Análisis de Sistemas.

Preguntas de Paradigmas OO.

Respuestas:

1. **¿Qué es un paradigma?**

En su simpleza, podría describirse la palabra “paradigma” como una forma de “ver” y/o “entender” el mundo, captarlo de cierta manera. Según Thomas Khun (físico y filosofo de la ciencia) lo describe como un conjunto de teorías, estándares y métodos que juntos representan una forma de organizar el conocimiento.

1. **¿Qué define el paradigma OO (Orientado a Objetos)?**

El paradigma orientado a objetos nace de un lenguaje de programación que se centra en objetos y clases para conformar dicho programa.

Sus conceptos claves que definen un paradigma OO son: Objetos, Clases, Herencia, envío de mensajes, polimorfismo, jerarquía, abstracción, reutilización, encapsulamiento y modularidad (Principalmente en los primeros 4 conceptos).

1. **Defina Análisis OO, Diseño OO y programación OO.**

**Análisis OO**: Es el proceso de examinar y modelar los requisitos del sistema enfocándose en identificar los objetos, clases y sus relaciones en el dominio del problema. Se centra en entender qué necesita el sistema hacer desde la perspectiva de los objetos.

**Diseño OO:** Es el proceso de definir la arquitectura del sistema, especificando cómo los objetos y clases identificados durante el análisis OO interactuarán para cumplir con los requisitos del sistema. Involucra la creación de diagramas y modelos que detallan las estructuras de clases, las relaciones entre ellas y la distribución de responsabilidades.

**Programación OO:** Es la implementación del diseño OO en un lenguaje de programación orientado a objetos. Implica escribir el código que define las clases, crea los objetos y establece las interacciones entre ellos para lograr el comportamiento deseado del sistema.

1. **¿Cuáles son las ventajas de la utilización del paradigma de objetos para el desarrollo de software?**

**Organización del código**: Facilita la estructuración del código en entidades manejables, lo que mejora la claridad y la legibilidad.

**Reutilización de componentes**: Permite reutilizar clases y objetos en diferentes proyectos o partes del mismo proyecto, reduciendo el tiempo y esfuerzo de desarrollo.

**Mantenimiento y actualización**: La estructura modular hace que sea más fácil localizar y corregir errores, así como implementar cambios sin afectar otras partes del sistema.

**Escalabilidad**: Facilita la ampliación y evolución del sistema, permitiendo agregar nuevas funcionalidades sin necesidad de reestructurar el código existente.

**Facilidad de comprensión**: Al modelar entidades del mundo real, los desarrolladores y otros interesados pueden entender mejor el diseño del sistema y su funcionamiento.

**Polimorfismo y flexibilidad**: Permite que una misma interfaz maneje diferentes tipos de objetos, simplificando la extensión del código y la integración de nuevas funcionalidades.

1. **¿Cómo resuelve el paradigma de objetos la problemática de la complejidad del software?**

Puede ser resuelto a través del encapsulamiento que aísla los datos y el comportamiento dentro de los objetos, reduciendo las interdependencias y minimizando el impacto de los cambios en una parte del sistema sobre otras. Asimismo, también el concepto de polimorfismo facilita el manejo de la diversidad de comportamientos a través de una interfaz común, lo que simplifica el diseño y la implementación de sistemas complejos.

1. **Definir clase y objeto. Dar ejemplos.**

**Objeto:** Cualquier cosa sobre la cual puedas emitir algún tipo de concepto. Los objetos son entidades del mundo real que combinan estado, comportamiento e identidad.

Un objeto es una instancia de la clase.

**Ej:** Motocicleta Marca: Honda; Cc: 150; Modelo: XR; Año: 2019.

**Clase:** Una clase es una plantilla que define un conjunto de atributos y métodos comunes a todos los objetos que se crearán a partir de ella. Es una abstracción que describe la estructura y el comportamiento de los objetos. Una clase es como un molde de galleta, este determina la forma y las características que la galleta (el objeto) va a tener, sin ser el objeto real. El molde (la clase) no determina, por ejemplo, el sabor que va a tener cada una de las galletas, ni los ingredientes que la compondrán.

**Ej:** Vehículo -> Motocicleta.

1. .
2. **¿Cuáles son las características de los objetos según su naturaleza?**

**Comportamiento**: Está definido por los métodos con que puede operar dicho objeto, es decir, qué operaciones se pueden realizar con él.

**Estado**: Representa uno o varios atributos a los que se habrán asignado unos valores concretos (datos).

**Identidad**: Es una propiedad de un objeto que lo diferencia del resto.

1. **¿Cuáles son los elementos esenciales y los secundarios del modelo orientado a objetos? Da un ejemplo de cada uno (que no sean los del libro).**

**Elementos Esenciales:**

**Abstracción:** Es el proceso de simplificar un sistema complejo ocultando los detalles innecesarios y resaltando las características esenciales.

**Ejemplo ->** Un auto puede ser abstraído por la clase AUTO con atributos como MARCA, MODELO, AÑO.

**Modularidad:** ES el principio de dividir un sistema en módulos independientes y manejables. Cada módulo representa una unidad funcional del sistema y puede desarrollarse, probarse y mantenerse de manera autónoma.

