

INSTITUTO FEDERAL DO PIAUÍ

TECNÓLOGO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS

Guilherme Daniel de sousa dias

- Interfaces
- Polimorfismo
- Sobrescrita
- Sobrecarga



INTERFACES



INTERFACES

- Interfaces especificam “**contratos**”, onde as classes que os implementam são obrigados a definir os métodos especificados nas interfaces.
- Normalmente as interfaces não possuem a implementação direta dos métodos, isso fica a cargo das classes concretas.



INTERFACES

- Dart não possui interfaces, ou melhor dizendo, não possui a palavra reservada **interface** para definir a criação de uma, o que é normal em várias linguagens de programação.
- As interfaces em Dart são **implícitas**, desse modo toda classe pode prover uma interface que pode ser usada por qualquer outra classe.
- Outra alternativa é usar a palavra **abstract** para torna a classe como abstrata de modo que não seja necessário implementar o corpo dos métodos das classes.
- É possível implementar várias interfaces, bastando separar por vírgulas.



```
abstract class Tributavel {  
    double calculaTributos();  
}
```




```
import 'ContaGenerica.dart';  
import 'Tributavel.dart';  
  
class ContaCorrente extends ContaGenerica implements Tributavel {  
    ContaCorrente(String correntista, double saldo) : super(correntista, saldo);  
  
    @override  
    double calculaTributos() {  
        return this.saldo * 0.10;  
    }  
}
```



Polimorfismo

Capacidade que uma Classe tem de herdar os atributo e métodos de uma outra classe podendo manter a mesma assinatura e modificar o comportamento.



```
abstract class ContaGenerica {
    String _correntista;
    double _saldo;
    bool pagaImposto;

    ContaGenerica(this._correntista, this._saldo, {this.pagaImposto = true});

    ContaGenerica.semImposto(this._correntista, this._saldo)
        : pagaImposto = false;

    double get saldo {
        return this._saldo;
    }

    String get correntista {
        return this._correntista;
    }

    void depositar(double valor) {
        this._saldo += valor;
    }

    double sacar(double valor) {
        return this._saldo -= valor;
    }
}
```



```
import 'ContaGenerica.dart';
import 'Tributavel.dart';

class ContaCorrente extends ContaGenerica implements Tributavel {
  ContaCorrente(String correntista, double saldo) : super(correntista, saldo);

  @override
  double calculaTributos() {
    return this.saldo * 0.10;
  }

  @override
  double sacar(double valor) {
    double taxa = valor * 0.05;
    return super.sacar(valor - taxa);
  }
}
```




```
import 'ContaGenerica.dart';
import 'Tributavel.dart';

class ContaSemImposto extends ContaGenerica implements Tributavel{
  ContaSemImposto.semImposto(String correntista, double saldo) :
    super.semImposto(correntista, saldo);

  @override
  double calculaTributos() {
    return 0.0;
  }
}
```



Sobrescrita

É um método da Orientação a Objetos que permite a uma classe que herde os métodos de um super classe a modificação do comportamento dos métodos herdados.

No Dart é necessário usar a assinatura `@override`. Desse modo, mesmo que um membro da superclasse seja renomeado, a subclasse pode funcionar normalmente.

O tipo de saída e o número de parâmetros devem ser os mesmos da super classe. Caso contrário um erro ocorrerá.



