VEM SER

Módulo 01 - Java + OO

Aula 03 - Paradigmas e OO

Conteúdo da aula

- Paradigmas de linguagens de programação;
- Programação Orientada a Objetos;
- Classes e objetos;
- Atributos e métodos;
- *Wrappers*;
- Pacotes;
- UML e diagramas.

Paradigmas de linguagem de programação

Paradigma Procedural/Imperativo

- Nesse paradigma, o foco está em **descrever os passos** a serem executados para alcançar um determinado resultado.
- O programa é estruturado em **sequência de instruções** e utiliza variáveis para armazenar e manipular dados.

Exemplos de linguagem que usam: C, Fortran e Pascal.

Paradigma Funcional

- No paradigma funcional, o foco está em **funções**.
- Os problemas são resolvidos quebrando-os em funções menores e combinando essas funções para obter o resultado desejado.
- As funções podem ser compostas, passadas como argumentos para outras funções e retornadas como resultados.

Exemplos de linguagem que usam: Haskell, Scala (também suporta outros paradigmas), Clojure, JavaScript, Python (também suporta outros paradigmas)

Paradigma de Programação Orientada a Objetos (POO)

Paradigma Orientado a Objetos

- Nesse paradigma, a ênfase está na modelagem de objetos que possuem propriedades (atributos) e comportamentos (métodos).
- Os objetos interagem entre si por meio de mensagens, permitindo a reutilização de código, encapsulamento e abstração.

Exemplos de linguagem que usam: Java, C++, C#, PHP, Python (também suporta outros paradigmas), Ruby.

Programação Orientada a Objetos

O intuito da orientação a objetos foi de aproximar as estruturas de um programa com as coisas no mundo real;
O nome **objeto** sendo algo genérico, pode representar qualquer coisa tangível;
Esse paradigma se baseia, principalmente, em **classes** e **objetos**.

Classes e Objetos

Classes e Objetos

= Objeto

= Classe

Crédito da imagem: https://www.devmedia.com.br/introducao-a-programacao-orientada-a-objetos-em-java/26452

Classes e Objetos

- Classes são como “moldes” ou “modelos”;
- Objetos são instâncias das classes “modelos”.

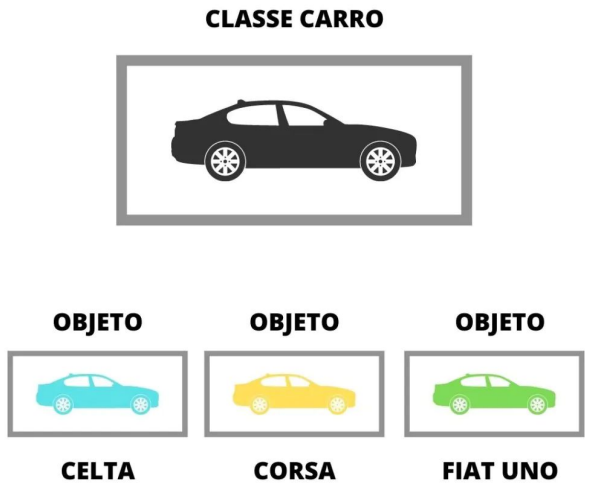
O que é instância?

- Quando criamos um objeto a partir de uma classe, estamos **instanciando uma classe**;
- É criado um objeto real que é uma representação específica dessa classe;
- Cada objeto criado a partir da classe é uma instância única com seu próprio conjunto de valores para as propriedades da classe;
- Cada instância (objeto) criada a partir de uma classe é independente das outras instâncias;
- Eles têm seu próprio estado e podem executar seus próprios métodos de forma separada;
- No entanto, eles compartilham a mesma estrutura e comportamentos definidos pela classe.

Classes e Objetos

```
class Carro {
}
```

```
Carro celta = new Carro();
Carro corsa = new Carro();
Carro fiatUno = new Carro();
```



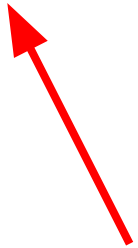
Crédito da imagem: https://programadoresdepre.com.br/o-que-e-programacao-orientada-a-objetos-poo/

Classes e Objetos

```
class Carro {
}
```

Declaração da classe

```
Carro celta = new Carro();
Carro corsa = new Carro();
Carro fiatUno = new Carro();
```



Declaração dos objetos

Atributos e métodos

Atributos e Métodos

- Atributos são **características** da classe;
- Métodos são **comportamentos** da classe.

Em termos mais técnicos

- **Classe:** representação de um item do mundo real, na forma de um tipo de dados personalizado;
- **Atributo:** dados dos objetos de uma classe;
- **Métodos:** operações que a instância da classe pode realizar.

