Programación Orientada a Objetos

Si nos paramos a observar el mundo que nos rodea, podemos apreciar que casi todo está formado por objetos. Existen coches, edificios, sillas, mesas, semáforos, ascensores e incluso personas o animales. Todos ellos pueden ser considerados objetos, con una serie de características y comportamientos.

Los programas son el resultado de la búsqueda y obtención de una solución para un problema del mundo real.

Si redactamos los programas utilizando los mismos términos de nuestro mundo real, es decir, utilizando objetos, y no los términos del sistema o computadora donde se vaya a ejecutar, conseguiremos que éstos sean más legibles y, por tanto, más fáciles de modificar.

Esto es precisamente lo que pretende la Programación Orientada a Objetos (POO), en inglés OOP (Object Oriented Programming), establecer una serie de técnicas que permitan trasladar los problemas del mundo real a nuestro sistema informático.

Características diferenciales de la POO

Dentro de las distintas formas de hacer las cosas en programación, distinguimos dos paradigmas fundamentales:
— Programación Estructurada, se crean funciones y procedimientos que definen las acciones a realizar, y que posteriormente forman los programas.
La Programación Estructurada se centra en el conjunto de acciones a realizar en un programa, haciendo una división de procesos y datos.

— Programación Orientada a Objetos, considera los programas en términos de objetos y todo gira alrededor de ellos.
La Programación Orientada a Objetos es un sistema o conjunto de reglas que nos ayudan a descomponer la aplicación en objetos. A menudo se trata de representar las entidades y objetos que nos encontramos en el mundo real mediante componentes de una aplicación. Es decir, debemos establecer una correspondencia directa entre el espacio del problema y el espacio de la solución.

La Programación Orientada a Objetos aplica de otra forma diferente la técnica de programación "divide y vencerás”.

Beneficios.

— Comprensión. Los conceptos del problema se hayan reflejados en el código del programa, por lo que la mera lectura del código nos describe la solución del problema en el mundo real.
— Modularidad. Facilita la modularidad del código, al estar las definiciones de objetos en módulos o archivos independientes, hace que las aplicaciones estén mejor organizadas y sean más fáciles de entender.
— Fácil mantenimiento. Cualquier modificación en las acciones queda automáticamente reflejada en los datos, ya que ambos están estrechamente relacionados.

— Seguridad. La probabilidad de cometer errores se ve reducida, ya que no podemos modificar los datos de un objeto directamente, sino que debemos hacerlo mediante las acciones definidas para ese objeto.
— Reusabilidad. Los objetos se definen como entidades reutilizables, es decir, que los programas que trabajan con las mismas estructuras de información, pueden reutilizar las definiciones de objetos empleadas en otros programas, e incluso las acciones definidas sobre ellos.

Características.

— Abstracción. Es el proceso por el cual definimos las características más importantes de un objeto, sin preocuparnos de cómo se escribirán en el código del programa, simplemente lo definimos de forma general. En la Programación Orientada a Objetos la herramienta más importante para soportar la abstracción es la clase. Básicamente, una clase es un tipo de dato que agrupa las características comunes de un conjunto de objetos.
— Modularidad. Una vez que hemos representado el escenario del problema en nuestra aplicación, tenemos como resultado un conjunto de objetos software a utilizar. Este conjunto de objetos se crean a partir de una o varias clases. Cada clase se encuentra en un archivo diferente, por lo que la modularidad nos permite modificar las características de la clase que define un objeto, sin que esto afecte al resto de clases de la aplicación.
— Encapsulación. También llamada "ocultamiento de la información". La encapsulación o encapsulamiento es el mecanismo básico para ocultar la información de las partes internas de un objeto a los demás objetos de la aplicación. Con la encapsulación un objeto puede ocultar la información que contiene al mundo exterior, o bien restringir el acceso a la misma para evitar ser manipulado de forma inadecuada.
— Jerarquía. Mediante esta propiedad podemos definir relaciones de jerarquías entre clases y objetos.
- La generalización o especialización, también conocida como herencia, permite crear una clase nueva en términos de una clase ya existente (herencia simple) o de varias clases ya existentes (herencia múltiple).
- La agregación, también conocida como inclusión, permite agrupar objetos relacionados entre sí dentro de una clase.
— Polimorfismo. Esta propiedad indica la capacidad de que varias clases creadas a partir de una antecesora realicen una misma acción de forma diferente.

Objetos.

Un objeto de software es una representación de un objeto del mundo real, compuesto de una serie de características y un comportamiento específico. Un objeto es un conjunto de datos con las operaciones definidas para ellos. Los objetos tienen un estado y un comportamiento.

Características.

Los objetos tienen unas características fundamentales que los distinguen:
- Identidad. Es la característica que permite diferenciar un objeto de otro. De esta manera, aunque dos objetos sean exactamente iguales en sus atributos, son distintos

entre sí. Puede ser una dirección de memoria, el nombre del objeto o cualquier otro elemento que utilice el lenguaje para distinguirlos.

