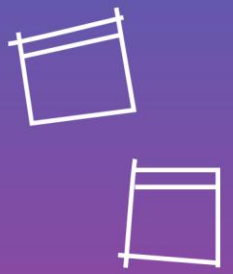
DBC
DIGITAL BUSINESS COMPANY®

VEM SER
DBC



Orientação a Objetos – Aula 1

Sumário

- Paradigma Procedural x OO
- Classes e Objetos
- Atributos e Métodos
- Wrappers
- Variáveis por Valor x Referência
- Pacotes

Paradigma **Procedural** x **OO**

- Na programação estruturada, um programa é tipicamente escrito em uma única rotina.

https://www.alura.com.br/artigos/poo-programacao-orientada-a-objetos

Paradigma **Procedural** x **OO**

- Na programação estruturada, um programa é tipicamente escrito em uma única rotina
- O intuito do OO e sua criação também foi o de aproximar o manuseio das estruturas de um programa ao manuseio das coisas do mundo real.

https://www.alura.com.br/artigos/poo-programacao-orientada-a-objetos

Paradigma **Procedural** x **OO**

- Na programação estruturada, um programa é tipicamente escrito em uma única rotina
- O intuito do OO e sua criação também foi o de aproximar o manuseio das estruturas de um programa ao manuseio das coisas do mundo real.
- Daí o nome "objeto" como uma algo genérico, que pode representar qualquer coisa tangível.

https://www.alura.com.br/artigos/poo-programacao-orientada-a-objetos

Paradigma **Procedural** x **OO**

- Na programação estruturada, um programa é tipicamente escrito em uma única rotina
- O intuito do OO e sua criação também foi o de aproximar o manuseio das estruturas de um programa ao manuseio das coisas do mundo real.
- Daí o nome "objeto" como uma algo genérico, que pode representar qualquer coisa tangível.
- Esse novo paradigma se baseia principalmente em dois conceitos chave: **classes** e **objetos**.

https://www.alura.com.br/artigos/poo-programacao-orientada-a-objetos

Classes e Objetos

- Classes são “moldes”, esses moldes podem dar origem à vários objetos.

https://www.alura.com.br/artigos/poo-programacao-orientada-a-objetos

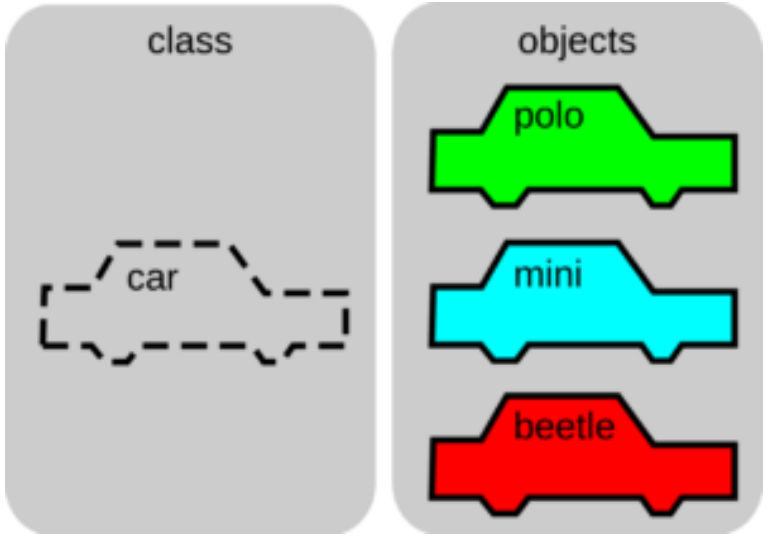
Classes e Objetos

- Classes são “moldes”, esses moldes podem dar origem à vários objetos.
- Objetos são instâncias das classes “moldes”.

https://www.alura.com.br/artigos/poo-programacao-orientada-a-objetos

Classes e Objetos

- Classes são “moldes”, esses moldes podem dar origem à vários objetos.
- Objetos são instâncias das classes “moldes”.



https://www.alura.com.br/artigos/poo-programacao-orientada-a-objetos

Classes e Objetos

```
class Carro {
}
```

```
Carro pegeout206 = new Carro();
Carro opala = new Carro();
Carro fiesta = new Carro();
```

Classes e Objetos

DECLARAÇÃO DE CLASSE

```
class Carro {
}
```

DECLARAÇÃO DE OBJETOS

```
Carro pegeout206 = new Carro();
Carro opala = new Carro();
Carro fiesta = new Carro();
```

Atributos e Métodos

- Atributos são características da nossa classe

Atributos e Métodos

- Atributos são características da nossa classe
- Métodos são comportamentos da nossa classe

