

afrodev

# programação orientada a objetos

material criado e ministrado por Sandyara Peres



# paradigma de programação

Um paradigma é um estilo de programação, um modelo, uma metodologia. Não se trata de uma linguagem, mas a forma como você soluciona problemas usando uma determinada linguagem de programação. Hoje no mercado existem os paradigma do tipo:

## **imperativo**

- Programação procedural;
- Computação paralela
- Programação orientada a objetos.

## **declarativo**

- Programação lógica;
- Programação funcional.

# orientação a objetos

A programação orientada a objetos surgiu como uma alternativa a essas características da programação estruturada. O intuito da 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. Todos os outros conceitos, igualmente importantes, são construídos em cima desses dois.

# classes

Classe em orientação a objetos é definida tanto como uma representação de uma entidade do mundo real como uma representação de uma entidade que faz parte do domínio da solução.

# objetos

Objetos são denominados instâncias de classes e possuem vida independente entre si, apesar de compartilharem o mesmo “molde” (Classe).

# 4

## relação entre classe x objeto

Classes servem de molde ou especificação para objetos.

Assim, os objetos do tipo de uma determinada classe são obrigados a implementar a especificação (molde) desta, ou seja, devem apresentar as mesmas características: informações (atributos) e comportamentos (operações).



5

# abstraindo...



Vamos abstrair uma pessoa do mundo real e fazer uma classe a partir dela.

A moça ao lado é cliente em um banco.  
Responda:

1. Quais são as características de uma pessoa física que um banco precisa e pode informar?
2. Quais são os comportamentos comuns que uma pessoa física executa em um banco?

5

# abstraindo...



## **características (atributos)**

- Nome completo;
- CPF;
- Saldo;

## **comportamentos (métodos)**

- Realizar saque

5

# abstraindo...

<b>Pessoa</b>
<ul style="list-style-type: none"><li>- nome : String</li><li>- cpf : String</li><li>- saldo : Double</li></ul>
<ul style="list-style-type: none"><li>+ realizaSaque() : void</li></ul>