Atributos

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

```
Carro celta = new Carro();
celta.modelo = “Celta”;
celta.ano = 2000;
celta.quilometragem = 1240899;
celta.dono = “Pedro”;
```

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

Retornos dos 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";
}
```

Vamos praticar!

Crie uma classe Pessoa

- Defina atributos: nome, sobrenome e idade
- Defina os métodos:
  - **conversar**(pessoa): void
    - Deve imprimir a mensagem "**Nome da pessoa X** conversou com **Nome da pessoa Y**"
  - **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)
  - **mandarMensagem**(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.

DBC

Wrappers

Wrappers

São os tipos primitivos em classes:

- int = **Integer**
- double = **Double**
- float = **Float**
- char = **Character**
- long = **Long**
- boolean = **Boolean**

Wrappers permitem valores **nulos**

Valor vs Referência

- 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).

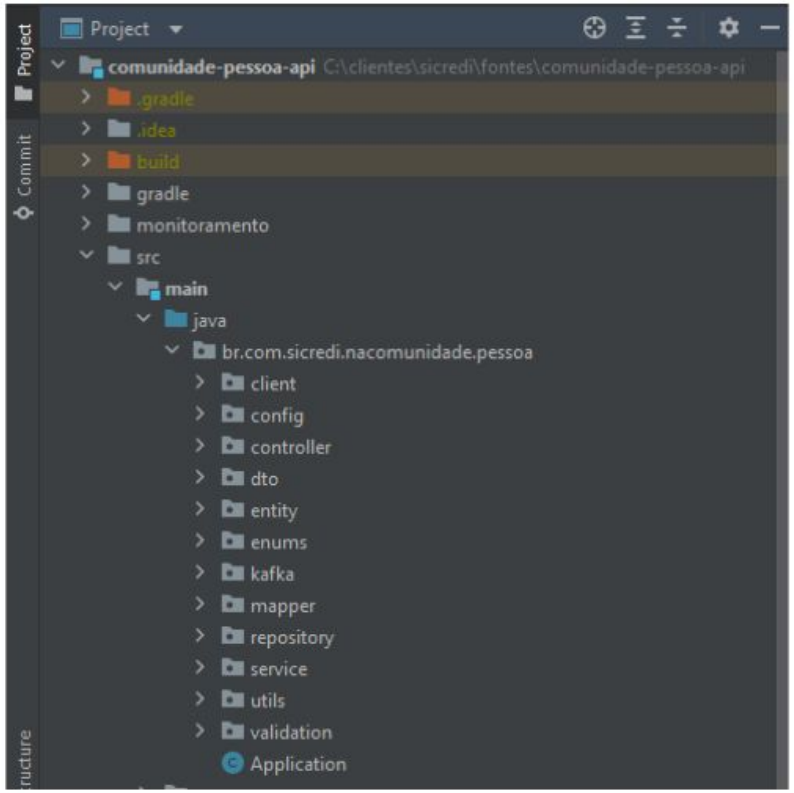
Valor vs Referência

Crédito da imagem: https://blog.penjee.com/passing-by-value-vs-by-reference-java-graphical/

Pacotes

Pacotes

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



UML e diagramas

UML (*Unified Modeling Language*)

A Linguagem Unificada de Modelagem é uma linguagem com foco na visualização de arquiteturas de *software*, criada por volta de 1995, sendo muito utilizada no mercado até os dias de hoje.

Diagrama

"Representação gráfica esboçada de alguma coisa; DELINEAÇÃO; ESBOÇO*: diagrama de conexão dos cabos.*" (Dicionário Aulete)

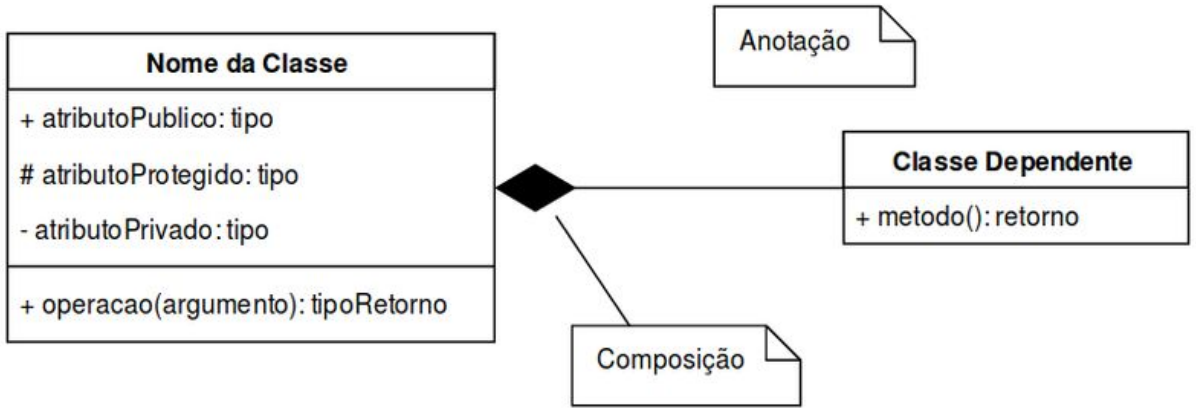
https://aulete.com.br/diagrama

Alguns tipos de diagrama no UML

- **Diagrama de classes;**
- **Diagrama de casos de uso;**
- Diagrama de objetos;
- Diagrama de componentes;
- Diagrama de pacotes;
- etc...

Diagrama de classes

- Em diagramas de classes, as classes são representadas por uma forma retangular dividida em três partes:
  - A parte superior exibe o nome da classe,
  - A do meio seus atributos;
  - A inferior suas operações ou métodos.

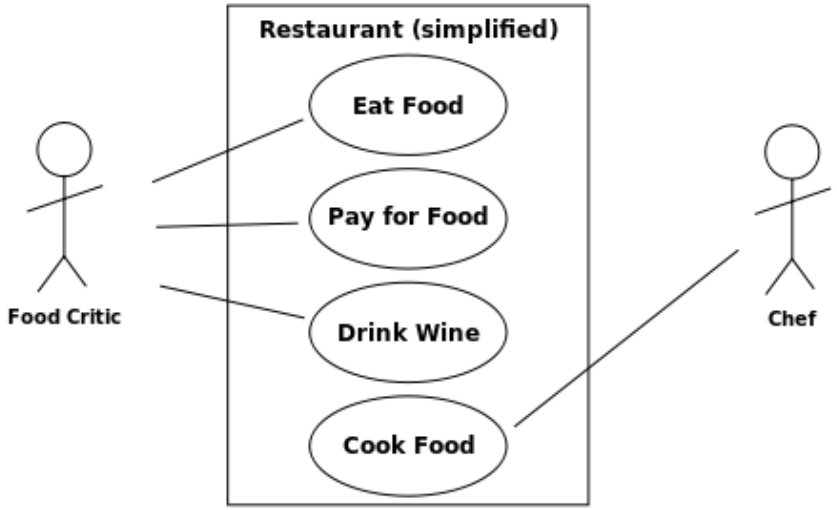


