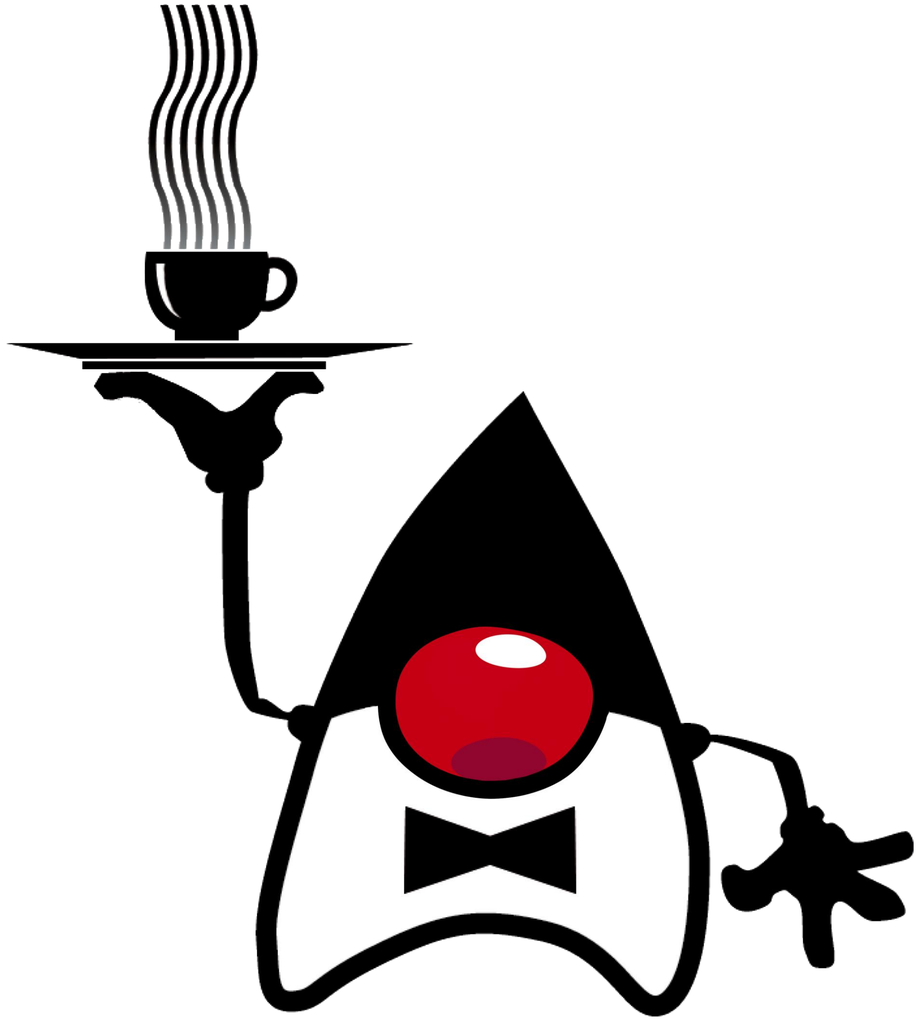
Trilha Java

Encontro 21 – (POO) Polimorfismo



vnt/school
powered by venturus

Recapitulação

1. Herança.
2. Árvore Hierárquica.
3. Tipos de Herança.

vnt/school
powered by venturus

Agenda

1. Introdução
2. Polimorfismo de Sobreposição
3. Assinatura de Classe
4. Polimorfismo de Sobrecarga
5. Exemplo
6. Exercícios

vnt/school
powered by venturus

Introdução

Polimorfismo

vnt/school

powered by venturus

POO

A programação orientada
a objetos é baseada em quatro
pilares .

Neste capítulos iremos abordar o ultimo tópico da base de POO, **Polimorfismo**.

vnt/school
powered by venturus

Polimorfismo

Dicionário:
Propriedade ou estado do que se apresenta sob várias formas.

POO:
Permite que um **mesmo nome** represente **vários comportamentos** diferentes.

vnt/school
powered by venturus

Polimorfismo

É a capacidade de um **objeto** ser referenciado de **formas diferentes** e com isso **realizar** as **mesmas tarefas** (ou chamadas de métodos) de **diferentes formas**.

Uma **subclasse** pode redefinir (sobrescrever) um **método herdado**.

vnt/school
powered by venturus

Conceitos

Polimorfismo

Habilidade de um objeto executar um método e obter o **comportamento correto** (ou desejado) em pontos apropriados do programa

Padrões de Projeto

Um padrão é uma descrição ou **modelo** a ser usado por uma **equipe.**

Padrões de Projeto facilitam a **reutilização** ao estabelecer um **vocabulário comum** no projeto

Refatoração

Processo de **alterar o código** para melhorar sua estrutura sem alterar seu comportamento.

Significa tornar o **código mais compreensível** e bem estruturado

Facilita o trabalho em **equipe** e a **manutenção.**

vnt/school
powered by venturus

Polimorfismo

A principal aplicação do **Polimorfismo** está na definição de regras e **padrões de projeto.**

Os dois principais modelos deste objetivo incluem ***classes abstratas*** que podem ser implementadas explicitamente (e utilizadas via **Herança**) ou implicitamente (através de ***interfaces***)

vnt/school
powered by venturus

Polimorfismo

Classes Abstratas

São *modelos para outras classes.*

Não podem ser instanciadas.

Classes mais especializadas herdam sua implementação.