Nome da classe



Atributos da classe  
e seu tipo

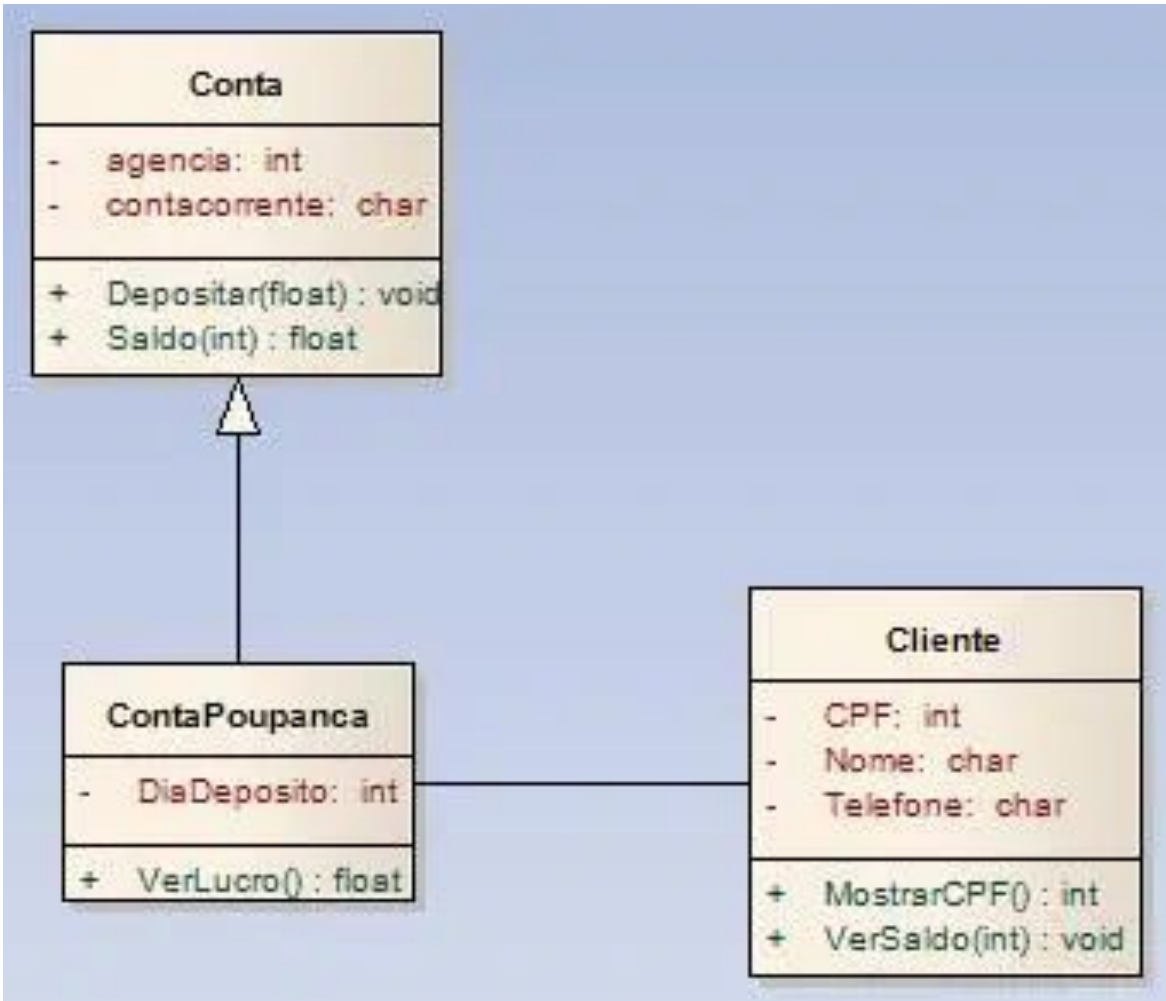


Métodos da classe e  
seu retorno



## 6

# diagrama de classes



Serve para ilustrar as classes a serem utilizadas em um sistema. Ele é estático pois especifica quais classes irão se interagir, mas não especifica o que acontece e quando ocorre essa a interação.

Veja esse link sobre a construção de diagramas de classe:

<http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/uml/diagramas/classes/classes1.htm>

# encapsulamento

Define a visibilidade de um atributo ou método de uma classe em relação a demais de um sistema. Esses modificadores podem ser:

## **public**

Acesso às características com esse modificador é totalmente livre.

## **private**

Acesso às características com esse modificador é permitido apenas de dentro da própria classe que as define.

## **protected**

Acesso às características com esse modificador é restrito a própria classe ou suas filhas.