Crédito da imagem: https://pt.wikipedia.org/wiki/Diagrama_de_classes

Diagrama de casos de uso

O diagrama de caso de uso resume os detalhes dos **usuários** do seu sistema (também conhecidos como atores) e as interações deles com o sistema. (Fonte: Lucidchart)

Diagrama de casos de uso



Crédito da imagem: https://pt.wikipedia.org/wiki/Diagrama_de_caso_de_uso

Recomendação de vídeo

https://www.youtube.com/watch?v=JQSsqMCVi1k

Recomendações de leitura

https://creately.com/blog/pt/diagrama/guia-de-tipos-de-diagramas-uml-aprenda-sobre-todos-os-tipos-de-diagramas-uml-com-exemplos/

https://medium.com/documenta%C3%A7ao-uml/diagrama-de-classes-ba91a9d29575

https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-classe-uml

https://www.lucidchart.com/pages/pt/diagrama-de-caso-de-uso-uml

Task #1 individual obrigatória

- **Para amanhã às 11h da manhã:**
- Criar um projeto "**conta-corrente1**" dentro da pasta "aula-03" do módulo 1
- Crie as classes conforme diagrama de classes:

https://lucid.app/lucidchart/c5aaf45a-5a18-43e2-a8bd-43ab5aafe994/edit?viewport_loc=-219%2C-20%2C2354%2C1184%2C0_0&invitationId=inv_24935292-7f97-4846-8a8c-502e88423a98

- 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
    - NÃO MANIPULAR A VARIÁVEL CHEQUE ESPECIAL

Task #2 em grupo

- Definam um nome para a equipe, sejam criativos;
- Crie um repositório no github para a equipe e coloque todos os membros como colaboradores;
- **Colocar o link do repositório do time em** https://docs.google.com/spreadsheets/d/1bcNNXJNTOmAg9UGKroH_M5gf8nnyf61-BM9ZonvOysM/edit#gid=0 **.**

Task #2 em grupo

- Após definidas as questões do slide anterior:
  - Criar o código das **4 classes** (que tenham **atributos** - obrigatório e métodos úteis - opcional)
  - Criar um primeiro diagrama de classes com as classes definidas (somente métodos **públicos** e atributos **públicos** neste primeiro momento)
  - Não se preocupem, será um **esboço** do trabalho final
- **Entrega obrigatória dia 08/01 - final do módulo de Java**

Links úteis

Documentação do Java: https://docs.oracle.com/javase/specs/jls/se17/html/index.html


Let's Tech Up Together