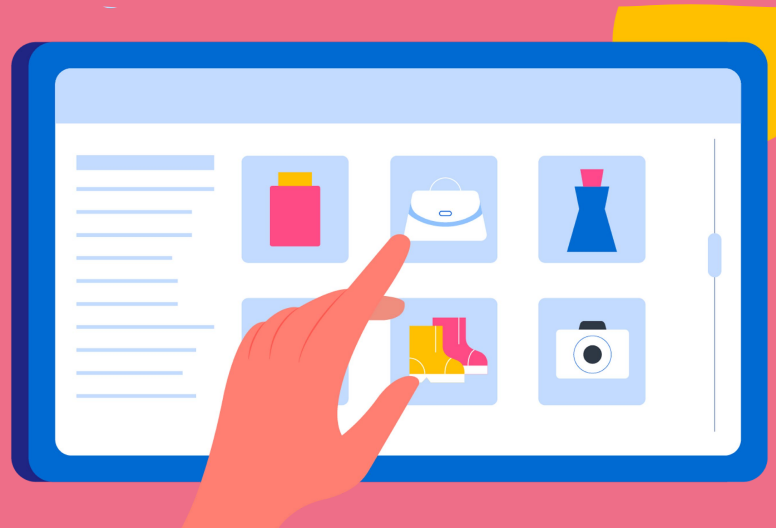
FIAP

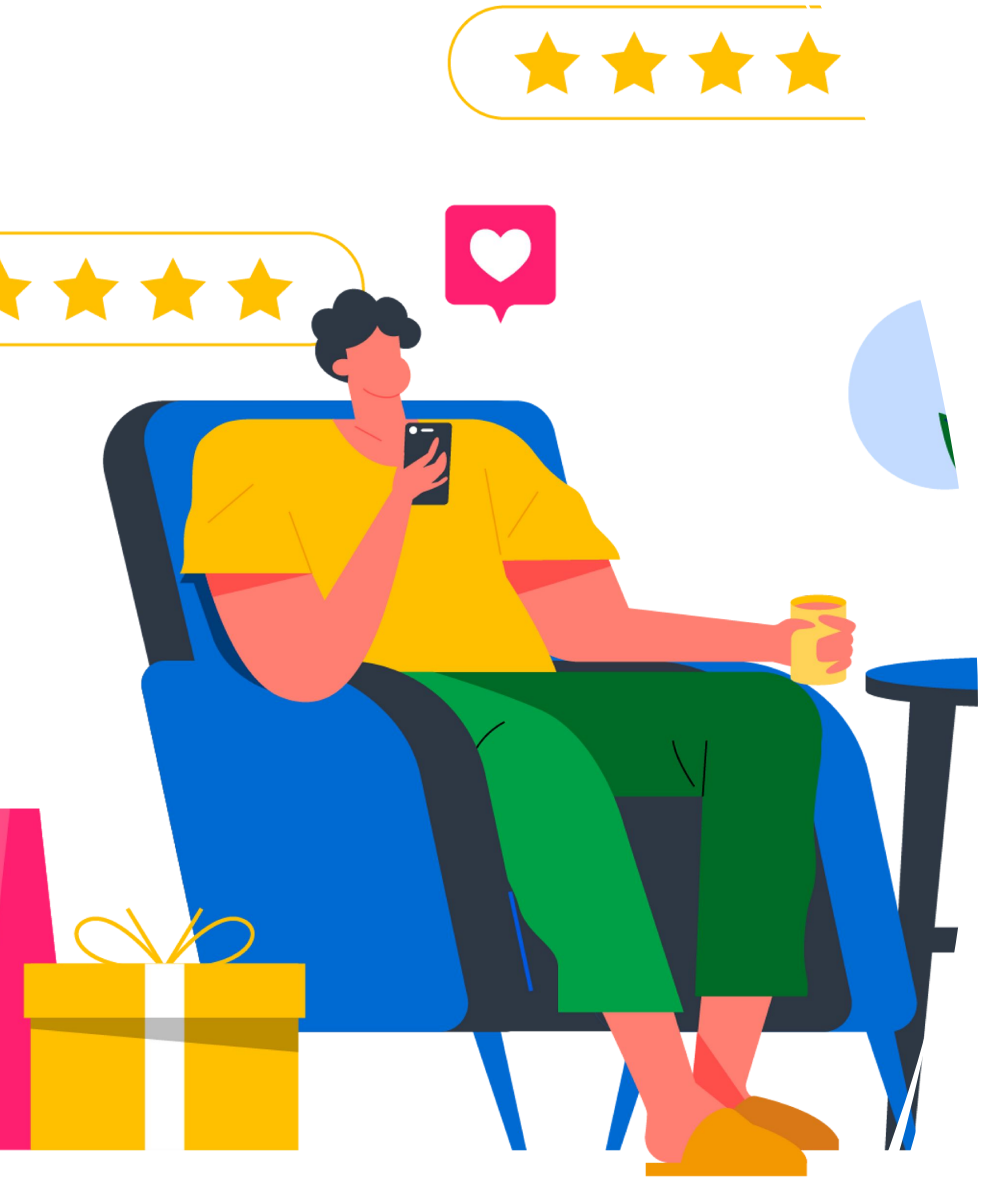
POO - Dart

Programação Orientada a Objetos



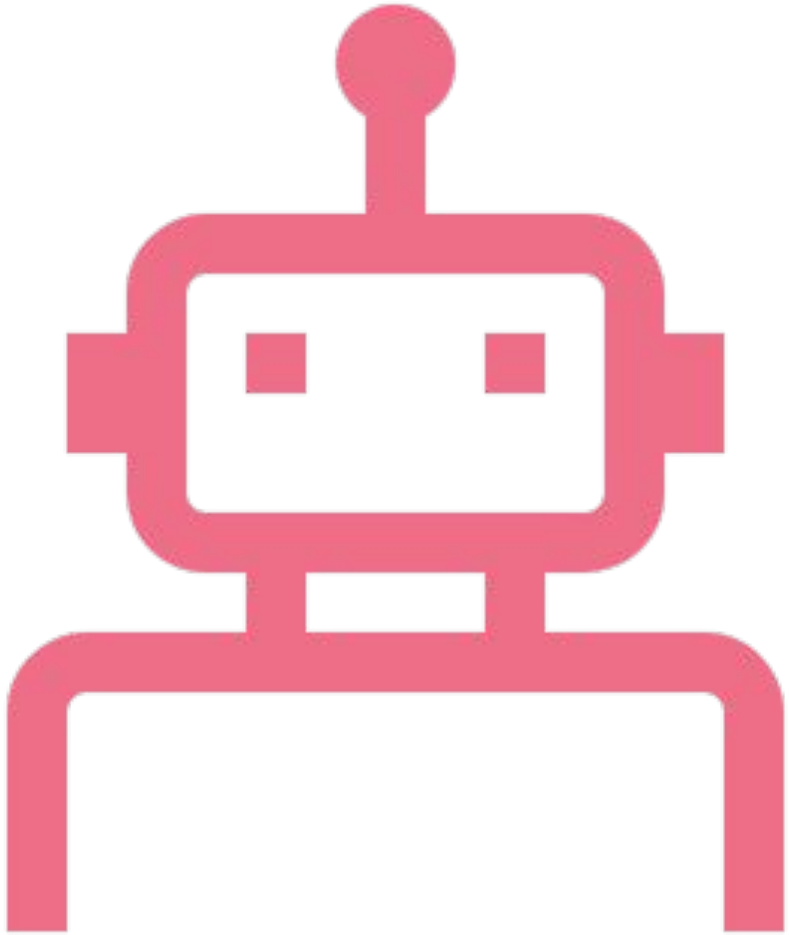
O que é?

- O paradigma de orientação a objetos em Dart traz boas vantagens, como a reutilização, a legibilidade e manutenibilidade do código, a natural modularização e a produção de código mais acessível, já que as estruturas