Atributos

```
class Carro {
    String modelo;
    double quilometragem;
    int ano;
    String dono;
}
```

Atributos

```
class Carro {
    String modelo;
    double quilometragem;
    int ano;
    String dono;
}
```

```
Carro fiesta = new Carro();
fiesta.modelo = "fiesta";
fiesta.ano = 2019;
fiesta.quilometragem = 15222.0D;
fiesta.dono = "Pedro Henrique Cardoso";
```

Métodos

```
class Carro {
    public void acelerar() {
        /* código do carro para acelerar */
    }

    public void frear() {
        /* código do carro para frear */
    }

    public void acenderFarol() {
        /* código do carro para acender o farol */
    }
}
```

Métodos

```
class Carro {
    public void acelerar() {
        /* código do carro para acelerar */
    }

    public void frear() {
        /* código do carro para frear */
    }

    public void acenderFarol() {
        /* código do carro para acender o farol */
    }
}
```

```
Carro fiesta = new Carro();
fiesta.acenderFarol();
```

Métodos

```
public void acelerar() {
        /* código do carro para acelerar */
}
```

```
public int acelerar() {
        /* código do carro para acelerar */
       return 1;
}
```

```
public String acelerar() {
        /* código do carro para acelerar */
        return “Acelerando”;
}
```

Let’s practice;

Kahoot

Exercício #1

- Crie uma classe Pessoa
  - Defina atributos: nome, sobrenome, idade e whatsapp
  - Defina comportamentos:
    - conversar(pessoa): void
      - Deve imprimir a mensagem "**PessoaX** conversou com **PessoaY**"
    - retornarNomeCompleto(): String
      - Deve retornar o nome + sobrenome
    - ehMaiorDeIdade(): boolean
      - Deve retornar se a idade da pessoa é maior do que 18 (true) se não retorna (false)
    - mandarWhatsApp(pessoa, String mensagem): void
      - Deve imprimir a mensagem "**PessoaX.nome** enviou: **mensagem** para **PessoaY.nome**"
  - Criar uma classe main, crie duas pessoas e teste cada um comportamentos definidos.

Wrappers

- São os tipos primitivos em classes
  - int = **Integer**
  - double = **Double**
  - float = **Float**
  - char = **Character**
  - long = **Long**
  - boolean = **Boolean**
- Wrappers permitem Valores **nulos**

https://pt.strephonsays.com/wrapper-class-and-vs-primitive-type-in-java-1235

Converter Valores

- https://docs.oracle.com/javase/specs/jls/se17/html/jls-5.html
- https://www.devmedia.com.br/conversoes-em-java/2695

Let’s practice;

Exercício #2

- Temos um vetor de String com 3 posições com os seguinte valores:

  String[] valores = {“01234”, “5680.8”, “670.2”};

  onde os índices:

  0. Código do funcionário
  1. Salário
  2. Descontos

  Faça:

  Converta o Código do funcionário para Integer e exiba no console

  Converta o Salário e Descontos para Double

  Calcule o (Salário – Descontos) e exiba no console o valor.