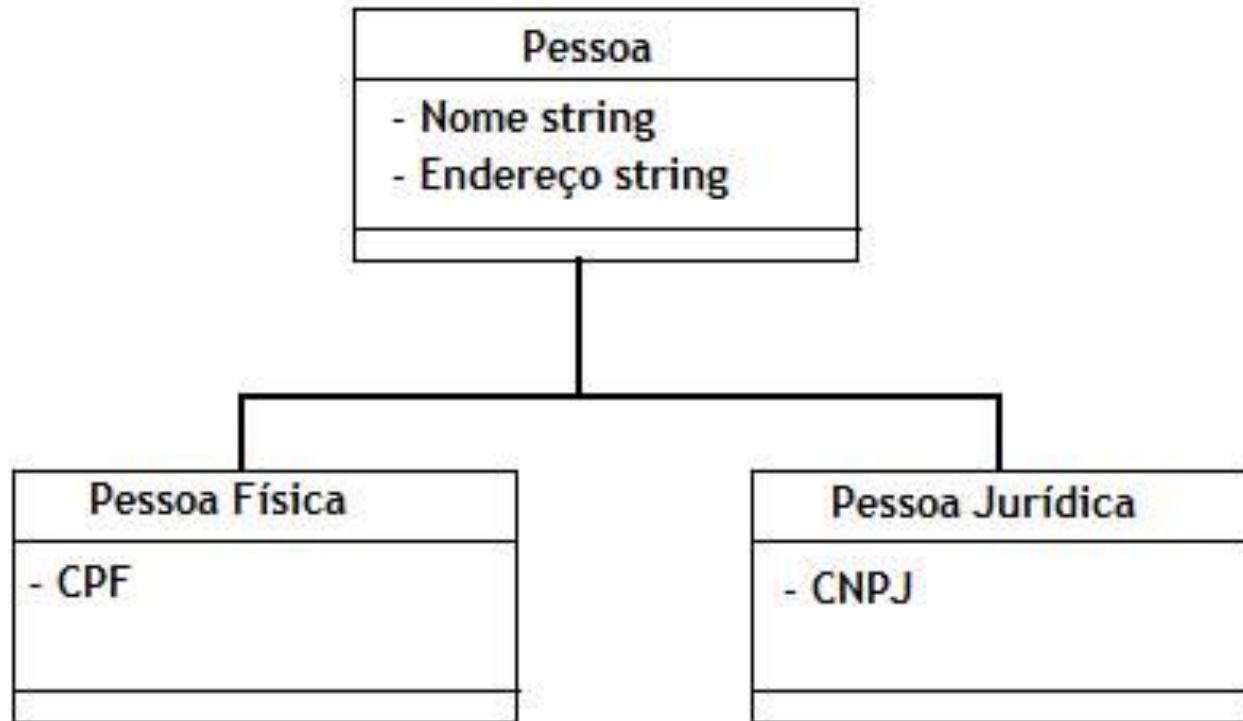
# herança

Generalização, especialização e herança são abstrações poderosas para compartilhar similaridades entre classes enquanto se preserva as especificidades das mesmas. São formas de relacionar classes por meio de hierarquias.

Generalização é o relacionamento entre uma determinada classe e uma ou mais versões refinadas (especializadas) desta.

A classe sendo especializada é denominada superclasse (ou classe base, classe mãe) e as classes refinadas ou especializadas são denominadas subclasses (ou classes descendentes, filhas).

# herança



Em um contexto bancário, essas duas classes são especialistas da classe Pessoa.

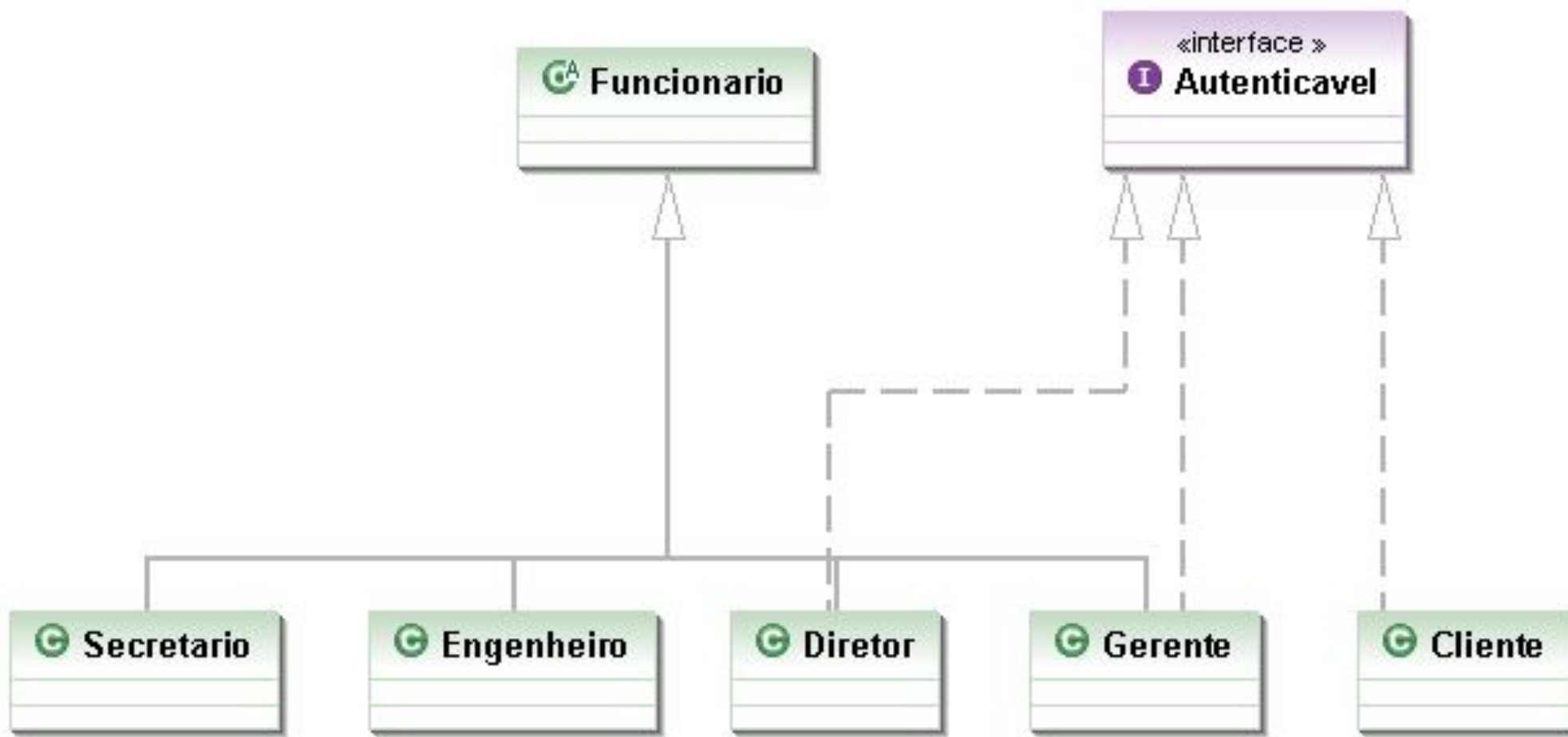
Assim, pessoa física e Pessoa Jurídica herdam os mesmos atributos e métodos da classe Pessoa como nome e endereço.

