Índice

[P.O.O Avançado 1](#_ycfi75gcrckw)

[Polimorfismo 1](#_johdsimps8zx)

[Classes abstratas 2](#_zebwp8b9d3e1)

[Interface 3](#_tplsy081vpet)

[Herança Múltipla 3](#_1kjdh2cjq0lz)

[Comandos 1](#_j7779swok7c4)

[Personal Pitch 1](#_smp4u1rs666w)

[5W2H 1](#_b17eh4b0rjsa)

[Pitch Deck 1](#_z0trn9rr27hg)

[Personal Elevator Pitch 1](#_26of3egsfnfb)

[Links e afins 1](#_v49ndfn8hwjl)

[Dúvidas 1](#_huow9eiz5of6)

[Temporários 1](#_sa4ureti8ny)

# 

**Assuntos: P.O.O Avançado, Personal Pitch, Comunicação Não-Violenta.**

**Abertura Alunos:**

Reflexão diária: Descarte de eletrônicos.

Letícia e Jaqueline Paula.

# [P.O.O Avançado](https://docs.google.com/document/d/1URATHqa6fEtU0MXaWDndrIQWNQBna8SPEQtV7eXEW9Y/edit?usp=sharing)

## [Polimorfismo](https://docs.google.com/document/d/1URATHqa6fEtU0MXaWDndrIQWNQBna8SPEQtV7eXEW9Y/edit?usp=sharing)

(Clique no título para ser direcionado as anotações anteriores de polimorfismo)

Polimorfismo é a capacidade de um objeto de decidir que método aplicar a si mesmo.

Embora a mensagem seja a mesma, os objetos podem responder diferentemente.

Ele pode ser aplicado a qualquer método que seja herdado de uma superclasse.

No polimorfismo, a subclasse é capaz de verificar se um método com o nome especificado tem os mesmos parâmetros para poder preencher os dados requeridos.

Caso não tenha esses parâmetros, a superclasse se torna responsável pelo processamento da mensagem, procura um método com esse nome e parâmetros, e se encontrar, chama este método.

No polimorfismo, você pode usar o mesmo método da superclasse, em diversas subclasses, mas permitindo que esse método produza resultados diferentes.

**Exemplo:**

Se você enviar uma mensagem ao objeto **Gerente**, pedindo o **aumentarSalario( )**, ele chamará o método **aumentarSalario( )** da classe **Gerente**, ao invés da classe **Empregado**.

E nesse caso, mesmo os métodos tendo o mesmo método com mesmo nome, o de **Gerente** e o de **Empregado** podem ter parâmetros diferentes.

No polimorfismo, para definir que um método é polimórfico, usamos o **@Override**, para definir que você está sobrescrevendo um método já existente na superclasse.

Em Java, o **@Override** é como dizer: "Estou reescrevendo algo que já existe na classe mãe (ou interface)". É uma forma de garantir que você está substituindo corretamente um método existente, evitando erros. É como seguir um modelo pré-existente para manter as coisas organizadas.

## [Classes abstratas](https://docs.google.com/document/d/1URATHqa6fEtU0MXaWDndrIQWNQBna8SPEQtV7eXEW9Y/edit?usp=sharing)

(Clique no título para ser direcionado as anotações anteriores de classes abstratas)

São classes que devem ser definidas para criar um modelo de implementação.

Elas não podem possuir um objeto instanciado.

Usado apenas para referência genérica.

As classes abstratas são boas ferramentas para se construir sistemas complexos, que possuam múltiplos níveis de hierarquia de classes.

Mesmo com classes abstratas, o polimorfismo continua válido.

Muitas classes podem usar a referência à classe abstrata e utilizar os métodos de cada instância específica.

Classes abstratas podem ter métodos não abstratos.

Classes abstratas podem ter métodos abstratos.

A classe que herda de uma classe abstrata, com métodos abstratos, DEVE redefinir o corpo do método.

Classes não abstratas não podem ter métodos abstratos.

Métodos abstratos só serão nomeados, mas estarão vazios.

Métodos não-abstratos vão ter suas funcionalidades específicas dentro dele.

Método abstrato

**public abstract void metodo();**

Método não-abstrato

**public void metodo(){ //implementar}**

Imagine que uma classe abstrata é como um modelo genérico de bolo, mas sem a receita completa. Ela deixa espaços em branco para os ingredientes específicos.

Outras classes, como "Bolo de Chocolate" ou "Bolo de Baunilha", podem usar esse modelo e preencher os espaços em branco com seus próprios ingredientes para se tornarem bolos completos e deliciosos.

A classe abstrata é como um guia para criar bolos, mas por si só não é um bolo pronto.

Na classe abstrata Bolo, um BoloBaunilha e um BoloChocolate podem ter diferentes métodos fazBolo(){[...]} com diferentes implementações - possuem receitas diferentes, apesar de serem bolos

## [Interface](https://docs.google.com/document/d/1URATHqa6fEtU0MXaWDndrIQWNQBna8SPEQtV7eXEW9Y/edit?usp=sharing)

(Clique no título para ser direcionado as anotações anteriores de interface)

Interface é um “”contrato”” no qual o objeto “”se compromete”” a **implementar todos os métodos**.

A interface é uma classe abstrata com métodos abstratos. O programador define interface, e o compilador “”enxerga”” uma classe abstrata com métodos abstratos.

A interface **não deve** ter corpo de métodos.

**NÃO SE CRIA UMA CLASSE PARA INTERFACE, E SIM UM INTERFACE EM SI!!!!!**

### [Herança Múltipla](https://docs.google.com/document/d/1URATHqa6fEtU0MXaWDndrIQWNQBna8SPEQtV7eXEW9Y/edit?usp=sharing)

(Clique no título para ser direcionado as anotações anteriores de herança múltipla)

Herança múltipla é quando a superclasse herda mais de uma superclasse. O Java não suporta herança múltipla.

### Comandos

**modificador class *nomeSubclasse* implements *nomeInterface* {**

***nomeMetodo1* (parametros {**

**corpo do método 1**

**}**

***nomeMetodo2* (parametros {**

**corpo do método 2**

**}**

**}**

Comando exemplo de criação de interface

**modificador class *nomeSubclasse*  extends *nomeSuperclasse***

**implements *nomeInterface1***

***nomeInterface2***

Sintaxe de criação de interface

**implements**

Comando para implementar uma interface

# [Personal Pitch](https://docs.google.com/document/u/0/d/1-_7-TT7nGNcAxLg_tvOJ-g46Um5tUmWR7HBEuTCo5dc/edit)

(Clique no link para ser direcionado as anotações anteriores de Personal Pitch)

### 5W2H

Perguntas a serem respondidas para determinar pontos chave do seu pitch.

* **Why** (Por quê)
* **What** (O que)
* **Who** (Quem)
* **Where** (Onde)
* **When** (Quando)
* **How** (Como)
* **How Much** (Quanto)

## Pitch Deck

Um pitch maior e mais completo, de cerca de 5 minutos. Geralmente usado para apresentação de projetos.

## Personal Elevator Pitch

Um pitch mais curto, de 30 segundos a 2 minutos. Geralmente mais usado para apresentações rápidas, pessoais, e de networking.

# Links e afins

**Vídeo Steve Cutts - Man, Reflexão diária:**

<https://www.youtube.com/watch?v=WfGMYdalClU>

**Apostila Personal Pitch (crie uma cópia):**

<https://docs.google.com/document/d/1-hwsINQVpqiNMHvbIf06ZbA9QH21zNy4riZ5Uj3wSNM/edit>

# 

# Dúvidas