<extends>

Interfaces

Interfaces são *padrões* definidos *através de especificações.*

Seus métodos são definidos, mas não implementados.

<implements>

vnt /school
powered by venturus

Assinatura do método

vnt/school

powered by venturus

Assinatura do método

Todo método tem uma **assinatura**.

É importante identificar quais **métodos** possuem a **mesma assinatura**.

Ter a mesma assinatura, significa ter mesmas **quantidades e os tipos de parâmetros**.

No exemplo ao lado, observe que os dois primeiros métodos possuem a mesma assinatura.

vnt/school
powered by venturus

public metodo calcMedia(n1: Real, n2: Real): Real

public metodo calcMedia(v1: Real, v2: Real): Int

public metodo calcMedia(bim: Int, n1: Real, n2: Real): Real

public metodo calcMedia(n1: Real, n2: Real, n3: Real, n4: Real): Real

public metodo calcMedia(medMin: Real, medMax: Real, sit: Caractere, bim: Int): Caractere

Tipos de Polimorfismo

Polimorfismo de Sobreposição

vnt/school
powered by venturus

Tipos de Polimorfismo

Sobreposição

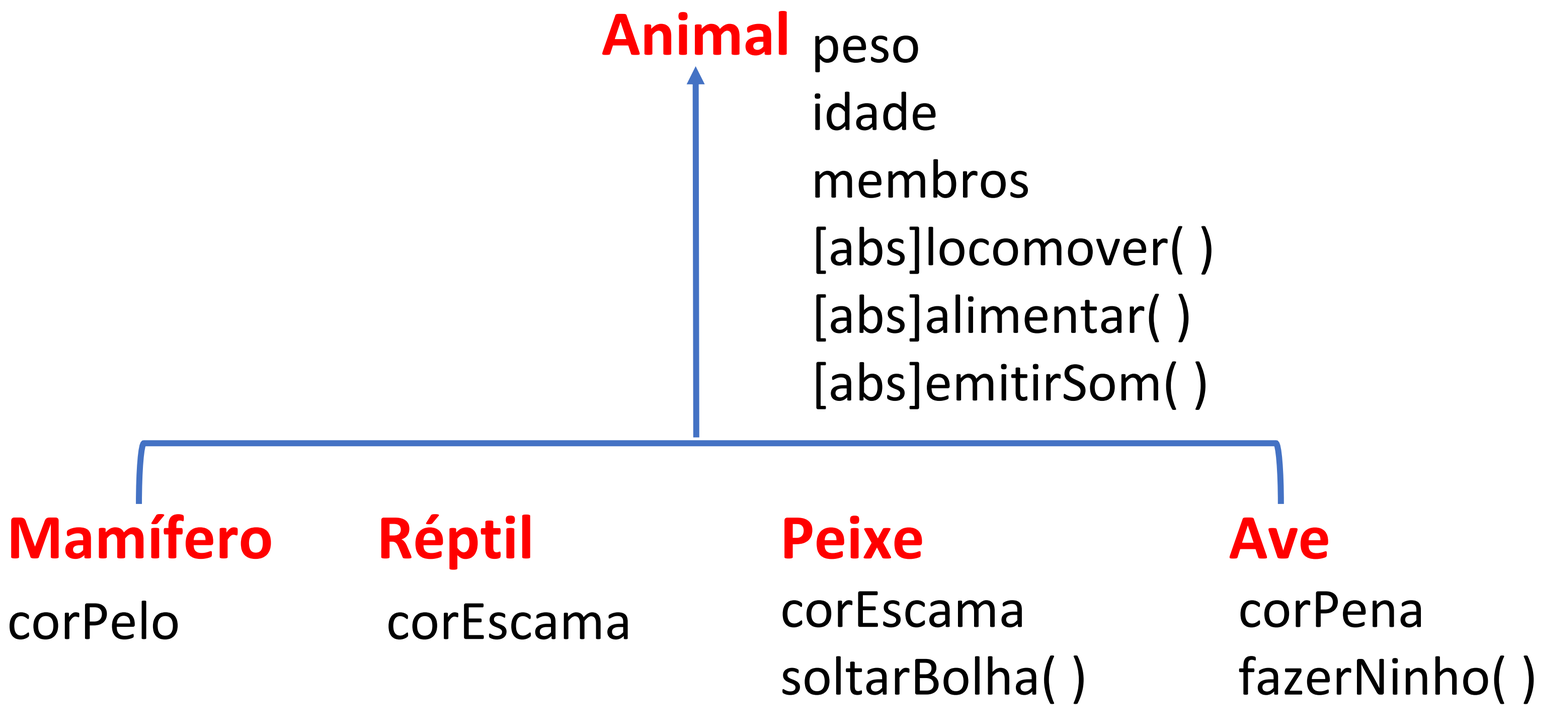
Sobrecarga

O tipo **Sobreposição** é mais utilizado que o tipo Sobrecarga.

Polim. Sobreposição

Suponha a **SuperClasse**

Animal com suas **SubClasses**.



vnt/school
powered by venturus

Polim. Sobreposição

Inicialmente construimos a **SuperClasse** Animal.

Veja que ela é **abstrata**.

```
classe abstrata Animal
protegido Real peso
protegido Inteiro idade
protegido Inteiro membros
publico metodo abstrato locomover()
publico metodo abstrato alimentar()
publico metodo abstrato emitirSom()
FimClasse
```

vnt/school
powered by venturus

Polim. Sobreposição

Em seguida construímos a primeira subclasse Mamífero.