criadas geralmente representam aspectos também existentes no mundo real.



CLASSES

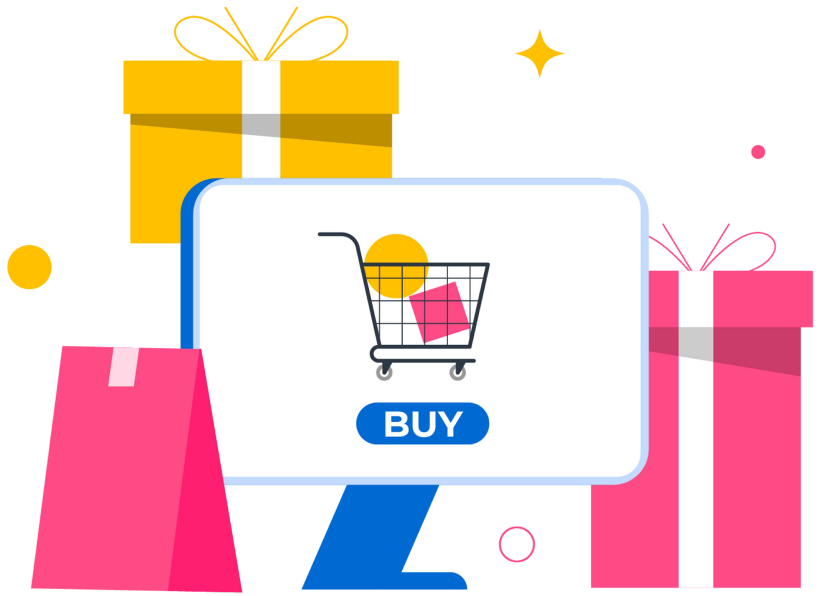
- Uma classe funciona como um “molde” para definição de outras estruturas. Classes, geralmente, são compostas pelo agrupamento de atributos ou características, e métodos ou ações. Uma classe define agrupamentos de atributos e métodos correlacionados e podem ser reaproveitados.



