UAX

Ejercicios Tema 2

Somos
#**UAX**makers

1 Ejercicios Tema 2

1. Análisis de la Evolución de la Abstracción

- **1.1. Investigación y Análisis Crítico:** Investiga tres lenguajes de programación: uno procedimental (como C), uno orientado a objetos (como Java), y uno que soporte multiparadigma (como Python). Describe cómo cada uno maneja la abstracción y cómo ha evolucionado este concepto a lo largo del tiempo. Analiza cómo la abstracción beneficia al desarrollo de software en términos de mantenibilidad, escalabilidad y legibilidad.

2. Profundización en POO

- **2.1. Diseño de Clases Avanzado:** Elige tres objetos de la lista creada en el ejercicio original (por ejemplo, una lámpara, un reloj y un libro). Para cada objeto, define una clase en un lenguaje de programación orientado a objetos de tu elección, asegurándote de incorporar atributos privados, métodos públicos, constructores, y destructores. Además, implementa métodos que demuestren la interacción entre estos objetos (como encender la lámpara cuando suena el reloj).

3. Simulación de Interacción entre Objetos

- **3.1. Simulación Compleja:** Amplía el escenario del ejercicio original para incluir al menos cinco objetos que interactúan en una secuencia de eventos (como prepararse por la mañana). Desarrolla un diagrama de flujo que muestre la interacción y luego escribe un pseudocódigo detallado o un script que simule la secuencia, enfocándote en el envío de mensajes entre objetos y el cambio de estado.

4. Modelado de Objetos Complejos y Clases

- **4.1. Diagrama de Clases Avanzado:** Para el objeto complejo seleccionado (como un teléfono móvil), desarrolla un diagrama de clases UML detallado que muestre no solo sus subobjetos y relaciones, sino también herencia, interfaces, y relaciones de dependencia/agregación.
- **4.2. Implementación de Clases Avanzada:** Implementa las clases definidas en el diagrama UML utilizando un lenguaje de programación orientado a objetos. Asegúrate de demostrar conceptos avanzados como encapsulamiento, herencia, y polimorfismo.

5. Proyecto de Herencia y Relaciones

- **5.1. Proyecto de Jerarquía de Herencia:** Basándote en el diagrama de clases de vehículos, expande el proyecto para incluir múltiples niveles de herencia (como Vehículo -> Terrestre -> Coche -> Sedán), e incorpora clases abstractas e interfaces donde sea apropiado. Implementa métodos específicos en cada nivel de herencia que demuestren comportamientos únicos.
- **5.2. Polimorfismo en Acción:** Crea una simulación donde diferentes tipos de vehículos (bicicletas, coches, camiones) son creados y pasados a una función que determina y muestra su movimiento. La función debe tratar a todos los vehículos de manera polimórfica, pero cada tipo de vehículo debe tener una implementación única de movimiento.

6. Polimorfismo y Mensajes Avanzados

- **6.1. Ensayo sobre Polimorfismo:** Escribe un ensayo detallado que explique el concepto de polimorfismo en la programación orientada a objetos, con ejemplos de código en un lenguaje de programación de tu elección. Discute cómo el polimorfismo contribuye a la flexibilidad y mantenibilidad del código.
- **6.2. Implementación de Polimorfismo Dinámico:** Diseña e implementa un sistema en el que diferentes clases derivadas de una superclase común sobrescriban un método específico. Demuestra cómo se pueden manejar objetos de estas clases de manera polimórfica a través de una interfaz común, y cómo el comportamiento varía en tiempo de ejecución dependiendo del tipo de objeto.