**Programação Orientada a Objetos**

Programação orientada a objetos (POO) é uma metodologia de programação adequada ao desenvolvimento de sistemas de grande porte, provendo modularidade e reusabilidade. A POO introduz uma abordagem na qual o programador visualiza seu programa em execução como uma coleção de objetos cooperantes que se comunicam através de mensagens. Cada um dos objetos é uma instância de uma classe e todas as classes formam uma hierarquia de classes unidas via relacionamento de herança.

Principais Pilares de POO

1. Encapsulamento
2. Herança
3. Abstração
4. Polimorfismo
5. Encapsulamento

O encapsulamento é um conceito fundamental na POO, que visa controlar o acesso aos atributos e métodos de um objeto. Isso é alcançado através da definição de níveis de visibilidade, como público, protegido e privado.

1. Herança

A herança é um conceito que permite a criação de hierarquias de classes, onde uma classe mais específica pode herdar os atributos e métodos de uma classe mais genérica. Isso permite a reutilização de código, pois características comuns podem ser definidas em uma classe base, e classes derivadas podem estender ou personalizar esse comportamento. A herança reflete a relação "é-um" entre classes, em que uma classe derivada é um tipo mais específico da classe base. Esse pilar contribui para a economia de tempo na codificação, pois evita a duplicação de código e promove uma estrutura organizada e hierárquica.

1. Abstração

A abstração envolve a representação de objetos do mundo real em modelos simplificados no contexto de um programa. É o processo de identificar as características essenciais de um objeto e ignorar os detalhes menos relevantes. Através da abstração, podemos criar classes que representam conceitos abstratos, como "Carro" ou "Cliente", e definir seus atributos e métodos relevantes. Esse pilar permite a criação de modelos que se assemelham às entidades do mundo real, facilitando a compreensão e a representação do sistema.

1. Polimorfismo

O polimorfismo se refere à capacidade de um objeto executar diferentes comportamentos com base no contexto em que é utilizado. Isso significa que objetos de classes diferentes podem responder a uma mesma mensagem (chamada de método) de maneiras distintas. O polimorfismo contribui para a flexibilidade do código, permitindo que diferentes classes compartilhem uma mesma interface e, ao mesmo tempo, implementem comportamentos específicos.

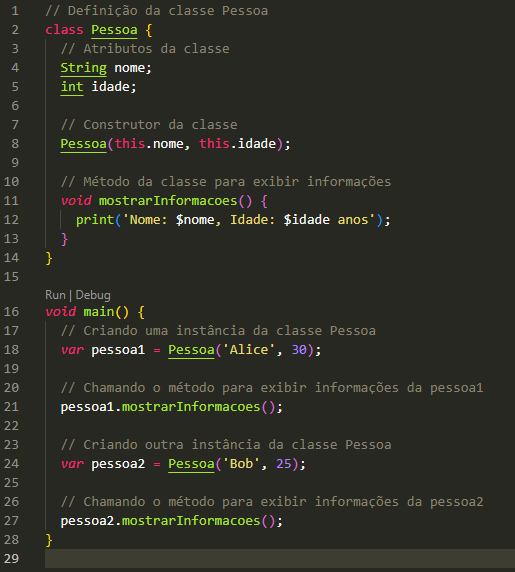
**Dart**

Programação Orientada a Objetos em Dart

A programação orientada a objetos (POO) é uma metodologia fundamental para o desenvolvimento de aplicativos em Dart. Neste contexto, Dart suporta todos os pilares essenciais da POO.

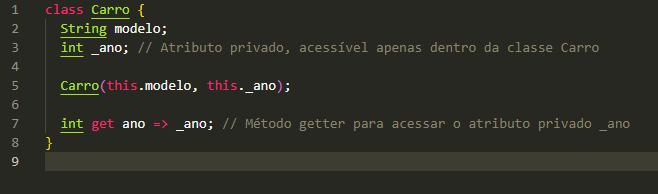
Classe

Em Dart todo objeto é uma instância de uma classe, e todas as classes, exceto Null, descendem de Object. A herança baseada em mixin significa que, embora cada classe (exceto a classe superior, Object?) tenha exatamente uma superclasse, o corpo de uma classe pode ser reutilizado em várias hierarquias de classes. Métodos de extensão são uma maneira de adicionar funcionalidade a uma classe sem alterar a classe ou criar uma subclasse. Modificadores de classe permitem controlar como as bibliotecas podem subtipar uma classe.



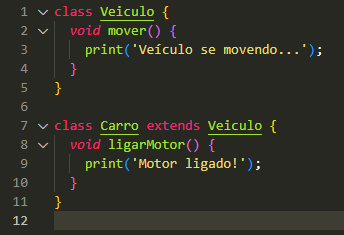
1. Encapsulamento em Dart:

Em Dart, o encapsulamento é alcançado através da definição de níveis de visibilidade para membros de uma classe. Os membros podem ser marcados como públicos, protegidos ou privados, usando os modificadores public, protected e private, respectivamente.



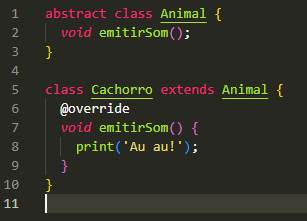
1. Herança:

Dart suporta herança de classes, permitindo que uma classe herde atributos e métodos de outra classe. Para herdar de uma classe, use a palavra-chave extends.



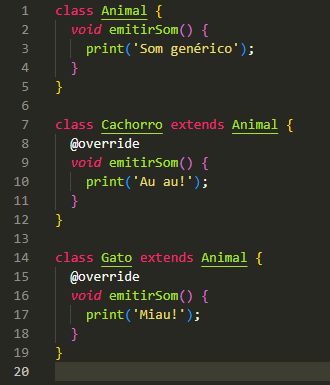
1. Abstração:

Dart permite criar classes abstratas usando a palavra-chave abstract. Classes abstratas não podem ser instanciadas diretamente e geralmente servem como modelos para outras classes.



1. Polimorfismo:

O polimorfismo em Dart é alcançado através da sobrescrita de métodos em classes derivadas. Isso permite que objetos de diferentes classes respondam de maneira diferente a métodos com o mesmo nome.



# Referências

Dart. (2023). *https://dart.dev/*. Obtido em 04 de Marco de 2024, de https://dart.dev/: https://dart.dev/language

Kamienski, P. C. (Julho de 1996). Introducão ao paradigma de orientacão a objectos. *Introducão ao paradigma de orientacão a objectos*, pp. 7-18.