



# Programação Orientada a Objetos

Orientação a Objetos e Classes

Prof. Marina de Lara



# 01

## Orientação a Objetos

Vamos entender um pouco do conceito deste paradigma de programação

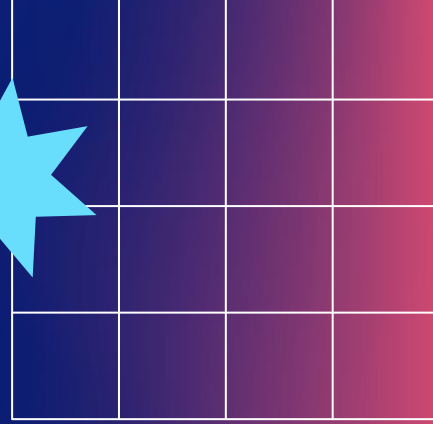
# A Orientação a Objetos



Quando começamos a programar aprendemos a fazer códigos que solucionam pequenos problemas e projetos. Mas e se caso você estiver implementando um grande sistema, como por exemplo um sistema de venda de veículos, como você imagina que ficaria o código com tantas funcionalidades?

# A Orientação a Objetos

A orientação a objetos têm como objetivo principal abstrair diferentes conceitos do mundo real para o código, de forma que conseguimos representar objetos, pessoas e funcionalidades dentro do código de uma maneira mais efetiva e organizada que facilita a implementação e manutenção de grandes sistemas.



# Como funciona?

Vamos fazer uma prática para entender melhor o conceito.

Quais características um carro possui?



# Características de um carro



- Marca
- Modelo
- Ano
- Cor
- Chassi

# Como funciona?

E quais comportamentos (ações) o carro pode executar?




# Comportamentos (ações) de um carro



- Ligar
- Desligar
- Acelerar
- Frear
- Alterar Marcha





# **Como ficaria a representação deste carro em um código?**

De que forma podemos representar as características  
e comportamentos de um carro no código?



# Representação no código

## Variáveis (Características)

String marca

String modelo

int ano

String cor

String chassi

## Funções (Comportamentos)

void ligar()

void desligar()

void acelerar()

void frear()

void trocarMarcha(int marcha)

# ✨ Novos Nomes ✨

## ~~Variáveis~~ **Atributos** **(Características)**

String marca

String modelo

int ano

String cor

String chassi

## ~~Funções~~ **Métodos** **(Comportamentos)**


void ligar()

void desligar()

void acelerar()


void frear()

void trocarMarcha(int marcha)



Pronto! Agora usamos variáveis (atributos) para representar as características do nosso carro e funções (métodos) para representar os comportamentos.

**Mas e agora? Colocamos  
tudo isso lá no main?  
E se precisarmos criar um  
caminhão ou uma moto,  
deixamos tudo junto?**





# 02

# Classes

A base de todo o paradigma da programação  
orientada a objetos são as Classes  
(Sua nova melhor amiga)

# O que é uma Classe?

Uma classe, no código, é a forma que vamos utilizar para representar um objeto dentro do nosso código. Podemos dizer que uma classe **é a representação de um objeto**. Na linguagem Java, praticamente em todos os casos, criamos uma classe em um novo arquivo *.java* no repositório *src* do nosso projeto.

## Carro

**String** marca  
**String** modelo  
**int** ano  
**String** cor  
**String** chassi

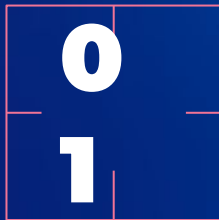
**void** ligar()  
**void** desligar()  
**void** acelerar()  
**void** frear()  
**void** trocarMarcha(**int** marcha)

```
public class Carro {  
    public String marca;  
    public String modelo;  
    public int ano;  
    public String cor;  
    public String chassi;  
  
    public void ligar() {  
        //Implementação Aqui  
    }  
    public void desligar() {  
        //Implementação Aqui  
    }  
    public void acelerar {  
        //Implementação Aqui  
    }  
    public void frear() {  
        //Implementação Aqui  
    }  
    public void trocarMarcha(int marcha) {  
        //Implementação Aqui  
    }  
}
```

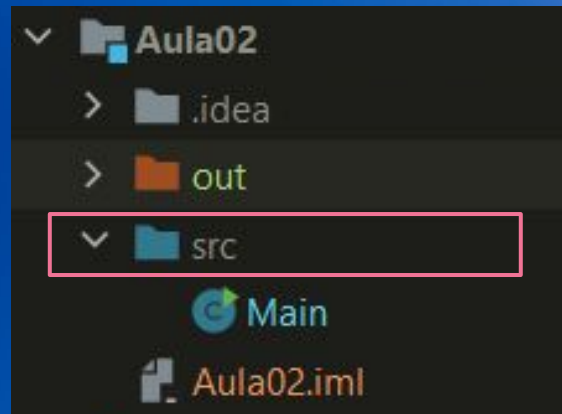
# O Código

O nosso objeto Carro no Java seria implementado como no exemplo ao lado. Criamos uma classe com o nome **Carro** que contém os atributos e métodos de acordo com o que planejamos anteriormente.

# Criando uma classe: IntelliJ



Encontre o repositório (pasta) **src** dentro do projeto e clique com o botão direito do mouse no repositório para abrir o menu de opções.

