Programación Orientada a Objetos

La programación orientada a objetos o OOP se refiere a lenguajes que usan objetos en la programación, usan objetos como fuente principal para implementar lo que sucederá en el código. Los objetos son vistos por el espectador o usuario, realizando tareas asignadas por usted. La programación orientada a objetos tiene como objetivo implementar entidades del mundo real como herencia, ocultación, polimorfismo, etc. en la programación. El objetivo principal de OOP es unir los datos y las funciones que operan en ellos para que ninguna otra parte del código pueda acceder a estos datos excepto esa función.

Declaración del método, consta de seis componentes:

Modificador de acceso: Define el tipo de acceso del método, es decir, desde dónde se puede acceder en su aplicación. En Java, hay 4 tipos de especificadores de acceso:

- **Public:** Accesible en todas las clases en su aplicación.
- **Protected:** Accesible dentro del paquete en el que se define y en su (s) subclase(s) (incluidas las subclases declaradas fuera del paquete).
- **Private:** Accesible solo dentro de la clase en la que está definido.

El tipo de devolución: el tipo de datos del valor devuelto por el método o nulo si no devuelve un valor.

Nombre del método: las reglas para los nombres de los campos también se aplican a los nombres de los métodos, pero la convención es un poco diferente.

Lista de parámetros: lista separada por comas de los parámetros de entrada que se definen, precedidos por su tipo de datos, dentro de los paréntesis incluidos. Si no hay parámetros, debe usar paréntesis vacíos ().

Lista de excepciones: las excepciones que espera que arroje el método. Puede especificar estas excepciones.

Cuerpo del método: es el bloque de código, encerrado entre llaves, que debe ejecutar para realizar las operaciones previstas.

Paso de mensajes: los objetos se comunican entre sí mediante el envío y la recepción de información entre sí. Un mensaje para un objeto es una solicitud de ejecución de un procedimiento y, por lo tanto, invocará una función en el objeto receptor que genera los resultados deseados. El paso de mensajes implica especificar el nombre del objeto, el nombre de la función y la información que se enviará.

Los conceptos de OOPS son los siguientes:

1. Clase
2. Objeto
3. Método y paso de método
4. Pilares de OOP
    - Abstracción
    - Encapsulación
    - Herencia
    - Polimorfismo
        - Polimorfismo en tiempo de compilación
        - Polimorfismo en tiempo de ejecución

Una clase es un proyecto o prototipo definido por el usuario a partir del cual se crean objetos. Representa el conjunto de propiedades o métodos que son comunes a todos los objetos de un tipo. Usando clases, puede crear múltiples objetos con el mismo comportamiento en lugar de escribir su código varias veces. Esto incluye clases para objetos que ocurren más de una vez en su código. En general, las declaraciones de clase pueden incluir estos componentes en orden:

Modificadores: una clase puede ser pública o tener acceso predeterminado (consulte esto para obtener más detalles).

Nombre de la clase: El nombre de la clase debe comenzar con la letra inicial en mayúscula por convención.

Superclase (si existe): el nombre del padre de la clase (superclase), si existe, precedido por la palabra clave extends. Una clase solo puede extender (subclase) un padre.

Interfaces (si las hay): una lista separada por comas de las interfaces implementadas por la clase, si las hay, precedida por la palabra clave implements. Una clase puede implementar más de una interfaz.

Cuerpo: el cuerpo de la clase está rodeado por llaves, {}.

Un objeto es una unidad básica de la Programación Orientada a Objetos que representa entidades de la vida real. Un programa típico de Java crea muchos objetos que, como sabe, interactúan invocando métodos. Los objetos son los que ejecutan su código, son la parte de su código visible para el espectador/usuario. Un objeto consiste principalmente en:

Estado: Está representado por los atributos de un objeto. También refleja las propiedades de un objeto.

Comportamiento: Está representado por los métodos de un objeto. También refleja la respuesta de un objeto a otros objetos.

Identidad: Es un nombre único dado a un objeto que le permite interactuar con otros objetos.

Método: un método es una colección de declaraciones que realizan alguna tarea específica y devuelven el resultado a la persona que llama. Un método puede realizar alguna tarea específica sin devolver nada. Los métodos nos permiten reutilizar el código sin volver a escribirlo, por lo que se consideran ahorradores de tiempo. En Java, cada método debe ser parte de alguna clase, que es diferente de lenguajes como C, C++ y Python.

Analicemos ahora los 4 pilares de los OOP:

Pilar 1: Abstracción

La abstracción de datos es la propiedad en virtud de la cual sólo se muestran al usuario los datos esenciales. Las unidades triviales o no esenciales no se muestran al usuario. Ej: Un automóvil se ve como un automóvil en lugar de sus componentes individuales.

La abstracción de datos también se puede definir como el proceso de identificar solo las características requeridas de un objeto, ignorando los detalles irrelevantes. Las propiedades y comportamientos de un objeto lo diferencian de otros objetos de tipo similar y también ayudan a clasificar/agrupar el objeto.

Considere un ejemplo de la vida real de un hombre que conduce un automóvil. El hombre solo sabe que presionar los aceleradores aumentará la velocidad del automóvil o que aplicar los frenos detendrá el automóvil, pero no sabe cómo, al presionar el acelerador, la velocidad en realidad aumenta. No conoce el mecanismo interno del automóvil o la implementación de los aceleradores, frenos, etc. en el automóvil. Esto es lo que es la abstracción.

En Java, la abstracción se logra mediante interfaces y clases abstractas. Podemos lograr un 100% de abstracción usando interfaces.

Pilar 2: Encapsulación

Se define como la agrupación de datos en una sola unidad. Es el mecanismo que une el código y los datos que manipula. Otra forma de pensar en la encapsulación es que es un escudo protector que evita que el código que se encuentra fuera de este escudo acceda a los datos.

Técnicamente, en la encapsulación, las variables o los datos de una clase se ocultan de cualquier otra clase y solo se puede acceder a ellos a través de cualquier función miembro de la clase en la que se declaran.

En la encapsulación, los datos de una clase se ocultan de otras clases, lo que es similar a lo que hace la ocultación de datos. Por lo tanto, los términos «encapsulación» y «ocultación de datos» se usan indistintamente.

La encapsulación se puede lograr declarando todas las variables de una clase como privadas y escribiendo métodos públicos en la clase para establecer y obtener los valores de las variables.

Pilar 3: Herencia

La herencia es un pilar importante de OOP (Programación Orientada a Objetos). Es el mecanismo en Java por el cual una clase puede heredar las características (campos y métodos) de otra clase.

Discutamos algunas terminologías importantes de uso frecuente:

Superclase: la clase cuyas características se heredan se conoce como superclase (también conocida como clase base o padre).

Subclase: La clase que hereda de la otra clase se conoce como subclase (también conocida como clase derivada, extendida o secundaria). La subclase puede agregar sus propios campos y métodos además de los campos y métodos de la superclase.

Reutilización: la herencia admite el concepto de «reutilización», es decir, cuando queremos crear una nueva clase y ya existe una clase que incluye parte del código que queremos, podemos derivar nuestra nueva clase de la clase existente. Al hacer esto, estamos reutilizando los campos y métodos de la clase existente.

Pilar 4: Polimorfismo

Se refiere a la capacidad de los lenguajes de programación orientados a objetos para diferenciar entre entidades con el mismo nombre de manera eficiente. Esto lo hace Java con la ayuda de la firma y declaración de estas entidades.