# interfaces

Não podemos realizar herança múltipla, para isso utilizamos interfaces.

A interface é um recurso muito utilizado em Java, bem como na maioria das linguagens orientadas a objeto, para “obrigar” a um determinado grupo de classes a ter métodos ou propriedades em comum para existir em um determinado contexto, contudo os métodos podem ser implementados em cada classe de uma maneira diferente.

# interfaces



# atividade proposta 1

Para esse exercício, foque apenas em esboçar o diagrama de classes, tentando extrair as classes, seus atributos e métodos bem como seus relacionamentos (como herança).

Sinta-se livre para desenvolver o projeto em um segundo momento. ;)

# atividade proposta 1

A lanchonete "Pobr's" está em fase de crescimento em seu delivery com a pandemia e quer seu funcionamento. Porém, os donos sempre operaram utilizando planilhas do Excel e ferramentas bastante simples, sendo assim, nenhum sistema personalizado foi criado e você pretende automatizar o trabalho desse estabelecimento.

Na entrevista com os donos do estabelecimento você começou a questioná-los sobre o funcionamento do estabelecimento. Alguns detalhes foram surgindo durante a conversa:



# atividade proposta 1

1. A lanchonete possui 3 principais itens de venda: pizzas, lanches e salgadinhos. Inicialmente, o sistema será testado para controlar as vendas desses 3 itens apenas.
2. Todos os itens vendidos devem conter: preço de venda, data de validade e peso.
3. O sistema da nossa lanchonete deverá criar um pedido, esse pedido será composto pelo nome do cliente, itens que foram consumidos e taxa de serviço.
4. O sistema deve permitir gerar a nota fiscal para entregar ao cliente. O vendedor poderá inserir o valor recebido em dinheiro e o sistema calcula e mostra o troco do cliente na tela.

## atividade proposta 2

Crie um app que “simule” a compra de apenas 1 ingresso para o show de sua banda favorita, no app, exibe o valor base do ingresso via TextView.