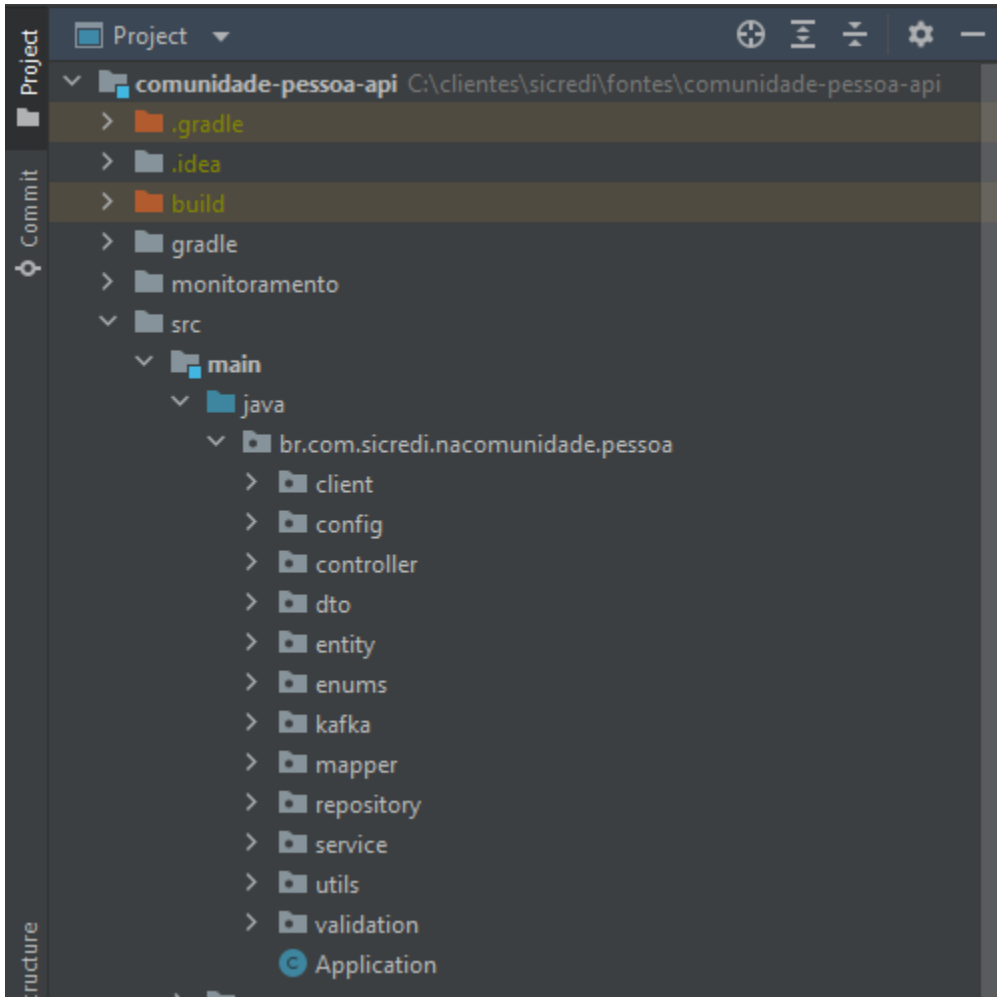
Variáveis por **Referência** e **Por Valor**

- Variáveis de tipos primitivos e Wrappers são **imutáveis**, portanto sempre são passados por valor (ou seja, nunca mudam ao executar o método).
- Objetos são sempre passados por referência, ou seja, podem ser mudados dentro do método (muito cuidado com isso).

Let’s practice;

Pacotes

- Servem para organizar as suas classes em pastas.
- Mais para frente veremos alguns padrões.



Homework #1

- Crie as classes conforme diagrama de classes:
  - https://lucid.app/lucidchart/5808a67c-d697-4ffb-8332-1acca28771c2/view
- Criar uma classe Main para testar **todas as operações** de ContaCorrente:
  - Esse teste deve ter ao menos 2 clientes com uma conta corrente cada um
  - 1 transferência entre eles
  - Ao final imprimir as duas contas
- Regras:
  - Não é permitido sacar mais do que o saldo + cheque especial
  - Não é permitido depositar, transferir e sacar valores negativos

Homework #2 Em Grupo

- Formar grupos de 3 integrantes
- Defina um nome para a equipe, exemplo: Batatinha Frita 123, CodeLords, Turma da pesada, turminha do lol, lords/ladys of code, xícara do java... sejam criativos.
- Crie um repositório “vemser-trabalho-final” no github para a equipe e coloque todos os membros como colaboradores.
- Escolher um tema para desenvolverem um sistema durante o programa
  - Não pode ser temas repetidos de outras equipes
- ATENÇÃO: Para validar o tema, converse com um dos instrutores!
- Podem haver temas parecidos mas não iguais, exemplo:
  - E-commerce para venda de celulares
  - E-commerce para venda de roupas
- Colocar os dados do time em https://docs.google.com/spreadsheets/d/18sJsOdzgEGV6UhmZ2GX6jsx9O5fmISjq_YQ6ViDsk_A/edit?usp=sharing

Homework #2 Em Grupo

- Após definidas as questões do slide anterior:
  - criar 4 classes (que tenham atributos e método úteis e que não seja a main)
  - criar um primeiro diagrama de classes com as classes definidas (somente métodos públicos e atributos públicos nesse primeiro momento)
  - Não se preocupem, será um esboço do trabalho final
  - Não precisa entregar no classroom =)

Sugestões de Temas

- Uma rede social (Instagram, Facebook ou Linkedin)
- Sistema para aluguel de quartos e casas (AIRBNB, Trivago)
- Loja Virtual (Magalu, Mercado Livre)
- Tocador de música (Spotify, Deezer)
- Jogos (Pokemon Pokedex, Digimon, RPG, Dungeons and Dragons)
- Sistema para empregos (Indeed, Infojobs)
- Sistema para encontros (Tynder, Happn)
- Sistema de transporte de carros (Uber, 99, InDrive)