14 Apéndice B: Resumen de las Principales Propiedades de la POO

¿Cuál es la diferencia entre Programación Orientada a Objetos (POO) y la Programación Estructurada?

Programación orientada a objetos:

- La programación orientada a objetos es un tipo de programación que se basa en objetos en lugar de solo funciones y procedimientos:
- Enfoque de abajo hacia arriba.
- Proporciona ocultación de datos.
- Puede resolver problemas de cualquier complejidad.
- El código se puede reutilizar, reduciendo así la redundancia.

Programación estructurada:

- Proporciona una estructura lógica a un programa donde los programas se dividen en funciones.
- Enfoque de arriba hacia abajo
- No proporciona ocultación de datos.
- Puede resolver problemas moderados
- No admite la reutilización del código.

¿Qué es la Programación Orientada a Objetos?

La programación orientada a objetos es un tipo de programación que se basa en objetos en lugar de solo funciones y procedimientos. Los objetos individuales se agrupan en clases. La programación orientada a objetos implementa entidades del mundo real como herencia, polimorfismo, ocultación, etc. en la programación. También permite vincular datos y código.

¿Por qué utilizar programación orientada a objetos?

- La programación orientada a objetos permite claridad en la programación, lo que permite la simplicidad en la resolución de problemas complejos.
- El código se puede reutilizar mediante herencia, lo que reduce la redundancia
- Los datos y el código están unidos mediante encapsulación.
- La programación orientada a objetos permite ocultar datos, por lo tanto, los datos privados se mantienen confidenciales
- Los problemas se pueden dividir en diferentes partes, lo que facilita su resolución.
- El concepto de polimorfismo le da flexibilidad al programa al permitir que las entidades tengan múltiples formas.

¿Cuáles son las características principales de la programación orientada a objetos?

- Herencia
- Encapsulación
- Polimorfismo
- Abstracción de datos

¿Qué es un objeto?

> Un objeto es una entidad del mundo real que es la unidad básica de la programación orientada a objetos, por ejemplo: silla, gato, perro, etc. Diferentes objetos tienen diferentes estados, atributos y comportamientos.

¿Qué es una clase?

Una clase es un prototipo que consta de objetos en diferentes estados y con diferentes comportamientos. Tiene una serie de métodos que son comunes a los objetos presentes dentro de esa clase.

¿Cuál es la diferencia entre una clase y una estructura?

Clase: modelo definido por el usuario a partir del cual se crean los objetos. Consiste en métodos o conjunto de instrucciones que se deben realizar sobre los objetos.

Estructura: una estructura es básicamente una colección de variables definidas por el usuario que son de diferentes tipos de datos.

¿Puedes llamar al método de la clase base sin crear una instancia?

Sí, puedes llamar a la clase base sin crear una instancia si:

- Es un método estático.
- La clase base es heredada por alguna otra subclase.

¿Cuál es la diferencia entre una clase y un objeto?

Objeto:

- Una entidad del mundo real que es una instancia de una clase.
- Un objeto actúa como una variable de la clase.
- Un objeto es una entidad física.
- Los objetos ocupan espacio en la memoria cuando se crean.
- Los objetos se pueden declarar cuando sea necesario.

Clase:

- Una clase es básicamente una plantilla o un plano dentro del cual se pueden crear objetos.
- Une métodos y datos en una sola unidad.

- Una clase es una entidad lógica.
- Una clase no ocupa espacio en la memoria cuando se crea.
- Las clases se declaran solo una vez.

¿Qué es la herencia?

La herencia es una característica de la programación orientada a objetos que permite que las clases hereden propiedades comunes de otras clases. Por ejemplo, si hay una clase como "vehículo", otras clases como "coche", "bicicleta", etc. pueden heredar propiedades comunes de la clase de vehículo. Esta propiedad le ayuda a deshacerse del código redundante, reduciendo así el tamaño total del código.

¿Cuáles son los diferentes tipos de herencia?

- Herencia única
- Herencia múltiple
- Herencia multinivel
- Herencia jerárquica
- Herencia híbrida

¿Cuál es la diferencia entre herencia múltiple y multinivel?

Herencia múltiple:

- La herencia múltiple entra en escena cuando una clase hereda más de una clase base.
- Ejemplo: una clase que define un hijo hereda de dos clases base, Madre y Padre.

Herencia multinivel:

- La herencia multinivel significa que una clase hereda de otra clase que a su vez es una subclase de alguna otra clase base.

- Ejemplo: una clase que describe un automóvil deportivo heredará de una clase base Auto, que a su vez hereda otra clase Vehículo.

¿Qué es la herencia híbrida?

La herencia híbrida es una combinación de herencia múltiple y de varios niveles.

¿Qué es la herencia jerárquica?

La herencia jerárquica se refiere a la herencia en la que una clase base tiene más de una subclase. Por ejemplo, la clase de vehículo puede tener "coche", "bicicleta", etc. como subclases.