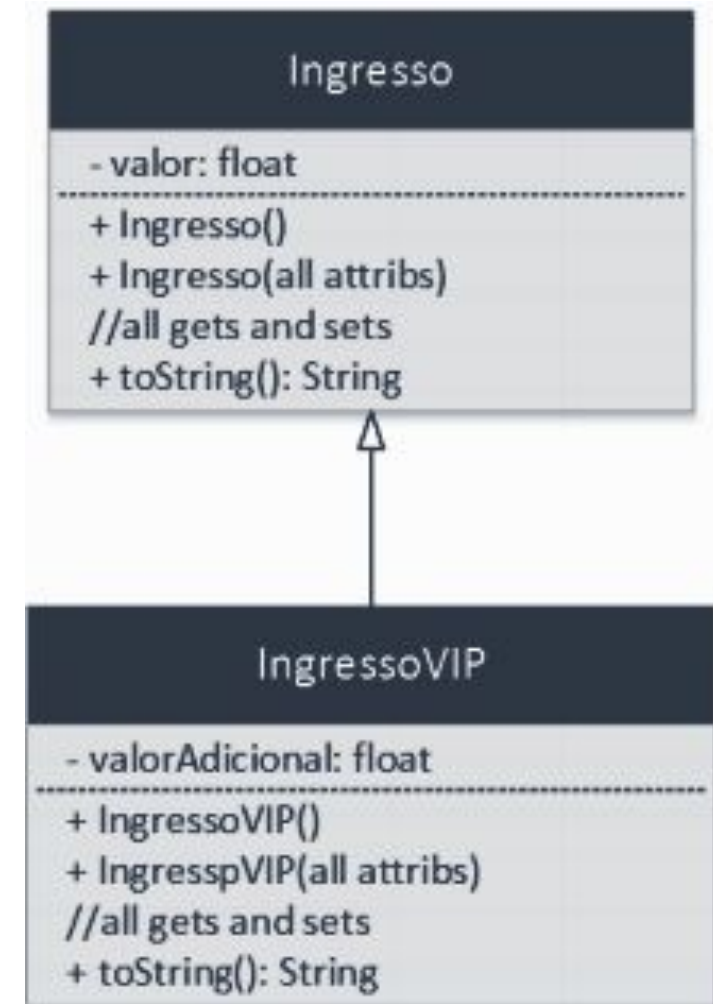
Através de um rádio button, o usuário pode escolher se ele quer um ingresso pista ou VIP, caso seja VIP haverá um acréscimo (você, desenvolvedor, escolherá).

Ao clicar em “Comprar”, é exibido um Toast de “Compra feita com sucesso”.

10

# atividade proposta 2

Crie uma classe IngressoVIP, que herda de Ingresso e possui um atributo valor Adicional. O método toString (que exibe o valor do ingresso) da classe IngressoVIP deve considerar que o valor do ingresso é o valor da superclasse somado ao valor Adicional do IngressoVIP.



# atividade proposta 3

Crie um aplicativo que um usuário pode se registrar com seu nome e data de nascimento clicando em “Gravar”.

Caso toque em “Calcular idade”, é exibido um [Toast](#) escrito: “[nome da pessoa], você tem [idade] anos.”.

Caso o usuário opte por buscar pela lista de usuários já gravados, o resultado positivo ou negativo é exibido também através de um Toast.

The wireframe shows a light gray rectangular container. At the top, the word "Pesquisar" is written in bold. Below it is a text input field with the placeholder "Nome do usuário". Underneath the input field is a dark blue button with the text "Buscar". A horizontal line separates this section from the one below. The section below is titled "Registre-se" in bold. It contains two text input fields: the first with the placeholder "Nome" and the second with the placeholder "Data de nascimento". Below these fields are two dark blue buttons: "Gravar" and "Calcular idade".

# atividade proposta 3

Crie uma classe para representar uma pessoa, com os atributos privados de nome, data de nascimento e altura.

Crie um método para calcular a idade da pessoa.

Adicione cada cliente criado em uma lista mutável.

Lembre-se de validar campos nulos e vazios antes de salvar!  
O EditText tem uma função de exibir mensagens de erro no campo.

Pode utilizar uma ScrollView também.

# atividade proposta 4 (desafio)

Crie uma classe Agenda que pode armazenar 10 pessoas e exibir seus nomes em uma ListView. Essa agenda precisa ser capaz de realizar as seguintes operações:

- Salvar alguém na agenda (precisa acrescentar na lista);
- Remover alguém da agenda (precisa remover na lista);
- Buscar pelo nome da pessoa na agenda.

Lembre-se que pode utilizar uma ScrollView em seu .xml para poder “scrollar” pela página independente do seu tamanho.

onde me achar  
clique para abrir o link

[github](#)

[linkedin](#)

[lattes](#)

[e-mail](#)

[instagram](#)



**sandyara  
peres**  
desenvolvedora &  
amante de café &  
mulher.