É inserido a ideia de polimorfismo com a sobreposição. @sobrepor

```
Classe Mamifero estende Animal
Privado String corPelo
@Sobrepor
publico metodo locomover()
    Escreva("Correndo")
FimMetodo
@Sobrepor
publico metodo alimentar()
    Escreva("Bebendo leite")
FimMetodo
@Sobrepo
publico metodo emitirSom()
    Escreva("Som de Mamífero")
fimMetodo
FimClasse
```

Polim. Sobreposição

A segunda subclasse é **Reptil**, que também estende a **Classe Animal**.

É inserido a ideia de polimorfismo com a sobreposição. @sobrepor

```
Classe Reptil estende Animal
Privado String corEscama
@Sobrepor
publico metodo locomover()
    Escreva("Rastejando")
FimMetodo
@Sobrepor
publico metodo alimentar()
    Escreva("Comendo Vegetais")
FimMetodo
@Sobrepo
publico metodo emitirSom()
    Escreva("Som de Reptil")
fimMetodo
FimClasse
```

Polim. Sobreposição

A terceira subclasse é **Peixe**, que também estende a Classe **Animal**.

Apresenta o método “**soltarBolha**”.

É inserido a ideia de polimorfismo com a sobreposição. @sobrepor

vnt/school
powered by venturus

```
Classe Peixe estende Animal
Privado String corEscama
@Sobrepor
publico metodo locomover()
    Escreva("Nadando")
FimMetodo
@Sobrepor
publico metodo alimentar()
    Escreva("Comendo substancias")
FimMetodo
@Sobrepo
publico metodo emitirSom()
    Escreva("Peixe não faz som")
FimMetodo
publico metodo soltarBolha()
    Escreva("Soltou uma bolha")
fimMetodo
FimClasse
```

Polim. Sobreposição

A quarta subclasse é **Ave**, que também estende a Classe Animal.

Apresenta o método “**fazerNinho**”.

É inserido a ideia de polimorfismo com a sobreposição. @sobrepor

vnt/school
powered by venturus

```
Classe Ave estende Animal
Privado String corPena
@Sobrepor
publico metodo locomover()
    Escreva("Voando")
FimMetodo
@Sobrepor
publico metodo alimentar()
    Escreva("Comendo frutas")
FimMetodo
@Sobrepor
publico metodo emitirSom()
    Escreva("Som de Ave")
FimMetodo
publico metodo fazerNinho()
    Escreva("Construiu um ninho")
fimMetodo
FimClasse
```

Polim. Sobreposição

Onde está o erro?

A classe **Animal** **não** pode ser **instanciada**. Todas as demais classes podem ser instanciadas.

Veja que apesar de termos o método locomover para todas as classes, a **ação** de cada um **é diferente**.

A mesma ideia serve para os demais métodos.

vnt/school
powered by venturus

```
//Programa Principal
a = new Animal( )
m = new Mamifero( )
r = new Reptil( )
p= new Peixe( )
a = new Ave( )

m.setPeso(85.3)
m.setTdade(2)
m.setMembros(4)
m.locomover( ) //correndo

p.setPeso(1.3)
p.setTdade(1)
p.setMembros(0)
p.locomover( ) //nadando
```

Polimorfismo

Isso é polimorfismo: o mesmo método faz coisas diferentes.

Polimorfismo de sobreposição: acontece quando substituímos um método de uma superClasse na sua subClasse, usando a **mesma assinatura.**

Polimorfismo

Vamos Praticar!!

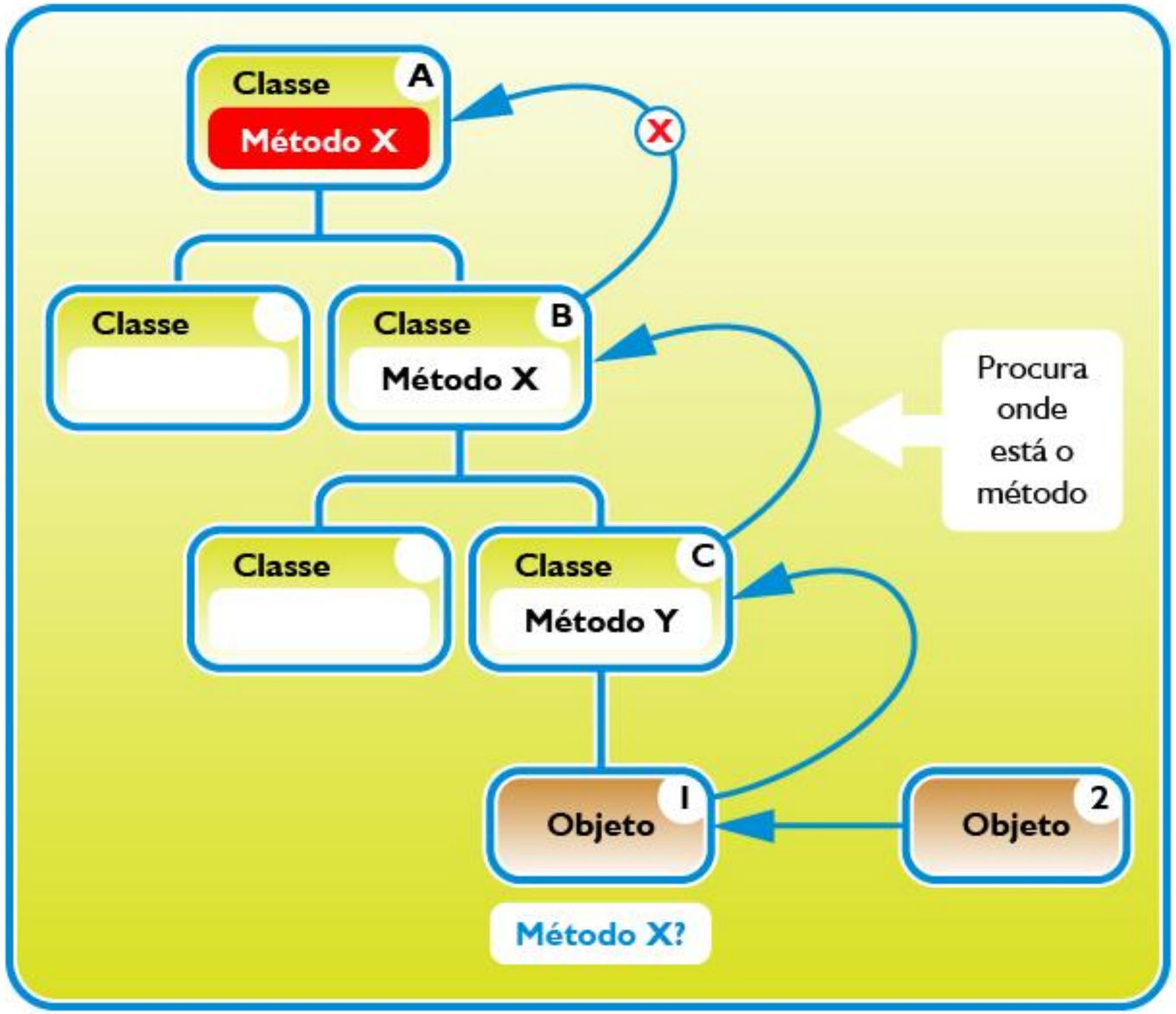
Veja no **NetBeans** a implementação do código.

