los diagramas de clases propuestos para los 3 ejercicios.	5	Todos los diagramas de clases están completos, son correctos, claros, coherentes con el patrón elegido y siguen buenas prácticas de modelado.	4	La mayoría de los diagramas son correctos y claros, con pequeños errores o mejoras necesarias en detalles menores.	3	Los diagramas están parcialmente correctos, pero con errores de relaciones, estructura o coherencia con el patrón seleccionado.	2	Los diagramas son confusos, incompletos o presentan errores graves en su estructura o relación con el patrón elegido.	0	No presenta diagramas de clases o estos son completamente incorrectos.
Evalúa la implementación en código de los 3 ejercicios, considerando el	5	Los 3 ejercicios están correctamente implementados, funcionales y alineados con el	4	Al menos 2 ejercicios están correctamente implementados y funcionales, con	3	Solo 1 ejercicio está correctamente implementado	2	Los ejercicios están incompletos o presenten errores graves	0	No se entrega código funcional o este

uso del lenguaje permitido, la coherencia con el diagrama y el patrón, y la funcionalidad.	diagrama de clases y el patrón seleccionado; el código es limpio y sigue buenas prácticas.	pequeñas áreas de mejora en claridad o buenas prácticas.		o o los 3 presentan errores menores que afectan parcialmente su funcionamiento.		que impiden su correcto funcionamiento.		no corresponde a los ejercicios solicitados.
--	--	--	--	---	--	---	--	--