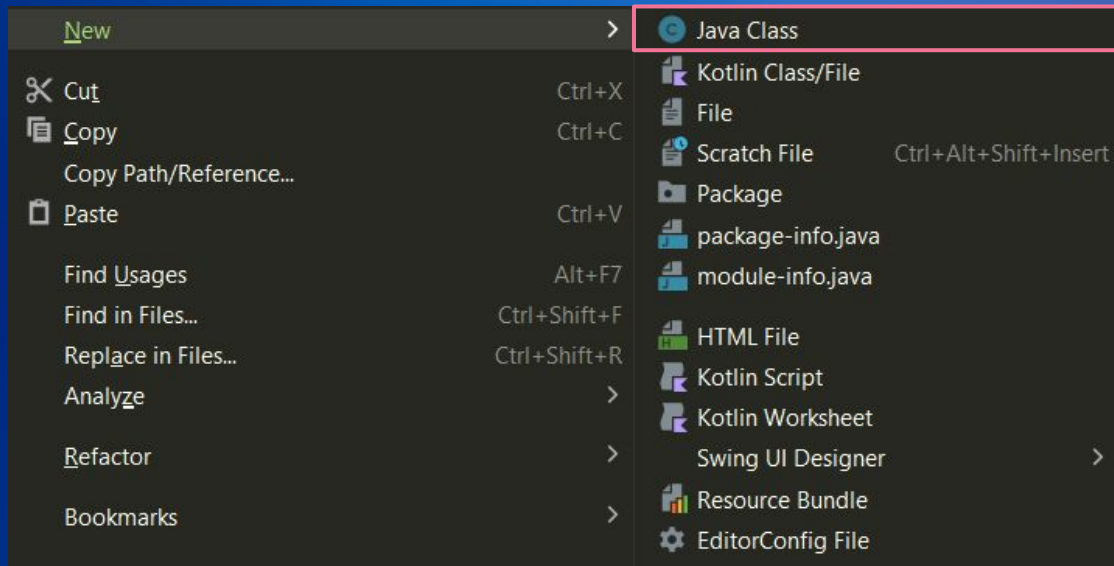




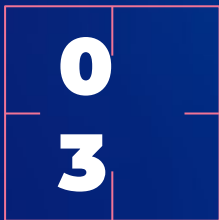
# Criando uma classe: IntelliJ

0  
2

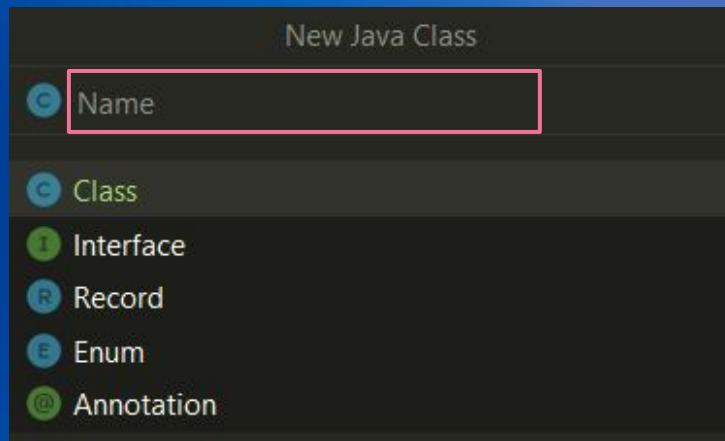
Navegue pela opção  
New e então clique na  
primeira opção:  
**Java Class**



# Criando uma classe: IntelliJ



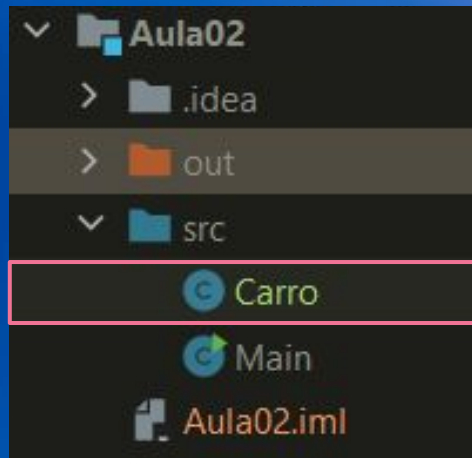
Na janela que aparece no meio do editor, substitua o placeholder *Name* pelo nome da sua classe. No caso do nosso exemplo: Carro



# Criando uma classe: IntelliJ



Agora que sua classe Carro já está criada no lugar certo, você pode voltar ao slide 15 e implementar o código como no exemplo.



# Lembretes e Boas Práticas!

- ★ Quando for implementar uma classe nunca pense em um objeto específico, sempre tente abstrair os conceitos daquele tipo de objeto de uma forma geral. Por exemplo: na classe Carro, não pense em um Hyundai HB20, ano 2021, cor prata... pense sempre nas características e comportamentos que se aplicam a **qualquer carro**.
- ★ Uma classe é a representação de **um único objeto**. Quando criamos uma classe **SEMPRE** teremos o nome no singular, pois aquela classe representa **UM** objeto.
- ★ Classes devem **SEMPRE começar com letra maiúscula, SEMPRE!**
- ★ Procure dar nomes claros para seus atributos e métodos, de forma que fique claro o que eles representam dentro do objeto. Lembre-se, quanto mais legível e compreensível for o seu código, melhor!
- ★ Métodos representam o comportamento de um objeto, então devem sempre ter nomes que indicam ação, por exemplo: correr, jogar, acelerar, descontar, etc.

# Bons Estudos!

## Dúvidas?

lara.marina@pucpr.br

Mensagem no Canvas

Servidor no Discord

CRÉDITOS: Este modelo foi criado pelo **Slidesgo**, e includes ícones da **Flaticon**, infográficos e imagens do **Freepik** e conteúdo de **Swetha Tandri**