.

vnt/school
powered by venturus

Polimorfismo

Polimorfismo

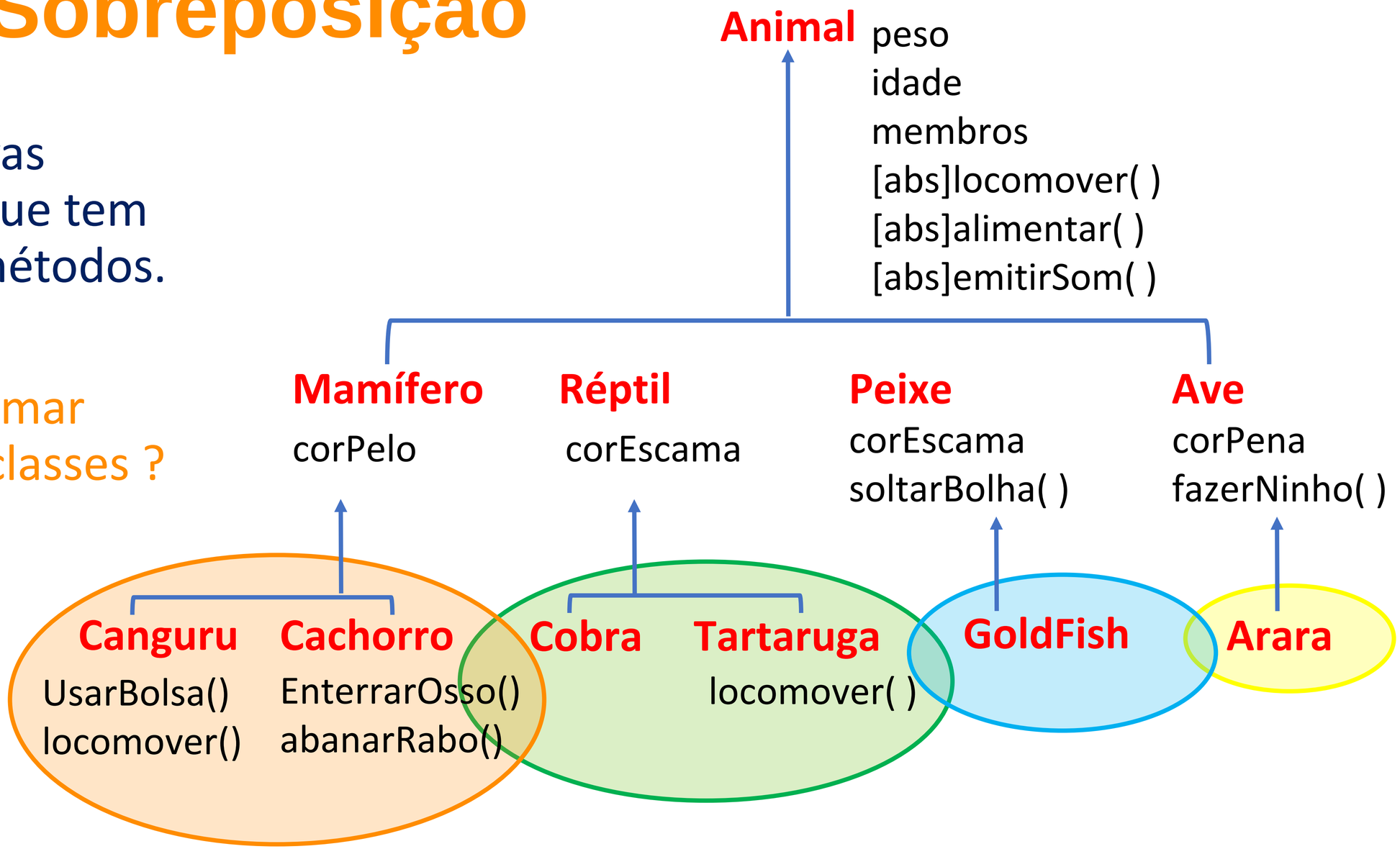


Coffee time!