¿Cuáles son las limitaciones de la herencia?

- Aumenta el tiempo y el esfuerzo necesarios para ejecutar un programa, ya que requiere saltar de una clase a otra.
- La clase padre y la clase hija se acoplan estrechamente
- Cualquier modificación al programa requeriría cambios tanto en la clase principal como en la clase secundaria.
- Necesita una implementación cuidadosa, de lo contrario conduciría a resultados incorrectos.

¿Qué es una superclase?

Una superclase o clase base es una clase que actúa como padre de alguna otra clase o clases. Por ejemplo, la clase Vehículo es una superclase de la clase Coche.

¿Qué es una subclase?

Una clase que hereda de otra clase se llama subclase. Por ejemplo, la clase Coche es una subclase o una derivada de la clase Vehículo.

¿Qué es el polimorfismo?

El polimorfismo se refiere a la capacidad de existir en múltiples formas. Se pueden dar múltiples definiciones a una única interfaz. Por ejemplo, si tiene una clase llamada Vehículo, puede tener un método llamado velocidad pero no puede definirlo porque diferentes vehículos tienen diferentes velocidades. Este método se definirá en las subclases con diferentes definiciones para diferentes vehículos.

¿Qué es el polimorfismo estático?

El polimorfismo estático (enlace estático) es un tipo de polimorfismo que ocurre en el momento de la compilación. Un ejemplo de polimorfismo en tiempo de compilación es la sobrecarga de métodos.

¿Qué es el polimorfismo dinámico?

El polimorfismo en tiempo de ejecución o polimorfismo dinámico (enlace dinámico) es un tipo de polimorfismo que se resuelve durante el tiempo de ejecución. Un ejemplo de polimorfismo en tiempo de ejecución es la anulación de métodos.

¿Qué es la sobrecarga de métodos?

La sobrecarga de métodos es una característica de la programación orientada a objetos que permite dar el mismo nombre a más de un método dentro de una clase si los argumentos pasados difieren.

¿Qué es la anulación de métodos?

La anulación de métodos es una característica de la programación orientada a objetos mediante la cual la clase secundaria o la subclase puede redefinir los métodos presentes en la clase base o la clase principal. Aquí, el método que se anula tiene el mismo nombre y la firma, lo que significa los argumentos pasados y el tipo de retorno.

¿Qué es la sobrecarga del operador?

La sobrecarga de operadores se refiere a la implementación de operadores utilizando tipos definidos por el usuario en función de los argumentos que se transmiten.

Diferenciar entre sobrecarga y anulación.

Sobrecarga:

- Dos o más métodos que tienen el mismo nombre pero diferentes parámetros o firma
- Resuelto durante el tiempo de compilación

Anulación:

- Métodos de redefinición de clases secundarias presentes en la clase base con los mismos parámetros/firma
- Resuelto durante el tiempo de ejecución

¿Qué es la encapsulación?

La encapsulación se refiere a unir los datos y el código que funciona en ellos en una sola unidad. Por ejemplo, una clase. La encapsulación también permite ocultar datos, ya que los datos especificados en una clase están ocultos para otras clases.

¿Qué son los especificadores de acceso?

Los especificadores de acceso o modificadores de acceso son palabras clave que determinan la accesibilidad de métodos, clases, etc. en POO. Estos especificadores de acceso permiten la implementación de la encapsulación. Los especificadores de acceso más comunes son público, privado y protegido. Sin embargo, hay algunos más que son específicos de los lenguajes de programación.

¿Cuál es la diferencia entre modificadores de acceso públicos, privados y protegidos?

Nombre	Accesibilidad desde propia clase	Accesibilidad desde clase derivada	Accesibilidad desde el mundo
Público	*Sí*	*Sí*	*Sí*
Privado	*Sí*	*No*	*No*
Protegido	*Sí*	*Sí*	*No*

¿Qué es la abstracción de datos?

La abstracción de datos es una característica muy importante de la programación orientada a objetos que permite mostrar solo la información importante y ocultar los detalles de implementación. Por ejemplo, mientras andas en Vehículo, sabes que si presionas el acelerador, la velocidad aumentará, pero no sabes cómo sucede realmente. Se trata de una abstracción de datos, ya que los detalles de implementación están ocultos para el usuario.

¿Cómo lograr la abstracción de datos?

La abstracción de datos se puede lograr mediante:

- Clase abstracta
- Método abstracto

¿Qué es una clase abstracta?

Una clase abstracta es una clase que consta de métodos abstractos. Estos métodos están básicamente declarados pero no definidos. Si estos métodos se van a utilizar en alguna subclase, deben definirse exclusivamente en la subclase.

¿Puedes crear una instancia de una clase abstracta?

No. No se pueden crear instancias de una clase abstracta porque no tiene una implementación completa. Sin embargo, se pueden crear instancias de subclases que hereden la clase abstracta.