CONSTRUTOR

- Em Programação Orientada a Objetos (POO), um construtor é um método especial que é chamado automaticamente quando um objeto é criado a partir de uma classe. O objetivo principal de um construtor é inicializar as variáveis de instância da classe (ou seja, as características ou propriedades do objeto).

Classes em Dart

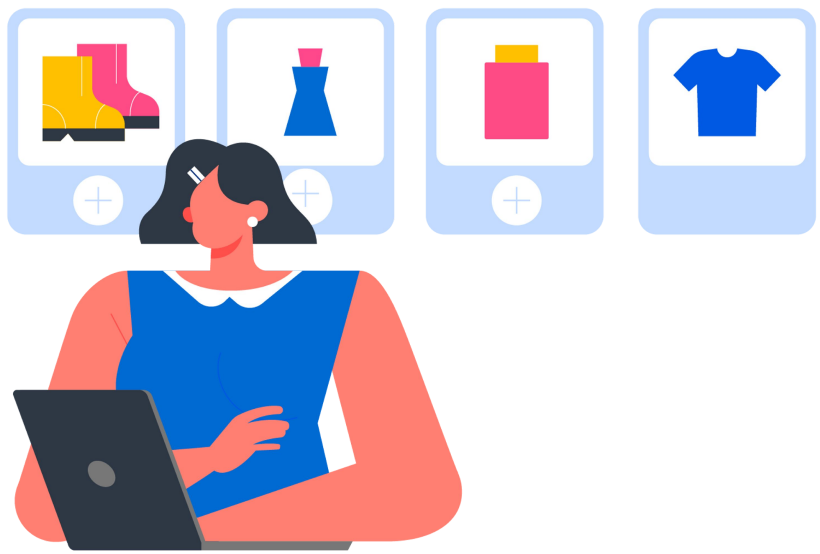



```dart
class Carro {
  String modelo;
  String marca;
  String chassi;
  String fabricante;

  Carro({
    required this.modelo,
    required this.marca,
    required this.chassi,
    required this.fabricante,
  });
```

OBJETOS

- Classes existem para definirmos os nossos moldes, ou seja, para definirmos o formato de estruturas que nosso código irá manipular. Mas, elas servem somente para ser moldes. Se quisermos as utilizar, precisamos colocar algo e criar uma estrutura que seja como esse molde. E é nesse momento que entram em cena os objetos.



```
Carro corsa = Carro(
    modelo: 'Corsa',
    marca: 'Chevrolet',
    chassi: 'ABC123',
    fabricante: 'Chevrolet',
);
```

MÉTODOS

- Os métodos são comportamentos e ações que os objetos podem ter.


```
void Ligar() {
    print("Carro ligado!");
}
```

HERANÇA

- Herança em Programação Orientada a Objetos é um mecanismo que permite a criação de novas classes a partir de classes já existentes, de forma que a nova classe possua todas as propriedades e métodos da classe original, além de poder adicionar suas próprias propriedades e métodos.
- Isso é feito por meio da criação de uma classe filha, que herda todas as características da classe mãe. A classe mãe é também conhecida como classe base ou superclasse, enquanto a classe filha é chamada de classe derivada ou subclasse.

HERANÇA

- A herança é útil porque evita a repetição de código, permitindo que classes similares compartilhem a mesma funcionalidade, enquanto ainda podem possuir suas próprias particularidades. Além disso, ela simplifica o processo de desenvolvimento, pois é possível criar novas classes a partir de classes já testadas e validadas.

```
class Animal {
  String nome;
  double peso;
  // pega as informações repassadas pelos animais gato e cachorro
  Animal(this.nome, this.peso);
  void comer(){
    print("$nome comeu");
  }
  void fazerSom(){
    print("$nome fez dom!");
  }
}
class Cachorro extends Animal {
  int fofura;
  // Temos que passar a função super, pois nome e peso não estão sendo passad
  Cachorro(String nome, double peso, this.fofura): super(nome, peso);
  void brincar(){
    fofura += 10;
    print("fofura do $nome aumentou para $fofura");
  }
}
class Gato extends Animal {
  Gato(String nome, double peso): super(nome, peso);
   estaAmigavel(){
    return true;
  }
}
```

```
void main(){
  Cachorro cachorro = Cachorro("Bob", 10.0, 100);
  cachorro.fazerSom();
  cachorro.comer();
  cachorro.brincar();
  Gato gato = Gato("Jerry", 10.0);
  cachorro.fazerSom();
  cachorro.comer();
  // como estou acessando um método do gato não se pode colocar somente cifrão,
  print("Está amigável? ${gato.estaAmigavel()}");
}
```

FONTES

- https://www.treinaweb.com.br/blog/orientacao-a-objetos-em-dart