Polim. Sobreposição

Suponha novas subclasses, que tem atributos e métodos.

Como programar essas novas classes ?



vnt/school
powered by venturus

Polim. Sobreposição

Veja que as classes “**Canguru**” e “**Cachorro**” estendem a classe “**Mamífero**”.

O método “**locomover()**” em “**Canguru**” sobrepõe o mesmo método em “Mamífero”.

vnt/school
powered by venturus

```
Classe Canguru estende Mamifero
publico metodo usarBolsa()
    Escreva("Usando bolsa")
FimMetodo
@Sobrepor
publico metodo locomover()
    Escreva("Saltando")
FimMetodo
FimClasse
```

```
Classe Cachorro estende Mamifero
publico metodo enterrarOsso()
    Escreva("Enterrando Osso")
FimMetodo
publico metodo abanarRabo()
    Escreva("Abanando Rabo")
FimMetodo
FimClasse
```

Polim. Sobreposição

As classes “**Cobra**”, “**GoldFish**” e “**Arara**” não tem nada a ser implementado.

A classe “**Tartaruga**” tem o método “**locomover**” que **sobrepõe** o mesmo método em “**Réptil**”.

vnt/school
powered by venturus

```
Classe Cobra estende Reptil

FimClasse

Classe Goldfish estende Peixe

FimClasse

Classe Arara estende Ave

FimClasse

Classe Tartaruga estende Reptil
@Sobrepor
publico metodo locomover()
    Escreva("Andando beeeeeem devagar")
FimMetodo
FimClasse
```

Polim. Sobreposição

Agora os **objetos** são **instanciados**.

Alguns **métodos** são **sobrepostos** devido a ideia do polimorfismo.

Classes abstratas **não** podem ser **instanciadas**.

vnt/school
powered by venturus

```
//Programa Principal
m = new Mamifero( )
c = new Canguru( )
k = new Cachorro( )

m.setPeso(5.3)
m.setTdade(8)
m.setMembros(4)
m.locomover( ) //correndo

c.setPeso(56.0)
c.setTdade(3)
c.setMembros(4)
c.locomover( ) //saltando

k.setPeso(5.0)
k.locomover( ) //correndo
```