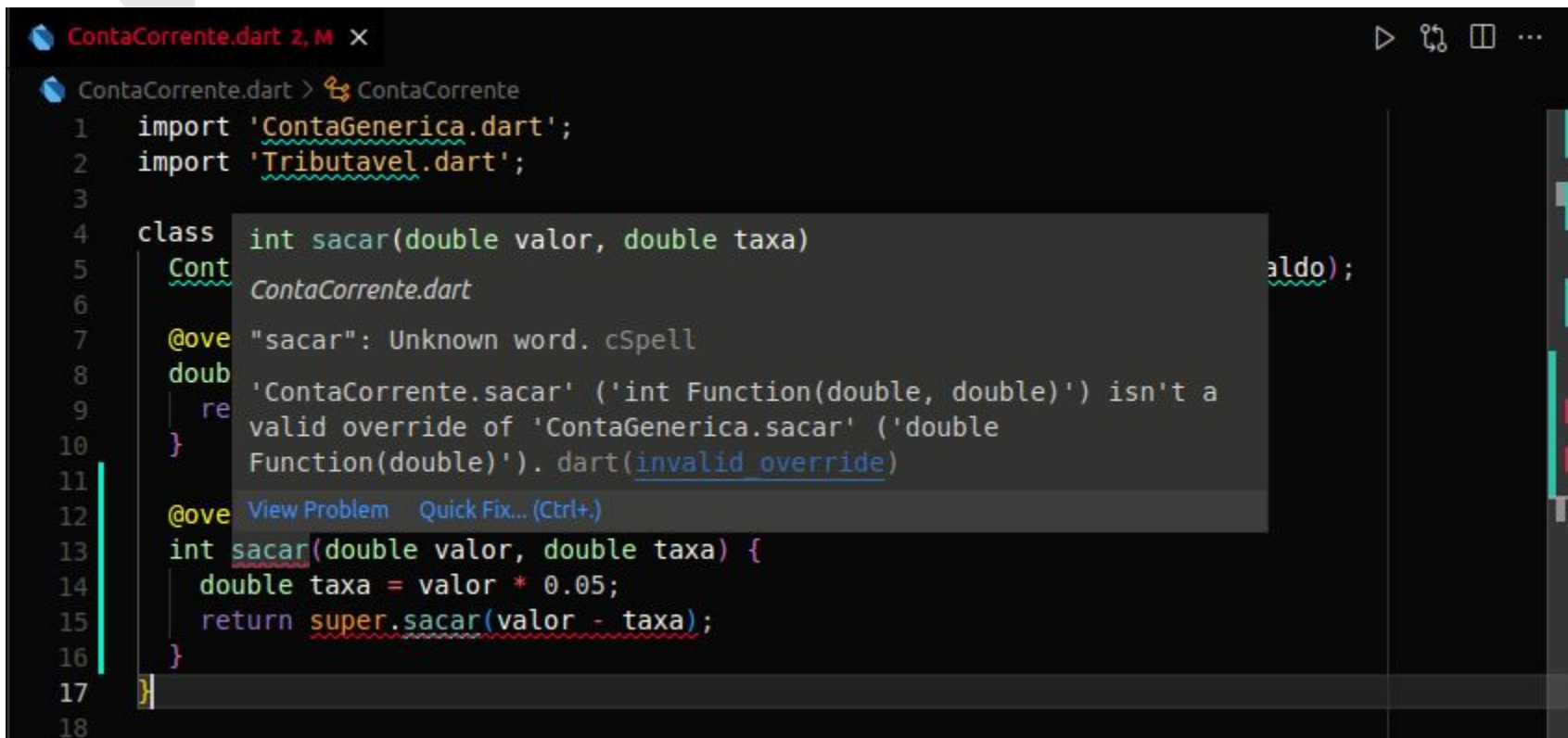
```
import 'ContaGenerica.dart';
import 'Tributavel.dart';

class ContaCorrente extends ContaGenerica implements Tributavel {
  ContaCorrente(String correntista, double saldo) : super(correntista, saldo);

  @override
  double calculaTributos() {
    return this.saldo * 0.10;
  }

  @override
  double sacar(double valor) {
    double taxa = valor * 0.05;
    return super.sacar(valor - taxa);
  }
}
```

Sobrescrita incorreta



The screenshot shows an IDE window titled 'ContaCorrente.dart 2, M x'. The code defines a class 'ContaCorrente' that inherits from 'ContaGenerica'. It imports 'ContaGenerica.dart' and 'Tributavel.dart'. The 'ContaCorrente' class has a method 'sacar' that calls 'super.sacar'. A tooltip is displayed over the 'sacar' method, showing an error: "'ContaCorrente.sacar' ('int Function(double, double)') isn't a valid override of 'ContaGenerica.sacar' ('double Function(double)'). dart(invalid override)". The tooltip also includes a 'View Problem' link and a 'Quick Fix...' option. The code is as follows:

```
1 import 'ContaGenerica.dart';
2 import 'Tributavel.dart';
3
4 class ContaCorrente {
5   int sacar(double valor, double taxa) {
6     double taxa = valor * 0.05;
7     return super.sacar(valor - taxa);
8   }
9 }
```



Sobrecarga

Dart não suporta sobrecarga de operadores do modo como é encontrado em outras linguagens, por exemplo mantendo o mesmo nome do Construtor e alterando o número de parâmetros.

Existem algumas formas de criar uma “sobrecarga artificial”:

- Utilizando nomes diferentes para métodos/construtores
- Parâmetros opcionais não-nomeados
- Parâmetros opcionais nomeados



```
abstract class ContaGenerica {
    String _correntista;
    double _saldo;
    bool pagaImposto;

    ContaGenerica(this._correntista, this._saldo, {this.pagaImposto = true});

    ContaGenerica.semImposto(this._correntista, this._saldo)
        : pagaImposto = false;

    double get saldo {
        return this._saldo;
    }

    String get correntista {
        return this._correntista;
    }

    void depositar(double valor) {
        this._saldo += valor;
    }

    double sacar(double valor) {
        return this._saldo -= valor;
    }
}
```



Referências

<https://dart.dev/guides/language/language-tour#implicit-interfaces>

<https://api.dart.dev/be/137507/dart-core/override-constant.html>

<https://dart.dev/guides/language/effective-dart/design#avoid-using-runtime-tests-to-fake-overloading>

<https://github.com/dart-lang/language/issues/1122>

<https://stackoverflow.com/questions/62626833/why-does-dart-not-allow-method-overloading>

<https://www.freecodecamp.org/news/constructors-in-dart/>