¿Qué es una interfaz?

Es un concepto de programación orientada a objetos que permite declarar métodos sin definirlos. Las interfaces, a diferencia de las clases, no son planos porque no contienen instrucciones detalladas ni acciones a realizar. Cualquier clase que implemente una interfaz define los métodos de la interfaz.

Diferenciar entre abstracción y encapsulación de datos.

Abstracción de datos:

- Resuelve el problema a nivel de diseño.
- Permite mostrar aspectos importantes mientras oculta detalles de implementación.

Encapsulación:

- Resuelve el problema a nivel de implementación.
- Une código y datos en una sola unidad y la oculta del mundo

¿Qué son las funciones virtuales?

Las funciones virtuales son funciones que están presentes en la clase principal y son anuladas por la subclase. Estas funciones se utilizan para lograr polimorfismo en tiempo de ejecución.

¿Qué son funciones virtuales puras?

Las funciones virtuales puras o funciones abstractas son funciones que sólo se declaran en la clase base. Esto significa que no contienen ninguna definición en la clase base y deben redefinirse en la subclase.

¿Qué es un constructor?

Un constructor es un tipo especial de método que tiene el mismo nombre que la clase y se utiliza para inicializar objetos de esa clase.

¿Qué es un destructor?

Un destructor es un método que se invoca automáticamente cuando se destruye un objeto. El destructor también recupera el espacio de almacenamiento dinámico asignado al objeto destruido, cierra los archivos y las conexiones de la base de datos del objeto, etc.

Tipos de constructores

Los tipos de constructores difieren de un lenguaje a otro. Sin embargo, todos los constructores posibles son:

- Constructor predeterminado
- Constructor parametrizado
- Copiar constructor
- Constructor estático
- Constructor privado

¿Qué es un constructor de copias?

Un constructor de copias crea objetos copiando variables de otro objeto de la misma clase. El objetivo principal de un constructor de copias es crear un nuevo objeto a partir de uno existente.

¿Para qué sirve 'finalizar'?

Finalizar como método de objeto utilizado para liberar recursos no administrados y limpiar antes de la recolección de basura (GC). Realiza tareas de gestión de memoria.

¿Qué es la recolección de basura (GC)?

GC es una implementación de gestión automática de memoria. El recolector de basura libera espacio ocupado por objetos que ya no existen.

Diferenciar entre una clase y un método.

Clase: es básicamente una plantilla que une el código y los datos en una sola unidad. Las clases constan de métodos, variables, etc.

Método: Conjunto de instrucciones invocables, también llamado procedimiento o función, que deben realizarse con los datos dados.

¿Diferenciar entre una clase abstracta y una interfaz?

Clase abstracta:

- Métodos: Puede tener métodos abstractos y otros.

- Variables finales: Puede contener variables finales y no finales.
- Accesibilidad de los datos de los miembros: Puede ser privado, público, etc.
- Implementación: Puede proporcionar la implementación de una interfaz.

Interfaz:

- Métodos: Sólo métodos abstractos
- Variables finales: Las variables declaradas son finales por defecto.
- Accesibilidad de los datos de los miembros: Público por defecto.
- Implementación: No se puede proporcionar la implementación de una clase abstracta

¿Qué es una variable final?

Una variable cuyo valor no cambia. Siempre se refiere al mismo objeto por la propiedad de no transversalidad.

¿Qué es una excepción?

Una excepción es un tipo de notificación que interrumpe la ejecución normal de un programa. Las excepciones proporcionan un patrón para el error y transfieren el error al controlador de excepciones para resolverlo. El estado del programa se guarda tan pronto como se genera una excepción.

¿Qué es el manejo de excepciones?

El manejo de excepciones en la programación orientada a objetos es un concepto muy importante que se utiliza para gestionar errores. Un controlador de excepciones permite generar y detectar errores e implementa un mecanismo centralizado para resolverlos.

¿Cuál es la diferencia entre un error y una excepción?

Error: Los errores son problemas que las aplicaciones no deberían encontrar.

Excepción: Condiciones que una aplicación podría intentar detectar

¿Qué es un bloque *try/catch*?

Se utiliza un bloque try/catch para manejar excepciones. El bloque *try* define un conjunto de declaraciones que pueden provocar un error. El bloque *catch* básicamente detecta la excepción.

¿Qué es un bloque *Finally*?

Un bloque *Finally* consta de código que se utiliza para ejecutar código importante, como cerrar una conexión, etc. Este bloque se ejecuta cuando sale el bloque *try*. También garantiza que el bloque finalmente se ejecute incluso en caso de que se encuentre alguna excepción inesperada.

¿Cuáles son las limitaciones de la programación orientada a objetos?

- Generalmente no es adecuado para problemas pequeños.
- Requiere pruebas intensivas.
- Toma más tiempo resolver el problema.
- Requiere una planificación adecuada.
- El programador debería pensar en resolver un problema en términos de objetos.