Tipos de Polimorfismo

Polimorfismo de Sobrecarga

vnt/school
powered by venturus

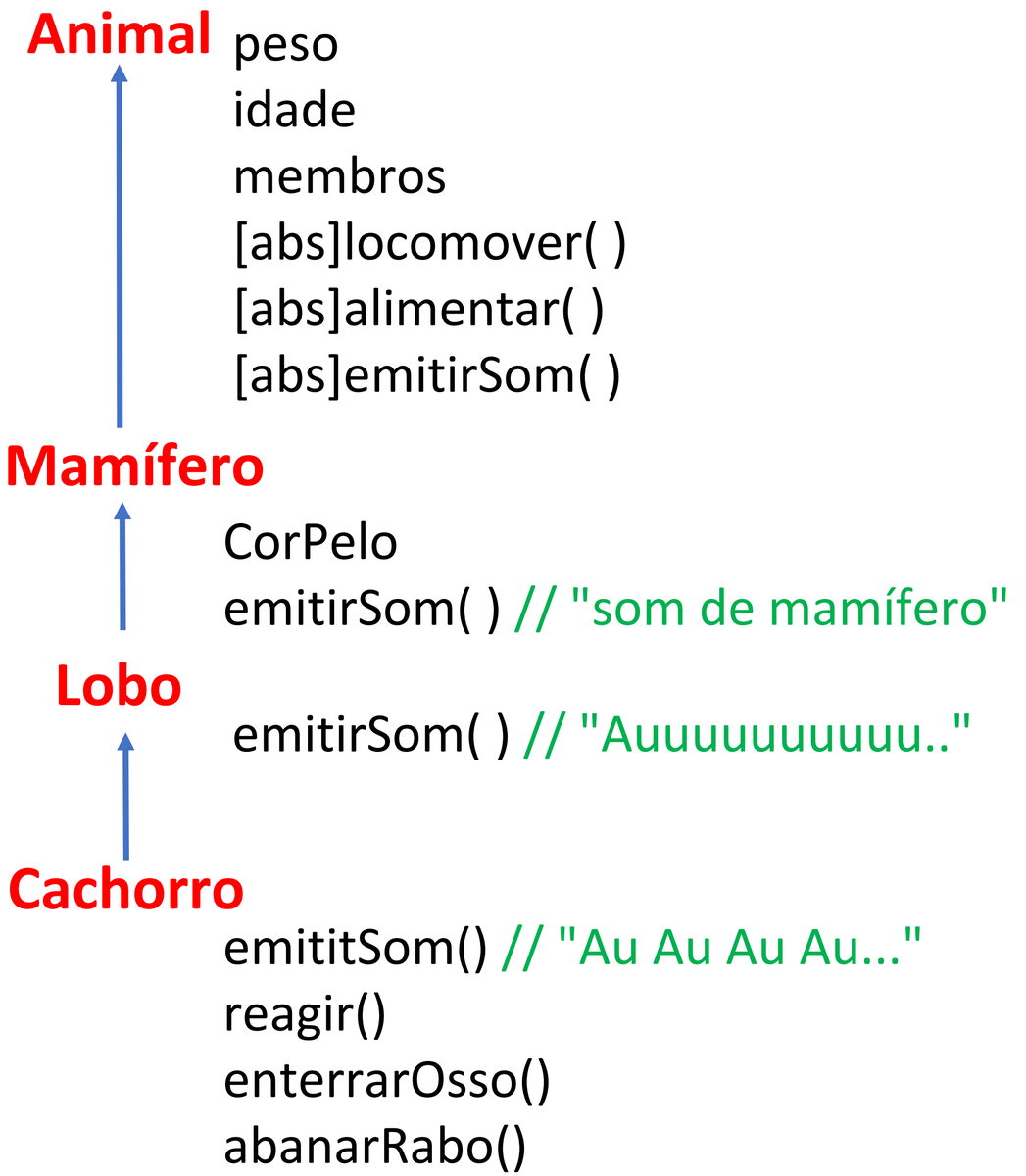
Polim. Sobrecarga

O método “**emitirSom**”:

Possui **assinaturas semelhantes**.

Estão em **classes diferentes**.

Na **superClasse** é **abstrato**.
Não pode ser instanciado.



vnt/school
powered by venturus

Polim. Sobrecarga

Todos os métodos “**emitirSom**” necessita **@Sobrepor** o mesmo método na SuperClasse “**Animal**”.

Todos os métodos possui a **mesma assinatura** e estão em **classes diferentes**.

vnt/school
powered by venturus

```
Classe abstrata Animal
publico metodo abstrato emitirSom()
...
Classe Mamifero estende Animal
protegido String corPelo
@Sobrepor
publico metodo emitirSom()
    Escreva("Som de Mamifero")
FimMetodo
FimClasse
Classe Lobo estende Mamifero
@Sobrepor
publico metodo emitirSom()
    Escreva("Auuuuuuuuu")
FimMetodo
FimClasse
Classe Cachorro estende Lobo
@Sobrepor
publico metodo emitirSom()
    Escreva("Au..Au..Au..Au..")
FimMetodo
FimClasse
FimClasse
```

Polim. Sobrecarga

Suponha que o método “**reagir()**”, da **classe cachorro**, tenha **comportamentos diferentes**.

Para tipo de ação o “**reagir()**” pode ser de forma diferente.

Isso é também **polimorfismo**.

Reagir( )	
Falar frase	Agradável: abanar e latir Agressiva: rosnar
Horário do dia	Manhã: abanar Tarde: abanar e latir Noite: ignorar
dono	É dono: abanar Não é: rosnar e latir
Idade e peso	Novo e leve: abanar Novo e pesado: latir Velho e leve: rosnar Velho e pesado: ignorar

vnt/school
powered by venturus

Polim. Sobrecarga

O mesmo método “**reagir()**” possui **assinaturas diferentes**.

Todos os métodos “**reagir()**” se encontra dentro da mesma classe “**Cachorro**”.

Este é um exemplo de **Polimorfismo de Sobrecarga**.

```
Classe Cachorro estende Lobo
publico metodo reagir(frase: String)
    ...
FimMetodo
publico metodo reagir(hora, min: Inteiro)
    ...
FimMetodo
publico metodo reagir(dono: Logico)
    ...
FimMetodo
publico metodo reagir(idade: Inteiro, peso: Real)
    ...
FimMetodo
FimClasse
```

vnt/school
powered by venturus

Polim. Sobrecarga

Para cada ação, a classe “**cachorro**” tem uma reação com o método “**reagir()**”.

Todos os métodos “**reagir()**” estão dentro da **mesma classe**.

vnt/school
powered by venturus

```
Classe Cachorro estende Lobo
publico metodo reagir(frase: String)
    Se(frase="toma comida" ou frase="olá")
        Escreva("Abanar e latir")
    Senão
        Escreva("Rosnar")
FimMetodo
publico metodo reagir(hora, min: Inteiro)
    Se(hora < 12)
        Escreva("Abanar")
    SenãoSe(hora>=18)
        Escreva("Ignorar")
    Senao
        Escreva("Abanar e latir")
FimMetodo
publico metodo reagir(dono: Logico)
    Se(dono=verdadeiro")
        Escreva("Abanar")
    Senão
        Escreva("Rosnar e Latir")
FimMetodo
FimClasse
```

Polim. Sobrecarga

O método “**reagir()**” está presente várias vezes dentro da **mesma classe** e com **assinaturas diferentes**.

Isso é **polimorfismo** de **Sobrecarga**.

vnt/school
powered by venturus

```
Classe Cachorro estende Lobo
publico metodo reagir(idade,, peso: Inteiro)
    Se(idade < 5)
        Se(peso < 10)
            Escreva("Abanar")
        Senão
            Escreva("Latir")
        fimSe
    Senão
        Se(peso<10)
            Escreva("Rosnar")
        Senao
            Escreva("Ignorar")
        fimSe
    fimSe
FimMetodo
FimClasse
```

Polim. Sobrecarga

Ao criar o objeto “**Cachorro**”, observe as **diversas reações** que podem acontecer.

