TEXT CLASS REVIEW

TEMAS A TRATAR EN EL CUE

- ¿Qué es la Orientación a Objetos?
- Programación Orientada a Objetos.
- Importancia de la orientación a objetos en la programación.
- Diferencias con programación estructurada.
- Orientación a objetos aplicada a la vida cotidiana.
- ¿Qué es un Objeto?
- ¿Qué es una Clase?
- Diferencia entre clase, instancia y objeto.
- Atributos de una clase.
- Estado de un objeto.
- Diferencia entre atributo y estado.
- Métodos de una clase.
- Comportamiento de un objeto.
- Diferencia entre método y comportamiento.
- Principios Básicos.
- Abstracción y encapsulamiento.

LA ORIENTACIÓN A OBJETOS (OO)

Hace ya décadas que el paradigma de Orientación a Objetos (OO) tomó su lugar en el desarrollo de software. Los conceptos fundamentales que se deben entender para poder ser buenos desarrolladores OO son: abstracción, encapsulación, identidad, modularidad, jerarquía, tipos, concurrencia y persistencia.

- La **abstracción** es el mecanismo a través del cual nos enfocamos en los aspectos esenciales o distintivos de algo, ignorando detalles irrelevantes.
- La **identidad** es la propiedad de un objeto, que lo distingue de todos los demás. La identidad es necesaria para que podamos hablar con un objeto sin confundirlo con otro, y para que puedan existir al mismo tiempo varios objetos de la misma clase.
- La **encapsulación** consiste en que los detalles de una clase, estructuras de datos, algoritmos, entre otros, se hacen privados, o encapsulan, para que sea imposible que otras clases dependan de ellos.

- La **modularidad** consiste en la descomposición de algo grande y complejo, en partes más sencillas y manejables.
- La **jerarquía** es una organización de elementos de acuerdo con su tipo, de acuerdo a una estructura de árbol. Los dos tipos de jerarquías más comunes en OO son: la jerarquía por herencia o generalización, y la jerarquía por agregación. En la primera, se aplica la frase "es un tipo de"; mientras que en la segunda se aplica "es parte de".
- La **concurrencia** se preocupa por administrar el acceso a recursos compartidos entre operaciones que se sobreponen en el tiempo (incluyendo la ejecución en paralelo).
- La **persistencia** de los objetos tiene un periodo de existencia, desde los más volátiles, hasta los más estables. Si un objeto requiere sobrevivir al proceso en que se ejecuta, entonces se dice que es persistente. En otras palabras, la persistencia es la capacidad de un objeto para existir más allá del proceso que lo ejecuta. Para implementarla, requerimos de algún mecanismo para almacenar datos. Existen diversos mecanismos, desde archivos en texto plano, hasta bases de datos relacionales (RDBMS) orientadas a objetos (OODBMS).

PROGRAMACIÓN ORIENTADA A OBJETOS

Es un paradigma de programación, es decir, un modelo o un estilo de programación que nos indica unas pautas sobre cómo trabajar con él. Principalmente se basa en el concepto de clases y objetos. Este tipo de programación se utiliza para estructurar un programa de software en piezas simples y reutilizables de planos de código (clases) para crear instancias individuales de objetos.

Con el paradigma de Programación Orientado a Objetos, lo que buscamos es dejar de centrarnos en la lógica pura de los programas, para empezar a pensar en objetos, lo que constituye la base de todo esto.

IMPORTANCIA DE LA PROGRAMACIÓN ORIENTADA A OBJETOS

- La POO permite que el código sea reutilizable, organizado y fácil de mantener.
- Sigue el principio de desarrollo de software utilizado por muchos programadores, para evitar duplicar el código y crear de esta manera programas eficientes.
- Evita el acceso no deseado a los datos, o la exposición de código propietario mediante la encapsulación y la abstracción, de la que nos referiremos en detalle más adelante.

- La POO ayuda en sistemas grandes, ya que, en vez de pensar en funciones, pensamos en las relaciones o interacciones de los diferentes componentes del sistema. Este paradigma de programación se asemeja al mundo real, agilizando el desarrollo de software.

DIFERENCIAS ENTRE LA PROGRAMACIÓN ESTRUCTURADA Y LA PROGRAMACIÓN ORIENTADA A OBJETOS

Una diferencia importante entre la Programación Estructurada y la programación Orientada a Objetos es que la primera separa los datos de las funciones; mientras que la segunda define un conjunto de objetos donde se combinan de forma modular los datos con las funciones.

La Programación Orientada a Objetos se basa en una nueva forma de pensar los problemas, declarando como variables o los tipos de datos los objetos del problema, y que, a su vez, cada objeto tiene anidadas variables que hacen referencia al dato.