- Estado. El estado de un objeto viene determinado por una serie de parámetros o atributos que lo describen, y los valores de éstos.
- Comportamiento. Son las acciones que se pueden realizar sobre el objeto. En otras palabras, son los métodos o procedimientos que realiza el objeto.

Propiedades y métodos de los objetos

- Campos, Atributos o Propiedades: Parte del objeto que almacena los datos. También se les denomina Variables Miembro. Estos datos pueden ser de cualquier tipo primitivo (boolean, char, int, double, etc) o ser su vez ser otro objeto.
- Métodos o Funciones Miembro: Parte del objeto que lleva a cabo las operaciones sobre los atributos definidos para ese objeto.

La única forma de manipular la información del objeto es a través de sus métodos. Es decir, si queremos saber el valor de algún atributo, tenemos que utilizar el método que nos muestre el valor de ese atributo. De esta forma, evitamos que métodos externos puedan alterar los datos del objeto de manera inadecuada. Se dice que los datos y los métodos están encapsulados dentro del objeto.

Clases

Una clase es una descripción de un conjunto de objetos que comparten una estructura y un comportamiento común. Y a partir de la clase, se crean tantas "copias" o "instancias" como necesitemos. Esas copias son los objetos de la clase.

Las clases constan de datos y métodos que resumen las características comunes de un conjunto de objetos. Un programa informático está compuesto por un conjunto de clases, a partir de las cuales se crean objetos que interactúan entre sí.

Estructura de una clase

Una clase es una plantilla o prototipo donde se especifican:

- Los atributos comunes a todos los objetos de la clase.
- Los métodos que pueden utilizarse para manejar esos objetos.

Para declarar una clase en Java se utiliza la palabra reservada class. La declaración de una clase está compuesta por:

- Cabecera de la clase. La cabecera es un poco más compleja que como aquí definimos, pero por ahora sólo nos interesa saber que está compuesta por una serie de modificadores, en este caso hemos puesto public que indica que es una clase pública a la que pueden acceder otras clases del programa, la palabra reservada class y el nombre de la clase.
- Cuerpo de la clase. En él se especifican encerrados entre llaves los atributos y los métodos que va a tener la clase.

Constructor

Un constructor es un método especial con el mismo nombre de la clase y que no devuelve ningún valor tras su ejecución.
Cuando creamos un objeto debemos instanciarlo utilizando el constructor de la clase.

El método constructor tiene las siguientes particularidades:

- El constructor es invocado automáticamente en la creación de un objeto, y sólo esa vez.
- Los constructores no empiezan con minúscula, como el resto de los métodos, ya que se llaman igual que la clase y las clases empiezan con letra mayúscula.
- Puede haber varios constructores para una clase.
- Como cualquier método, el constructor puede tener parámetros para definir qué valores dar a los atributos del objeto.
- El constructor por defecto es aquél que no tiene argumentos o parámetros.
- Es necesario que toda clase tenga al menos un constructor.

Getters y Setters

Los Setters & Getters son métodos de acceso lo que indica que son siempre declarados públicos, y nos sirven para dos cosas:

- Setters: Del Inglés Set, que significa establecer, pues nos sirve para asignar un valor inicial a un atributo, pero de forma explícita, además el Setter nunca retorna nada (Siempre es void), y solo nos permite dar acceso público a ciertos atributos que deseemos el usuario pueda modificar.
- Getters: Del Inglés Get, que significa obtener, pues nos sirve para obtener (recuperar o acceder) el valor ya asignado a un atributo y utilizarlo para cierto método.

Ciclo de vida de los objetos

Las instancias u objetos tienen un tiempo de vida determinado. Cuando un objeto no se va a utilizar más en el programa, es destruido por el recolector de basura para liberar recursos que pueden ser reutilizados por otros objetos.
A la vista de lo anterior, podemos concluir que los objetos tienen un ciclo de vida, en el cual podemos distinguir las siguientes fases:

- Creación, donde se hace la reserva de memoria e inicialización de atributos.
- Manipulación, que se lleva a cabo cuando se hace uso de los atributos y métodos del objeto.
- Destrucción, eliminación del objeto y liberación de recursos.

Creación de un objeto

Veamos primero cómo declarar un objeto. Para la definición del tipo de objeto debemos emplear la siguiente instrucción:

<tipo> nombre_objeto;

Donde:

- tipo es la clase a partir de la cual se va a crear el objeto, y
- nombre_objeto es el nombre de la variable referencia con la cual nos referiremos al objeto.

Los tipos referenciados o referencias se utilizan para guardar la dirección de los datos en la memoria del ordenador.
Nada más crear una referencia, ésta se encuentra vacía. Cuando una referencia a un objeto no contiene ninguna instancia se dice que es una referencia nula, es decir, que contiene el valor null. Esto quiere decir que la referencia está creada pero que el objeto no está instanciado todavía, por eso la referencia apunta a un objeto inexistente llamado “nulo".