Polimorfismo de Sobrecarga, pois o mesmo método se encontra dentro da **mesma classe** e com **diversas assinaturas**.

```
//Programa Principal
k = new Cachorro( )

k.reagir("Olá")                 //Abanar e Latir
k.reagir("Vai apanhar")   //Rosnar
k.reagir(11, 45)              //Abanar
k.reagir(21, 00)              //Ignorar
k.reagir(verdadeiro)       //Abanar
k.reagir(falso)               //Rosnar e Latir
```

vnt/school
powered by venturus

Polim. Sobrecarga

Polimorfismo de Sobreposição

Assinaturas iguais

Classes diferentes

Polimorfismo de Sobrecarga

Assinaturas diferentes

Mesma classe.

vnt/school

powered by venturus

Vamos Praticar!!

Exercício

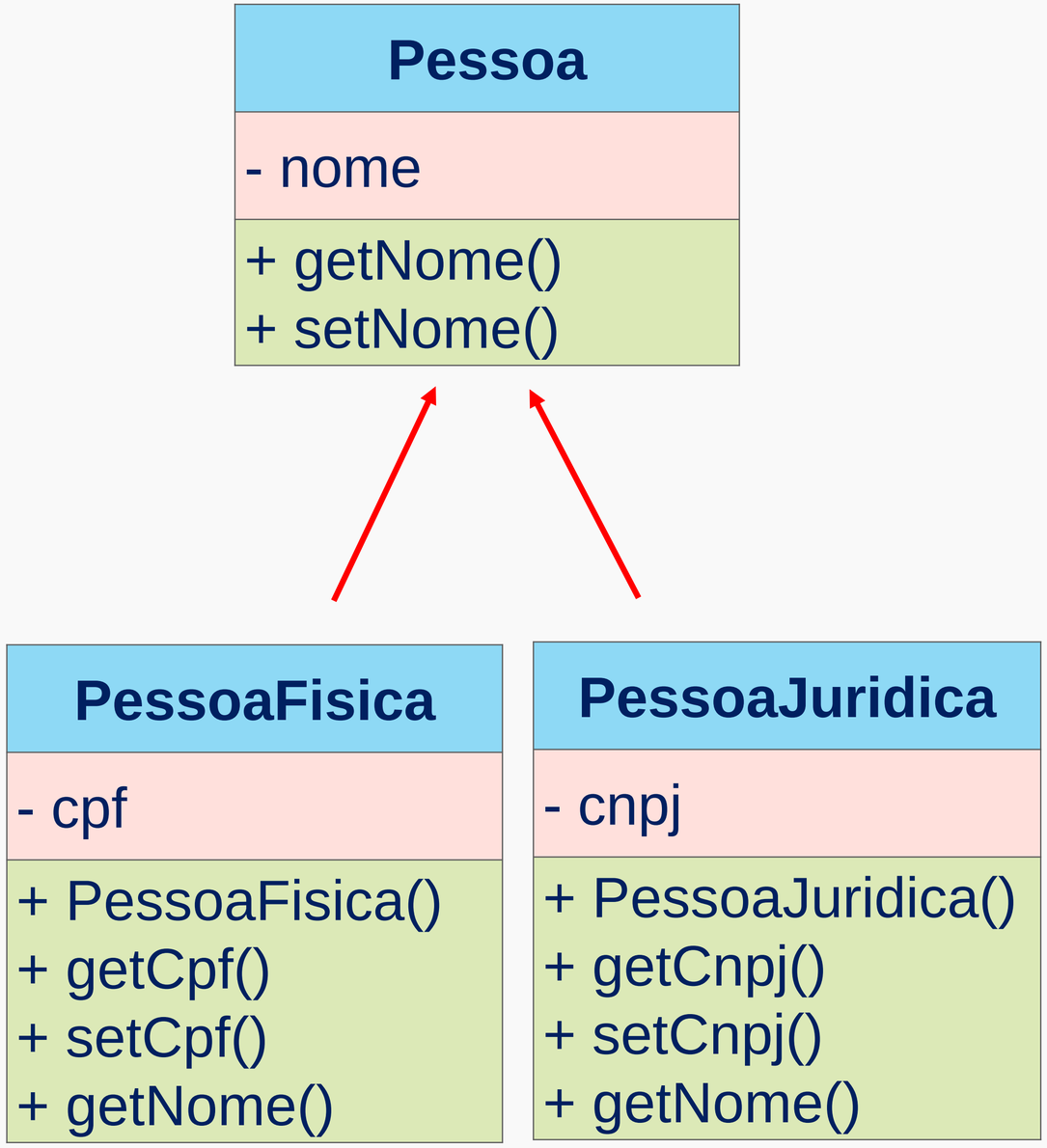
Polimorfismo

vnt/school

powered by venturus

Atividade 1

Escreva um código para o diagrama apresentado ao lado. Use o conceito de polimorfismo para sobrepor o método getNome(). Instancie os objetos PessoaFisica e PessoaJuridica. O retorno deve conter o nome e o cpf, se pessoa física. Ou o nome e cnpj se pessoa jurídica.



