

# AULA 8

Refatorando projetos



An illustration of a woman with short dark hair, wearing a black t-shirt and green pants, holding a green Android robot. She is standing next to a large green rectangular screen. The background is light gray.

OI

# SUPER TRUNFO

Orientação à objetos

## SUPER TRUNFO

Super trunfo é um jogo de cartas. Veja o código do aplicativo Super trunfo para entender como o jogo funciona.

<https://github.com/marcellalcs/supertrunfo>



## O QUE É PROGRAMAÇÃO ORIENTADA À OBJETOS

Objeto: coisa material ou abstrata que pode ser percebida pelos sentidos e descrita por meio das suas **características**, **comportamento** e **estado** atual.

## QUAIS OBJETOS EXISTEM NO APP SUPER TRUNFO?

- JOGADOR
- VEÍCULO
- CONDUTOR
- CARTAS
- ...

É um modelo de código de programa extensível para criar objetos, fornecendo valores iniciais para o estado (variáveis de membro) e implementações de comportamento (funções ou métodos de membro)

## VEHICLE CLASS - KOTLIN

Palavra reservada class

```
class Vehicle ()
```

Não precisamos de mais nada pra definir uma classe

Nome começa com letra maiúscula

Para criar uma instância

```
val newVehicleOne = Vehicle()
```

## VEHICLE CLASS - KOTLIN

```
class Vehicle () {  
    var maxAcceleration: Int  
}
```

Para definir um atributo desta forma, precisamos inicializá-lo

```
class Vehicle () {  
    var maxAcceleration: Int = 0  
}
```



## VEHICLE CLASS - KOTLIN

```
val newVehicleOne = Vehicle()  
newVehicleOne.maxAcceleration = 100
```

Getters e setters são definidos automaticamente para qualquer propriedade

```
class Vehicle () {  
    var maxAcceleration: Int = 0  
    get() = field * 3  
    set(value) {  
        field = value/3  
    }  
}
```

É possível sobre escrever os getters e setters, adicionando comportamentos personalizados

## DESAFIO

```
val newVehicleOne = Vehicle()  
newVehicleOne.maxAcceleration = 100  
val maxAcceleration : Int = newVehicleOne.maxAcceleration
```

Qual o valor de maxAcceleration?

Podemos trabalhar com as propriedades dos objetos declarando-as como no exemplo anterior. Mas essa **não é a forma mais comum de declarar propriedades de classes em Kotlin.**

## CLASS CONSTRUCTORS - KOTLIN

```
class Vehicle (  
    val maxAcceleration: Int,  
    val accelerationTime: Int,  
    val passengers: Int,  
    val weight: Int,  
    val doors: Int,  
    val style: String,  
    val gears: Int,  
    val type: String)
```

## CLASS CONSTRUCTORS - KOTLIN

```
val newVehicleOne = Vehicle(  
    maxAcceleration: 100,  
    accelerationTime: 120,  
    passengers: 5,  
    weight: 120,  
    doors: 2,  
    style: "sedã",  
    gears: 5,  
    type: "car")
```

## CLASS CONSTRUCTORS - KOTLIN

```
val testAcceleration : Int = newVehicleOne.accelerationTime  
newVehicleOne.maxAcceleration = 10
```

Por que o erro?

## DATA CLASS - KOTLIN

```
dataclass.kt x
1  fun main(args: Array<String>) {
2      val vehicleOne = Vehicle( name: "Uno", accelerationTime: 10)
3      val vehicleTwo = Vehicle( name: "Uno", accelerationTime: 10)
4
5      if(vehicleOne == vehicleTwo){
6          print("mesmo carro")
7      } else{
8          print("carros diferentes")
9      }
10
11 }
12
13 class Vehicle (var name: String, var accelerationTime: Int)
```

O que será impresso no console?

A linguagem Kotlin introduz o conceito de classes de dados, que representam classes simples usadas como contêineres de dados e não encapsulam nenhuma lógica adicional.



## DESAFIO - CLASS DRIVER

Crie a class Driver

- **Public:** Se você não especificar nenhum modificador de visibilidade, public é usado por padrão, o que significa que suas declarações estarão visíveis em qualquer lugar;
- **Private:** Se você marcar uma declaração como privada, ela só ficará visível dentro do arquivo que contém a declaração;
- **Internal:** Se você marcá-lo interno, ele ficará visível em qualquer lugar no mesmo módulo;
- **Protected:** protegido não está disponível para declarações de nível superior.