La Programación Estructurada pretende resolver un problema de principio a fin en una sola estructura de código. Mientras que la Programación Orientada a Objetos resuelve el problema identificando los actores que tienen participación en el problema, y también sus acciones.

Podemos aplicar la POO a la vida cotidiana. Un ejemplo sencillo de un objeto, podemos definir a un animal. Un animal tiene una edad, por lo que creamos un nuevo atributo de 'edad' y, además, puede envejecer, por lo que definimos un nuevo método. Datos y lógicas, esto es lo que se define en muchos programas como la definición de una clase, que es la definición global y genérica de muchos objetos.

ORIENTACIÓN A OBJETOS APLICADA A LA VIDA COTIDIANA

La Programación Orientada a Objetos (POO) como una forma especial de programar, más cercana a cómo expresaríamos las cosas en la vida real que otros tipos de programación, que permite diseñar mejor las aplicaciones, llegando a mayores cuotas de complejidad, sin que el código se vuelva inmanejable.

CLASES Y OBJETOS

Para entender este paradigma primero tenemos que comprender qué es una clase, y qué es un objeto.

Un **objeto** es una entidad que agrupa un estado y una funcionalidad relacionadas. El estado del objeto se define a través de variables llamadas atributos, mientras que la funcionalidad se modela a través de funciones a las que se les conoce con el nombre de métodos del objeto.

En la POO interesa el comportamiento del objeto. El software que implementa el objeto contiene estructuras de datos y operaciones que expresan dicho comportamiento.

Entonces, dentro del software orientado a objeto, un objeto es cualquier cosa, real o abstracta, acerca de la cual almacenamos datos y los métodos que controlan dichos datos.

Un ejemplo de objeto podría ser un vehículo, en el que tendríamos atributos como la marca, el número de puertas, o el tipo de carburante; y métodos como arrancar y parar. O bien cualquier otra combinación de atributos y métodos según lo que fuera relevante para nuestro programa.

Una **clase,** por otro lado, no es más que una plantilla genérica a partir de la cual se instancian los objetos; plantilla que es la que define qué atributos y métodos tendrán los objetos de esa clase. Volviendo a nuestro ejemplo: en el mundo real existe un conjunto de objetos a los que llamamos vehículos, y que poseen un conjunto de atributos y comportamientos comunes, esto es a lo que llamamos clase. Sin embargo, tenemos vehículos distintos unos de otros, y aunque pertenecen a la misma clase de objetos, son distintos.

Partiendo de lo anterior, podemos **diferenciar entre una clase, instancia y objeto.** Una clase es la definición en tiempo de diseño de un objeto dentro del programa, y representan un conjunto de variables y métodos para operar con datos (del objeto). El objeto es una copia de la clase. La instancia es una variable que contiene la dirección de memoria del objeto. También puede tener varios objetos de la misma clase y luego varias instancias de cada uno de esos objetos. Un objeto o instancia es la materialización de la clase, en tiempo de ejecución.

ATRIBUTOS DE UNA CLASE

Los atributos, también llamados datos o variables miembro, son porciones de información que un objeto posee o conoce de sí mismo. Son las características individuales que diferencian un objeto de otro, y que determinan su apariencia, estado u otras cualidades. Los atributos se guardan en variables denominadas de instancia, y cada objeto particular puede tener valores distintos para estas variables.

Una clase puede tener cualquier número de atributos, o no tener ninguno. Se declaran con un identificador y el tipo de dato correspondiente. Los atributos son las propiedades que pueden asumir los objetos dentro de una clase. Por ejemplo: el atributo «color» puede tener el valor «azul»

o «blanco», y en nuestro ejemplo de la clase vehículo, un atributo puede ser la marca, modelo y color.

ESTADO DE UN OBJETO

Abarca todas las propiedades del objeto, y los valores actuales de cada una de esas propiedades. Las propiedades de los objetos suelen ser estáticas, mientras los valores que toman estas propiedades cambian con el tiempo.

El hecho de que los objetos tengan estado implica que ocupan un espacio, en el mundo físico, y en la memoria del computador. El estado de un objeto está influido por la historia de éste. No deben confundirse los objetos, que existen en el tiempo, son mutables, tienen estado, pueden ser creados, destruidos y compartidos; con los valores (los asignados a una variable, por ejemplo) que son cantidades con las propiedades de ser atemporales, inmutables. El estado de un objeto representa el efecto acumulado de su comportamiento.

La **diferencia entre atributo y estado** en la POO es que los atributos son las propiedades o características del objeto, mientras el estado de un objeto viene definido por los valores que toma en un instante determinado los atributos que define al objeto. El valor concreto de una propiedad o atributo de un objeto se llama estado.

