Programacion Orientada a Objetos.

Dany Julian Zenteno Tancara

Agosto 4 del 2016

Programacion Orientada a Objetos

Definicion

Se puede hablar de POO cuando se reunen las caracteristicas de : abstraccion, encapsulamiento, herencia y polimorfismo; y los conceptos basicos que las forman: objetos, mensajes, clases, instancias y metodos. Es una solucion para el desarrollo de soluciones computacionales utilizando componentes de Software (objetos de software).

Objeto: componente o codigo de software que contiene en si mismo tanto sus caracteristicas (campos) como sus comportamientos (metodos); se accede atravez de su inerfaz. Conceptos Basicos ;

- Un objeto es una encapsulacion abstracta de informacion junto con los metodos o procedimientos para manipularla.
- Una clase equivale a la generalizacion o abstraccion de un tipo especifico de objetos.
- Un mensaje representa una accion a tomar por un determinado objeto.
- Una instancia es la correcion de una clase.
- Un metodo consiste en la implementacion en una clase de protocolo de respuesta a los mensajes dirigidos a los objetos de la misma.

Características:

> - La **abstracción:** Consiste en la generalización conceptual de un determinado conjunto de

objetos y de sus atributos y propiedades, dejando en un segundo término los detalles concretos de cada objeto. ¿Qué se consigue con la **abstracción**? Bueno, básicamente pasar del plano material (cosas que se tocan) al plano mental *(cosas que se piensan).*

- La **encapsulación:** Se refiere a la capacidad de agrupar y condensar en un entorno con límites bien-definidos distintos elementos. Cuando hablemos de **encapsulación** en general siempre nos referiremos, pues, a **encapsulación** abstracta. De manera informal, primero generalizamos *(la abstracción)* y luego decimos: la generalización está bien, pero dentro de un cierto orden: hay que poner límites *(la encapsulación)*, y dentro de esos límites vamos a meter, a saco, todo lo relacionado con lo abstraído: no sólo datos, sino también métodos, comportamientos, etc. Por un lado es una abstracción pues, de acuerdo con la definición establecida anteriormente, es en ésta donde se definen las propiedades y atributos genéricos de determinados objetos con características comunes *(recordemos el ejemplo de la sala de cine).* La Clase es, por otro lado, una **encapsulación** porque constituye una cápsula o saco que encierra y amalgama de forma clara tanto los datos de que constan los objetos como los procedimientos que permiten manipularlos. Las Clases se constituyen, así, en abstracciones encapsuladas.

- La **herencia:** Se aplica sobre las clases. O sea, de alguna forma las clases pueden tener descendencia, y ésta heredará algunas características de las clases "padres". Si disponemos las clases con un formato de árbol genealógico, tenderemos lo que se denomina una estructura jerarquizada de clases. La OOP promueve en gran medida que las relaciones entre objetos se basen en construcciones jerárquicas. Esto es, las clases pueden heredar diferencialmente de otras clases *(denominadas "superclases")* determinadas características, mientras que, a la vez, pueden definir las suyas propias. Tales clases pasan, así, a denominarse "subclases" de aquéllas. La **herencia** se implementa mediante un mecanismo que se denomina derivación de clases: las superclases pasan a llamarse clases base, mientras que las subclases se constituyen en clases derivadas. El mecanismo de **herencia** está fuertemente entroncado con la reutilización del código en OOP. Una clase derivada posibilita, el fácil uso de código ya creado en cualquiera de las clases base ya existentes. El concepto de **herencia** constituye un estrato básico del paradigma de objetos, pero esto no significa que todas las relaciones entre clases en OOP deban ajustarse siempre a este modelo jerárquico. Es necesario establecer si la pretendida relación entre objetos es de pertenencia o de derivación. En una relación típica de pertenencia un objeto contiene al otro

- **Polimorfismo:** Esta propiedad, como su mismo nombre sugiere múltiples formas, se refiere a la posibilidad de acceder a un variado rango de funciones distintas a través del mismo interfaz. O sea, que, en la práctica, un mismo identificador puede tener distintas formas (distintos cuerpos de función, distintos comportamientos) dependiendo, en general, del contexto en el que se halle inserto. El polimorfismo se puede establecer mediante la sobrecarga de identificadores y operadores, la ligadura dinámica y las funciones virtuales.

> El término sobrecarga se refiere al uso del mismo identificador u operador en distintos contextos y con distintos significados. La sobrecarga de funciones conduce a que un mismo nombre pueda representar distintas funciones con distinto tipo y número de argumentos. En el ámbito de la OOP, la sobrecarga de funciones equivale a que un mismo mensaje puede ser enviado a objetos de diferentes clases de forma que cada objeto respondería al mensaje apropiadamente. La sobrecarga de operadores permite, por otro lado, el desarrollo de un código más coherente, como especialización de la sobrecarga de funciones, posibilitando la re-definición (para tipos de datos definidos-por-el-usuario) de las operaciones realizadas por éstos **(+, -, *, >, etc.).**

Otros Conceptos

1. **Agregación:** *Composición de un objeto por otros. Es una relación más débil que la que existe entre el atributo y el objeto al cual pertenece, y más fuerte que una asociación.*
2. **Concurrencia:** *Propiedad que distingue un objeto activo de otro inactivo.*
3. **Persistencia:** *Es la propiedad de un objeto cuya existencia trasciende el tiempo y/o el espacio (ej. el objeto continua existiendo luego de que su creador deja de existir / la ubicación de un objeto se mueve a un espacio de direcciones diferente de aquella donde fue creada).*
4. **Visibilidad:** *capacidad de restringir el acceso a atributos y servicios de un objeto. Particularmente importante en el diseño e implementación. (ej.: público / protegido / privado)*