



SENAI *Serviço Nacional
de Aprendizagem
Industrial*



Programação Orientada a Objetos (POO) com Dart

Prof. Mauro Andrade

Curso Técnico de Desenvolvimento de Sistemas

O que é POO?

- Paradigma de programação baseado em **objetos**.
- Cada objeto representa **entidades do mundo real**.
- Objetivo: organizar o código e **facilitar manutenção e reuso**.
- Exemplo: Um “Carro” tem atributos (cor, modelo, ano) e métodos (ligar, acelerar, frear).

Existem outros paradigmas?

Paradigma Imperativo

- **Ideia principal:** O programador **diz passo a passo o que o computador deve fazer.**
- **Características:** Baseia-se em comandos e estruturas de controle como if, for, while.

Paradigma Estruturado

- **Subtipo do imperativo**, mas com foco em **organizar o código em blocos bem definidos** (funções, laços, condicionais).

Conceitos Fundamentais

Base da POO: forma de organizar o código em torno de **objetos**, que representam entidades do mundo real.

Os quatro pilares são:

- **Abstração** (simplificar o mundo real – ignora detalhes)
 - **Encapsulamento** (Proteger os dados do objeto)
 - **Herança** (A classe “filha” herda atributos e métodos da classe “pai”.)
 - **Polimorfismo** (Muitas formas)
-
- Para entender melhor os 4 pilares, precisamos antes estudar os conceitos Fundamentais: Classe, Objeto, Atributos e Métodos.

Conceitos Fundamentais

- Classe: modelo ou molde do objeto.
- Objeto: instância (exemplo concreto) da classe.
- Atributos: características do objeto.
- Métodos: ações que o objeto executa.

Criando uma Classe em Dart

```
class Carro {  
  String modelo;  
  int ano;  
  
  void ligar() {  
    print("O carro $modelo está ligado!");  
  }  
}
```

class → define uma classe.
String modelo e int ano → atributos.
ligar() → método.

Criando um Objeto

```
void main() {  
    var carro1 = Carro();  
    carro1.modelo = "Fusca";  
    carro1.ano = 1980;  
  
    carro1.ligar();  
}
```

class → define uma classe.
String modelo e int ano → atributos.
ligar() → método.

Saida: O carro Fusca está ligado!

Exercícios - Classe

- Crie uma classe pessoa com atributos:
 - Nome
 - Idade
 - Profissao
- Instancie 3 objetos, e coloque e preencha os atributos com dados fictícios.
- Mostre os dados dos 3 objetos.

Exercícios - Classe

- Crie uma classe moto com atributos:
 - Ano
 - Fabricante
 - Cor
- Na classe moto crie 2 métodos:
 - Acelerar -> exibe “Acelerando!”
 - Buzinar -> exibir “Biiiiiiii” ou “Buzinando”
- Instancie 3 objetos, e coloque e preencha os atributos com dados fictícios.
- Mostre os dados e chame os metodos dos 3 objetos.

Construtores

```
class Pessoa {  
    String nome;  
    int idade;  
  
    Pessoa(this.nome, this.idade);  
}  
  
void main() {  
    var p = Pessoa("Maria", 25);  
    print("${p.nome} tem ${p.idade} anos.");  
}
```

Construtores permitem **inicializar atributos** ao criar o objeto.

Encapsulamento

```
class Conta {  
    double _saldo = 0;  
  
    double get saldo => _saldo;  
    void depositar(double valor) => _saldo  
    += valor;  
}
```

- Controla o **acesso aos dados internos** de um objeto.
- Usa **getters** e **setters** para manipular atributos privados.

Herança

```
class Animal {  
    void comer() => print("Comendo...");  
}
```

```
class Cachorro extends Animal {  
    void latir() => print("Au au!");  
}
```

```
void main() {  
    var dog = Cachorro();  
    dog.comer();  
    dog.latir();  
}
```

Permite que uma classe herde características de outra.

Polimorfismo

```
class Forma {  
    void desenhar() => print("Desenhando  
uma forma");  
}
```

```
class Circulo extends Forma {  
    @override //opcional  
    void desenhar() => print("Desenhando  
um círculo");  
}
```

- Permite que um mesmo método tenha comportamentos diferentes.
- O método desenhar() se comporta de forma diferente dependendo da classe.

Abstração

```
abstract class Pagamento {  
    void processar();  
}  
  
class Cartao implements Pagamento {  
    void processar() => print("Pagamento  
com cartão");  
}
```

- Foca no essencial, ocultando detalhes de implementação.

Benefícios da POO

- Reutilização de código
- Facilidade de manutenção
- Organização e modularização
- Maior clareza no raciocínio lógico

- Foca no essencial, ocultando detalhes de implementação.

Prática Proposta

Crie uma classe Aluno com:

- Atributos: nome, nota1, nota2
- Método: calcularMedia()
- Exiba a média do aluno.

Desafio Extra

Crie as classes:

- Funcionario (nome, cargo, salário)
- Gerente herda de Funcionario e tem bônus.
- Mostre o salário total do gerente.