Los nombres de la clase empiezan con mayúscula, como String, y los nombres de los objetos con minúscula, como mensaje, así sabemos qué tipo de elemento utilizando.

Una vez declarado el objeto es necesario instanciarlo. Para ello utilizamos la orden new con la siguiente sintaxis:
nombre_objeto = new <Constructor_de_la_Clase>([<par1>, <par2>, ..., <parN>]);
Donde:

- **nombre_objeto** es el nombre de la variable referencia con la cual nos referiremos al objeto,
- **new** es el operador para crear el objeto,
- **Constructor_de_la_Clase** es un método especial de la clase, que se llama igual que ella, y se encarga de inicializar el objeto, es decir, de dar unos valores iniciales a sus atributos, y **par1-parN**, son parámetros que puede o no necesitar el constructor para dar los valores iniciales a los atributos del objeto.

Destrucción de un objeto. Recolector de basura.

Cuando un objeto deja de ser utilizado, es necesario liberar el espacio de memoria y otros recursos que poseía para poder ser reutilizados por el programa. A esta acción se le denomina destrucción del objeto.

En Java la destrucción de objetos corre a cargo del recolector de basura (garbage collector). Es un sistema de destrucción automática de objetos que ya no son utilizados.

Lo que se hace es liberar una zona de memoria que había sido reservada previamente mediante el operador new. Esto evita que los programadores tengan que preocuparse de realizar la liberación de memoria.

El recolector de basura se ejecuta en modo segundo plano y de manera muy eficiente para no afectar a la velocidad del programa que se está ejecutando. Lo que hace es que periódicamente va buscando objetos que ya no son referenciados, y cuando encuentra alguno lo marca para ser eliminado. Después los elimina en el momento que considera oportuno.

Justo antes de que un objeto sea eliminado por el recolector de basura, se ejecuta su método finalize(). Si queremos forzar que se ejecute el proceso de finalización de todos los objetos del programa podemos utilizar el método runFinalization() de la clase System.

Utilización de métodos
Los métodos, junto con los atributos, forman parte de la estructura interna de un objeto. Los métodos contienen la declaración de variables locales y las operaciones que se pueden realizar para el objeto, y que son ejecutadas cuando el método es invocado. Se definen en el cuerpo de la clase y posteriormente son instanciados para convertirse en métodos instancia de un objeto.

Parámetros y valores devueltos

Los métodos se pueden utilizar tanto para consultar información sobre el objeto como para modificar su estado. La información consultada del objeto se devuelve a través de lo que se conoce como valor de retorno, y la modificación del estado del objeto, o sea, de sus atributos, se hace mediante la lista de parámetros.

El valor de retorno es la información que devuelve un método tras su ejecución.

La lista de argumentos en la llamada a un método debe coincidir en número, tipo y orden con los parámetros del método, ya que de lo contrario se produciría un error de sintaxis.

Métodos estáticos

Los métodos estáticos son aquellos métodos definidos para directamente, sin necesidad de crear un objeto de dicha clase. También se llaman métodos de clase.
Para llamar a un método estático utilizaremos:

- El nombre del método, si lo llamamos desde la misma clase en la que se encuentra definido.
- El nombre de la clase, seguido por el operador punto (.) más el nombre del método estático, si lo llamamos desde una clase distinta a la que se encuentra definido:

Los métodos estáticos no afectan al estado de los objetos instanciados de la clase (variables instancia), y suelen utilizarse para realizar operaciones comunes a todos los objetos de la clase.

El operador this

Los constructores y métodos de un objeto suelen utilizar el operador this. Este operador sirve para referirse a los atributos de un objeto cuando estamos dentro de él.

Librería de objetos.

Conforme nuestros programas se van haciendo más grandes, el número de clases va creciendo. Meter todas las clases en único directorio no ayuda a que estén bien organizadas, lo mejor es hacer grupos de clases, de forma que todas las clases que estén relacionadas o traten sobre un mismo tema estén en el mismo grupo.

Un paquete de clases es una agrupación de clases que consideramos que están relacionadas entre sí o tratan de un tema común.
Las clases de un mismo paquete tienen un acceso privilegiado a los atributos y métodos de otras clases de dicho paquete. Es por ello por lo que se considera que los paquetes son también, en cierto modo, unidades de encapsulación y ocultación de información.

Paquetes

Para organizar mejor las cosas, un paquete, en vez de clases, también puede contener otros paquetes. Es decir, podemos hacer subpaquetes de los paquetes y subpaquetes

de los subpaquetes y así sucesivamente. Esto permite agrupar paquetes relacionados en un paquete más grande.

Sentencia import

Cuando queremos utilizar una clase que está en un paquete distinto a la clase que estamos utilizando, se suele utilizar la sentencia import.