**Ejemplo ->** En una aplicación de comercio electrónico, los módulos pueden incluir Gestión de productos, Carrito de compras, Procesamiento de pagos, y Gestión de usuarios. Cada módulo se puede desarrollar y actualizar de forma independiente.

**Encapsulamiento: Principio** de ocultar los detalles internos de un objeto y exponer solo una interfaz pública. Esto protege el estado interno del objeto de accesos indebidos y permite controlar cómo se accede y modifica dicho estado.

**Ejemplo ->** Clase: Cuenta bancaria. Atributo interno: Dinero. Método externo: Depositar; Retirar; Consultar.

**Jerarquía:** Es la organización de clases en niveles de generalización y especialización mediante relaciones de herencia. Las clases más generales están en la parte superior y las más especializadas en la parte inferior.

**Ejemplo ->** Clase: Animal -> Mamífero -> (Perro; Gato).

->Ovíparo -> (Paloma, Pollo).

**Elementos Secundarios:**

**Tipificación**: características precisas que comparten una serie de objetos. Puesta en vigencia de la clase de los objetos.

**Concurrencia**: permite a diferentes objetos actuar al mismo tiempo. Distingue objetos activos de inactivos.

**Persistencia**: cantidad de espacio que ocupa y tiempo que dura un objeto. Conservación del estado del objeto en el espacio y en el tiempo.

1. . **¿Qué permite modelar el Modelo de Objetos del Dominio del Problema? ¿Qué herramienta se utiliza para desarrollarlo?**

El Modelo de Objetos del Dominio del Problema permite modelar las entidades, atributos y relaciones del mundo real dentro del contexto del problema que se está resolviendo. Proporciona una representación conceptual del sistema que facilita la comprensión y el diseño del software.

Una herramienta comúnmente utilizada para desarrollar el Modelo de Objetos del Dominio del Problema es el **Diagrama de Clases UML (Unified Modeling Language)**.

1. **¿Cómo se identifican las clases que participan de un dominio de problema?**

**Análisis del dominio**: Examinar el problema y el contexto en el que se encuentra, identificando los objetos y entidades relevantes del mundo real.

**Revisión de requisitos**: Leer y analizar los requisitos del sistema para encontrar sustantivos (nombres) que indiquen posibles clases y verbos que sugieran métodos.

**Entrevistas y talleres**: Consultar con los expertos del dominio y las partes interesadas para obtener su perspectiva sobre las entidades y sus interacciones.

**Identificación de patrones**: Utilizar patrones de diseño y soluciones previas similares para identificar clases comunes en el dominio.

**Modelado conceptual**: Crear modelos conceptuales o diagramas de entidades para visualizar y refinar la identificación de clases.

**Iteración y refinamiento**: Revisar y ajustar continuamente las clases identificadas a medida que se obtiene más información sobre el dominio del problema.

1. **El diagrama de clases ¿qué modela del sistema? ¿Qué vista del sistema muestra, dinámica o estática, interna o externa? Explique.**

El diagrama de clases modela la estructura estática del sistema, mostrando las clases, sus atributos, métodos y las relaciones entre ellas (como asociaciones, herencias, y composiciones).

Muestra una vista estática -> Enfocándose en la estructura y la organización de los componentes del sistema en términos de clases y relaciones. No muestra cómo las clases interactúan en tiempo de ejecución.

Muestra una vista interna -> Aunque un diagrama de clases puede usarse para comunicar aspectos tanto internos como externos del sistema, se centra principalmente en los detalles internos de la implementación y la arquitectura del sistema.

1. **¿Cuáles son los elementos que conforman una clase?**

Los elementos que conforman una clase son: Nombre, Caracteristicas, Comportamiento y Responsabilidades.

1. **¿Cuáles son las vistas de una clase?**

**Vista estructural**: Muestra la estructura de la clase, incluyendo sus atributos, métodos y relaciones con otras clases. Se puede visualizar mediante diagramas de clases UML.

**Vista funcional**: Describe el comportamiento de la clase, cómo sus métodos interactúan entre sí y con los métodos de otras clases. Puede representarse mediante diagramas de secuencia o diagramas de actividad UML.

**Vista de implementación**: Detalla la implementación concreta de la clase en código, incluyendo definiciones de atributos y métodos, así como cualquier lógica específica de la implementación.

**Vista de uso**: Muestra cómo se utiliza la clase en diferentes escenarios, a través de casos de uso o diagramas de interacción que ilustran las instancias de la clase y sus interacciones.

1. **Envío de mensajes.**

El “envío de mensajes” se refiere a la comunicación entre objetos. Cuando un objeto necesita que otro objeto realice una acción o proporcione información, envía un mensaje a ese objeto.

**Mensajes y Métodos**: En términos de implementación, un mensaje corresponde a la invocación de un método en el objeto receptor. El objeto emisor solicita al receptor que ejecute uno de sus métodos.

**Por ejemplo**: Si consideramos una clase Persona con un método saludar (), un objeto persona1 podría enviar un mensaje a un objeto persona2 pidiéndole que ejecute su método saludar (). Esto sería equivalente a persona2.saludar() en un lenguaje de programación orientado a objetos.

Enlace a MAPA MENTAL:

https://www.canva.com/design/DAGIbE1r688/w\_q8EbOM3zHgEBSDYtv58w/view?utm\_content=DAGIbE1r688&utm\_campaign=designshare&utm\_medium=link&utm\_source=editor