

Programación orientada a objetos

1. Introducción

La Programación Orientada a Objetos (POO) es un paradigma de programación que ha supuesto un cambio radical respecto a la programación tradicional. El desarrollo de software gira entorno a los objetos, encapsulando métodos y variables en módulos llamados **objetos**, los cuales pueden modificar de forma indirecta sus variables mediante operaciones. Los objetos se utilizan para construir programas.

Ampliamente extendido gracias a:

- Alta reutilización de código
- Facilidad de mantenimiento
- Representación más sencilla de estructuras (son objetos del mundo real)

2. Conceptos básicos

Objetos

- Objetos: contienen tanto los datos como las operaciones para manipularlos
 - Estructura de un objeto
 1. Propiedades o atributos
 2. Métodos o procedimientos
 3. Eventos o mensajes
 - Atributos: Describen el aspecto que va a tener el objeto. Definen el estado de un objeto
 - Visibilidad de variables
 1. Públicos: accesibles por cualquier objeto
 2. Privados: solo accesibles desde el propio objeto
 3. Protegidos: solo accesibles por el propio objeto y sus hijos
 - Tipos de variables
 1. Primitivo: están definidas de un tipo concreto (carácter, booleano, etc)
 2. Referencia: variables que hacen referencia a objetos de una determinada clase
 3. Miembros: variables definidas en una clase pero fuera de cualquier método
 4. Locales: se definen dentro de un método o en un bloque entre llaves

Métodos

Los métodos son funciones asociadas a un objeto. Estos pueden ser públicos, privados o protegidos, mutadores o observadores, constructores o destructores. Los primeros modifican algún atributo del objeto, los segundos solo obtienen información, los **constructores** se activan al crearse el objeto y definen los valores iniciales de las propiedades, mientras que los **destructores** se activan al eliminarse el objeto y suelen usarse para liberar recursos.

Mensajes

- Los objetos se comunican entre sí mediante mensajes: nombre de objeto + método + parámetros
- Los mensajes conectan al objeto con el mundo exterior

Clases

Las clases son una estructura estática que define los atributos y métodos de un conjunto de objetos que pertenecen a la misma familia. Estos objetos, también conocidos como **instancias**, se crean durante la ejecución del programa.

Existen **clases abstractas**, aquellas que no tendrán instancias. Para facilitar el desarrollo y reutilizar clases, existen las **bibliotecas de clases**, conjunto de clases disponibles en un mismo lugar para su uso.

3. Propiedades

- Herencia
- Encapsulación
- Polimorfismo
- Abstracción

4. Herencia

La **herencia** es un mecanismo de programación orientada a objetos que permite definir nuevas clases basadas en clases existentes. Esto permite reutilizar el código al añadir y/o redefinir nuevas variables y métodos.

La **herencia múltiple** se realiza mediante **interfaces**, que contienen declaraciones de métodos sin definición y constantes, y que pueden ser implementadas por una o varias clases.

Existen varios **tipos de herencia** que se clasifican en términos de los niveles de herencia, la cantidad de clases de las que se hereda, la estructura con la que se hereda y la selectividad con la que se hereda. Estos tipos incluyen herencia raíz, intermedios y terminales, simple y múltiple, estricta y no estricta y selectiva y no selectiva.

5. Encapsulación

La programación orientada a objetos permite ocultar información que no es pertinente o necesaria para otro objeto. Esto se puede implementar mediante permisos aplicando diferentes niveles de acceso.

Por ejemplo, en Java, las clases se pueden declarar como públicas o package, mientras que las variables pueden ser public, private, protected o package. Esto permite controlar el acceso y evitar el uso inadecuado de los datos.

6. Polimorfismo

- Una referencia a un objeto de una clase puede hacer referencia a cualquiera de sus clases derivadas
- Solo se pueden utilizar los métodos de la clase con que han sido definidas las referencias
- Lo mismo se puede hacer con interfaces, siempre que las clases que se asocian a la referencia implementen dicha interfaz.
- En este caso solo se pueden ejecutar métodos de la interfaz

7. Abstracción

- Una clase abstracta es una de la que no se pueden crear objetos

- Sirve para que otras clases hereden de ella
- Los métodos son obligatoriamente abstractos
- Subclases de estas clases heredan de esta clase abstracta

8. Otras

La **persistencia** permite que los objetos mantengan su valor cuando finaliza la ejecución del programa, para lo cual se deben serializar en un flujo de caracteres y almacenar en un disco o base de datos.

a extensibilidad y la reutilización permiten a los programadores añadir nuevas funcionalidades a los objetos existentes o reutilizarlos en otros proyectos.

9. Agrupación de clases

- Varias clases se pueden agrupar en un package.
- Existen packages predefinidos incluidos en el lenguaje (API de Java)
- Usuario puede crear sus propios packages con clases que estén relacionadas

10. Metodología

11. Diseño orientado a objetos

- Método de diseño
 - Identificar objetos
 - Identificar operaciones
 - Establecer visibilidad
 - Establecer interfaz
 - Implementación

12. Lenguajes de programación orientado a objetos

Existen dos tipos de lenguajes orientados a objetos: **puros** e híbridos. Los lenguajes de programación puramente orientados a objetos, como **C++** y Java, están diseñados para construir aplicaciones orientadas a objetos desde el principio.

Los lenguajes **híbridos**, como Python, permiten programar tanto en un estilo orientado a objetos como en un estilo de programación de procedimientos. Esto les permite a los desarrolladores aprovechar las ventajas de ambos enfoques.