Exercícios

```java
package aula21;
public class Atividade2 {
    public static void main(String[] args) {
        PessoaFisica fisica = new PessoaFisica();
        fisica.setNome("Joao");
        fisica.setCpf(12345678901L);

        PessoaJuridica juridica = new PessoaJuridica();
        juridica.setNome("SeTecnologia");
        juridica.setCnpj(100010001234567 8L);

        Pessoa[] pessoas = new Pessoa[2];
        pessoas[0] = fisica;
        pessoas[1] = juridica;

        for (Pessoa pessoa : pessoas) {
            System.out.println(pessoa.getNome());
        }
```

??

vnt/school
powered by venturus


Review
e
Preview

POO - Benefícios

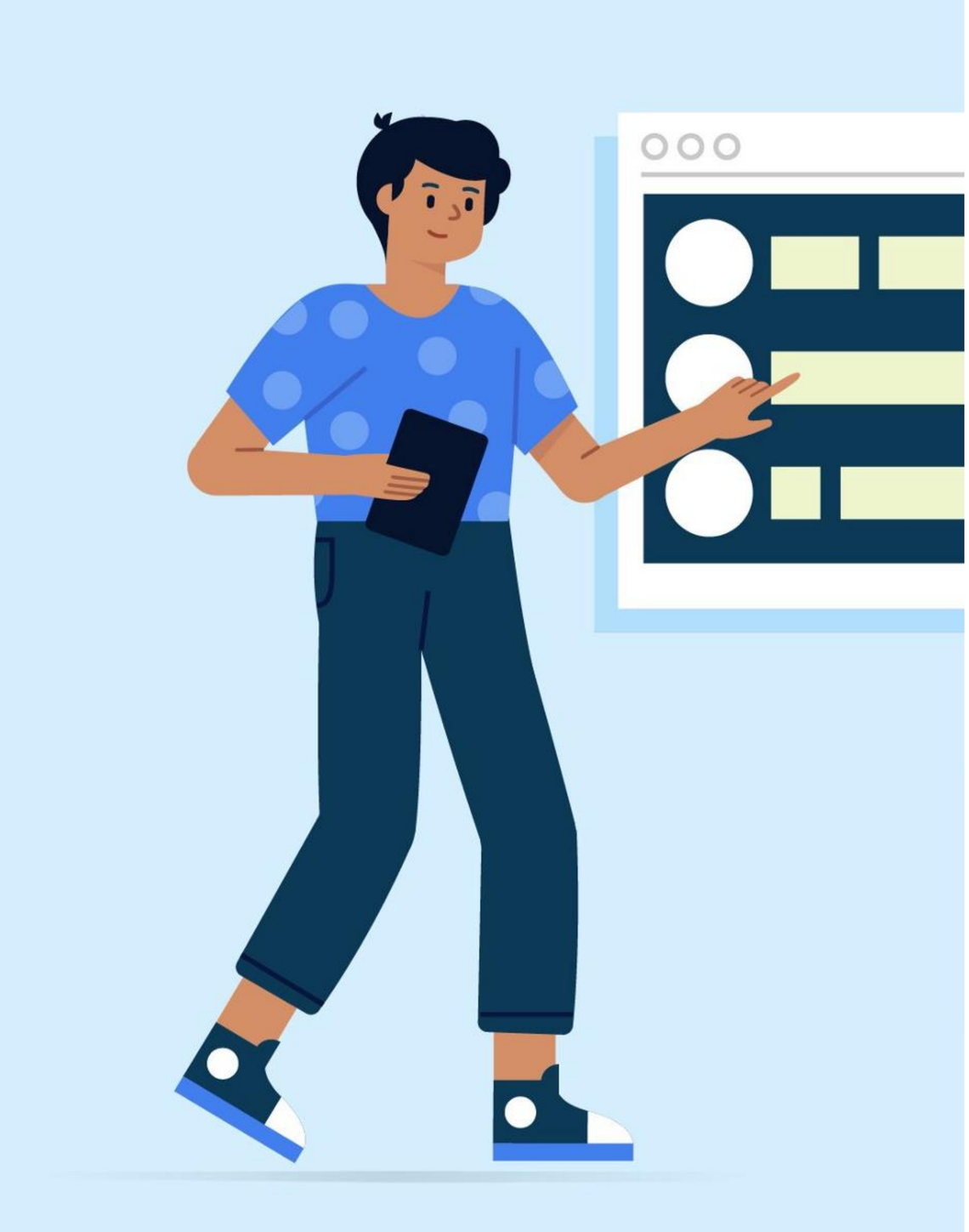
Propõe uma representação mais fácil de ser compreendida e realista.

Reutilização de código.

Otimização do tempo
de desenvolvimento.

Facilidade na leitura e manutenção de código.

vnt/school
powered by venturus



Comunidade VNT

Dica de hoje

O link abaixo apresenta um bom texto sobre o conceito de Polimorfismo. Este é um tópico muito importante abordado em programação orientada a objeto. O texto além de apresentar conceitos importantes, aborda também uma prática que contribui para o aprendizado.

http://www.universidadejava.com.br/java/java-polimorfismo/

Boa leitura!!

Referências

[1] A. Goldman, F. Kon, Paulo J. S. Silva; Introdução à Ciência da Computação com Java e Orientação a Objetos (USP). 2006. Ed. USP.
[2] Algoritmo e lógica de programação. Acessado julho/2022: https://visualg3.com.br/
[3] G. Silveira; Algoritmos em Java; Ed. Casa do Código.
[4] M. T. Goodrich, R. Tamassia; Estrutura de dados e algoritmos em Java. Ed Bookman. 2007.
[5] Algoritmo e lógica de programação. Acessado julho/2022: https://www.cursoemvideo.com/
[6] P. Silveira, R. Turini; Java 8 Pratico: lambdas, streams e os novos recursos da linguagem. Ed. Casa do Código.
[7] Linguagem Java: Curso acessado em agosto/2022: https://www.udemy.com/
[8] Linguagem Java: Curso acessado em setembro/2022: https://www.cursoemvideo.com/