MÉTODOS EN LAS CLASES

Son las funcionalidades asociadas a los objetos, que son implantadas o programadas dentro de las clases. Es decir, cuando estamos programando las clases, las funciones que creamos dentro, asociadas a esas clases, las llamamos métodos.

COMPORTAMIENTO DE UN OBJETO

Es la manera en que una clase de objetos puede hacer cualquier cosa para sí o para otros objetos. El comportamiento de una clase determina qué objetos de esa clase hacen cambiar sus atributos, y también qué hacen cuando otros objetos les piden hacer algo. El comportamiento para una clase de objetos se implementa a través de métodos.

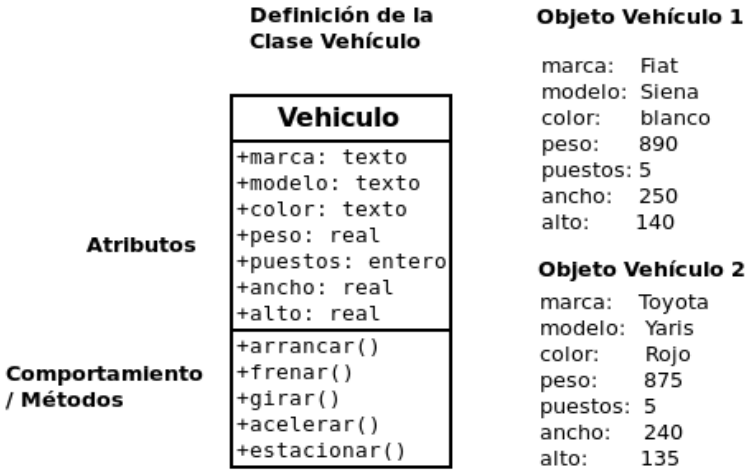

Ilustración 1 Definición de clase y objetos

DIFERENCIA ENTRE MÉTODO Y COMPORTAMIENTO

El método en sí es un algoritmo asociado a un objeto (o una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". Mientras que el comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.

Por lo tanto, el comportamiento permite que los objetos se comuniquen entre sí por medio de sus métodos. Una clase o un objeto pueden llamar métodos en otra clase u objeto, y cambien su comportamiento. Por ejemplo:

- Para informar un cambio a otro objeto.
- Para indicar al otro objeto que cambie algo acerca de sí mismo.
- Para pedir a otro objeto que haga algo.

PRINCIPIOS BÁSICOS ABSTRACCIÓN Y ENCAPSULAMIENTO

La **abstracción** consiste en aislar un elemento de su contexto, o del resto de los elementos que lo acompañan. En programación, el término se refiere al énfasis en el "¿qué hace?" más que en el "¿cómo lo hace?".

En la abstracción denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar "cómo" se implementan estas características.

Los procesos, las funciones o los métodos pueden también ser abstraídos, y, cuando lo están, una variedad de técnicas es requerida para ampliar una abstracción. La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a armar un conjunto de clases que permitan modelar la realidad, o el problema que se quiere atacar.

La **Encapsulación** es el proceso en el cual se comparten los elementos de una abstracción que constituyen su estructura y comportamiento. Sirve para separar la interfaz de una abstracción y su implementación. Es un concepto complementario al de abstracción, y esconde la implementación del objeto que no contribuye a sus características esenciales.

La encapsulación da lugar a que las clases se dividan en dos partes:

• La **interfaz**: captura la visión externa de una clase, abarcando la abstracción del comportamiento común de esa clase.

• La **implementación**: comprende la representación de la abstracción, así como los mecanismos que conducen al comportamiento deseado.

Se conoce también como ocultamiento o privacidad de la información, es decir, en la Programación Orientada a Objetos, se denomina encapsulamiento al ocultamiento del estado de los datos miembro de un objeto, de manera que solo se pueda cambiar mediante las operaciones definidas para él.

El aislamiento protege a los datos asociados de un objeto contra su modificación por quien no tenga derecho a acceder a ellos, eliminando efectos secundarios e interacciones. De esta forma el usuario de la clase puede obviar la implementación de los métodos y propiedades para concentrarse sólo en cómo usarlos. Por otro lado, se evita que el usuario pueda cambiar su estado de maneras imprevistas e incontroladas.

Tipos de encapsulamiento:

• Estándar: predeterminado.

• Abierto: hace que el miembro de la clase pueda ser accedido desde el exterior de la Clase y cualquier parte del programa.

• Protegido: sólo es accesible desde la Clase y las clases que heredan (a cualquier nivel).

• Semi cerrado: sólo es accesible desde la clase heredada.

• Cerrado: sólo es accesible desde la Clase.