Conceptos de OOP de Java

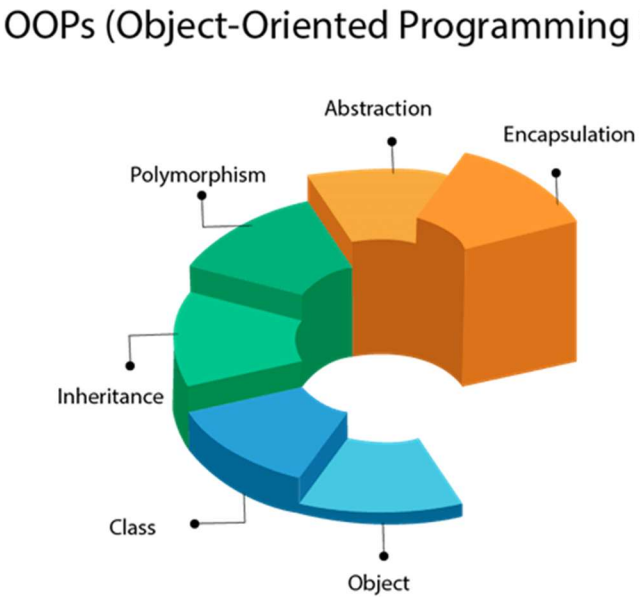
La programación orientada a objetos es un paradigma que proporciona muchos conceptos, como **herencia** , **enlace de datos** , **polimorfismo** , etc.

El objetivo principal de la programación orientada a objetos es implementar entidades del mundo real, a través de los componentes del lenguaje como son: objeto, clases, abstracción, herencia, polimorfismo, etc.

POO (sistema de programación orientado a objetos)

Un Objeto pretende ser una entidad del mundo real, como un coche, una cuenta bancaria, etc. **La programación orientada a objetos** es una metodología o paradigma para diseñar un programa utilizando clases y objetos. Simplifica el desarrollo y mantenimiento de software al proporcionar algunos conceptos:

- Objeto
- Clase
- Herencia
- Polimorfismo
- Abstracción
- Encapsulamiento



Además de estos conceptos, hay algunos otros términos que se utilizan en el diseño orientado a objetos:

- Acoplamiento
- Cohesión
- Asociación
- Agregación
- Composición

Objeto

Un objeto se puede definir como una instancia de una clase. Un objeto contiene una dirección y ocupa algo de espacio en la memoria.

Los objetos pueden comunicarse sin conocer los detalles de los datos o códigos de los demás. Lo único necesario es el tipo de mensaje aceptado y el tipo de respuesta devuelta por los objetos.

Ejemplo: un perro es un objeto porque tiene estados como color, nombre, raza, etc., así como comportamientos como mover la cola, ladrar, comer, etc.

Clase

La colección de objetos se llama clase. Es una entidad lógica.

Una clase también se puede definir como un mapa a partir del cual puede crear un objeto individual. La clase no consume espacio en memoria.

Herencia

Cuando una clase adquiere todas las propiedades y comportamientos de una clase padre, se conoce como herencia. Los objetos creados de estas clases también adquieren dichos comportamientos, Proporciona reutilización de código. Se utiliza para lograr el polimorfismo de tiempo de ejecución.

Polimorfismo

Si *una tarea se realiza de diferentes formas*, se conoce como polimorfismo.

En Java, utilizamos la sobrecarga de métodos y la anulación de métodos para lograr el polimorfismo. Por ejemplo: actualizar una cuenta bancaria, solo su saldo, su saldo y su numero de IBAN, su propietario… Todas son tareas actualizar, pero se comportan de manera de diferente

Abstracción

Ocultar detalles internos y mostrar funcionalidad se conoce como abstracción. Por ejemplo, podríamos tener una clase comportamiento que recogiera metodos que desconocemos de una llamada telefónica, como establecer la conexión o enviar la voz, y nosotros ejecutaríamos la parte que si conocemos, como marcar el numero.

En Java, utilizamos la clase abstracta y la interfaz para lograr la abstracción.

Encapsulamiento

El código de envoltura y los datos juntos en una sola unidad (clase) se conocen como encapsulación . Es como considerar un contenedor en el que tenemos cosas mezcladas.

Una clase java es el ejemplo de encapsulación. El nivel de encapsulación dependerá de los modificadores de visibilidad que tenga la clase. Java bean es la clase completamente encapsulada porque todos los miembros de datos son privados.

Acoplamiento

El acoplamiento se refiere al conocimiento o información o dependencia de otra clase. Surge cuando las clases se conocen entre sí. Si una clase tiene la información detallada de otra clase, existe un fuerte acoplamiento. En Java, utilizamos modificadores privados, protegidos y públicos para mostrar el nivel de visibilidad de una clase, método y campo. Se suele utilizar el uso de interfaces para reducir este problema. Lo ideal es conseguir un acoplamiento lo más bajo posible o dicho de otra manera, que las clases sean lo más independiente posible de otras.

Cohesión

La cohesión se refiere al nivel de un componente que realiza una única tarea bien definida. Una sola tarea bien definida se realiza mediante un método altamente cohesivo. El método débilmente cohesivo dividirá la tarea en partes separadas. El paquete java.io es un paquete altamente coherente porque tiene clases e interfaz relacionadas con E / S. Sin embargo, el paquete java.util es un paquete débilmente cohesivo porque tiene clases e interfaces no relacionadas.

Asociación

La asociación representa la relación entre los objetos. Aquí, un objeto puede asociarse con uno o muchos objetos. Puede haber cuatro tipos de asociación entre los objetos:

- Uno a uno
- Uno a muchos
- Muchos a uno, y
- Muchos a muchos

Comprendamos la relación con ejemplos. Un país puede tener un primer ministro (uno a uno), y un primer ministro puede tener muchos ministros (uno a muchos). Además, muchos parlamentarios pueden tener un primer ministro (muchos a uno), y muchos ministros pueden tener muchos departamentos (muchos a muchos).

La asociación puede ser unidireccional o bidireccional.

Agregación

La agregación es una forma de lograr la Asociación. La agregación representa la relación donde un objeto contiene otros objetos como parte de su estado. Representa la relación débil entre los objetos. También se denomina como una relación *has-a* en Java. Como, la herencia representa la relación *is-a* . Es otra forma de reutilizar objetos.

Composición

La composición también es una forma de lograr la Asociación. La composición representa la relación donde un objeto contiene otros objetos como parte de su estado. Existe una fuerte relación entre el objeto contenedor y el objeto dependiente. Es el estado en el que los objetos que contienen no tienen una existencia independiente. Si elimina el objeto primario, todos los objetos secundarios se eliminarán automáticamente.