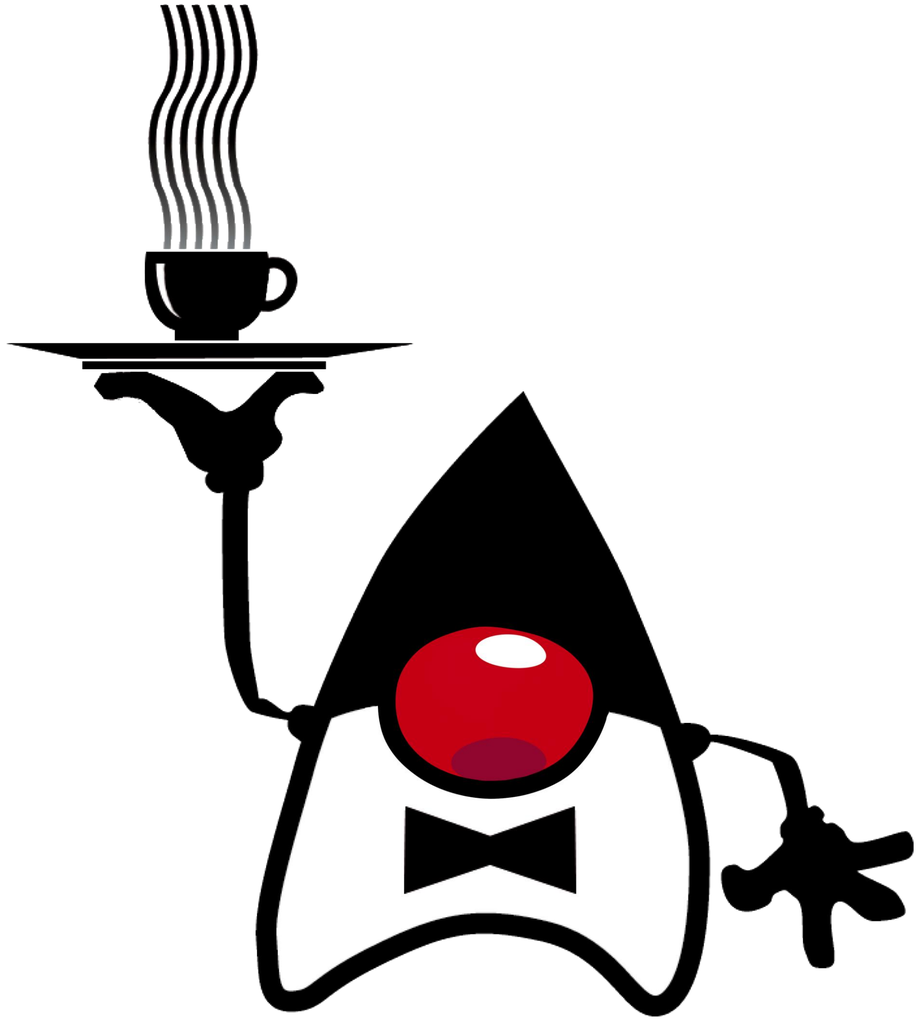
Trilha Java

Encontro 19 – Herança



vnt/school

powered by venturus

Recapitulação

1. Encapsulamento.
2. Relacionamento entre classes.
3. Agregação

vnt/school
powered by venturus

Agenda

1. Herança.
2. Árvore Hierárquica.
3. Tipos de Herança.
4. Exemplos.
5. Atividades.

vnt/school
powered by venturus

Herança

Características e comportamentos

vnt/school
powered by venturus

Pilares de POO

O conceito de **abstração e encapsulamento** já foram apresentados.

Agora iremos abordar:

Herança!

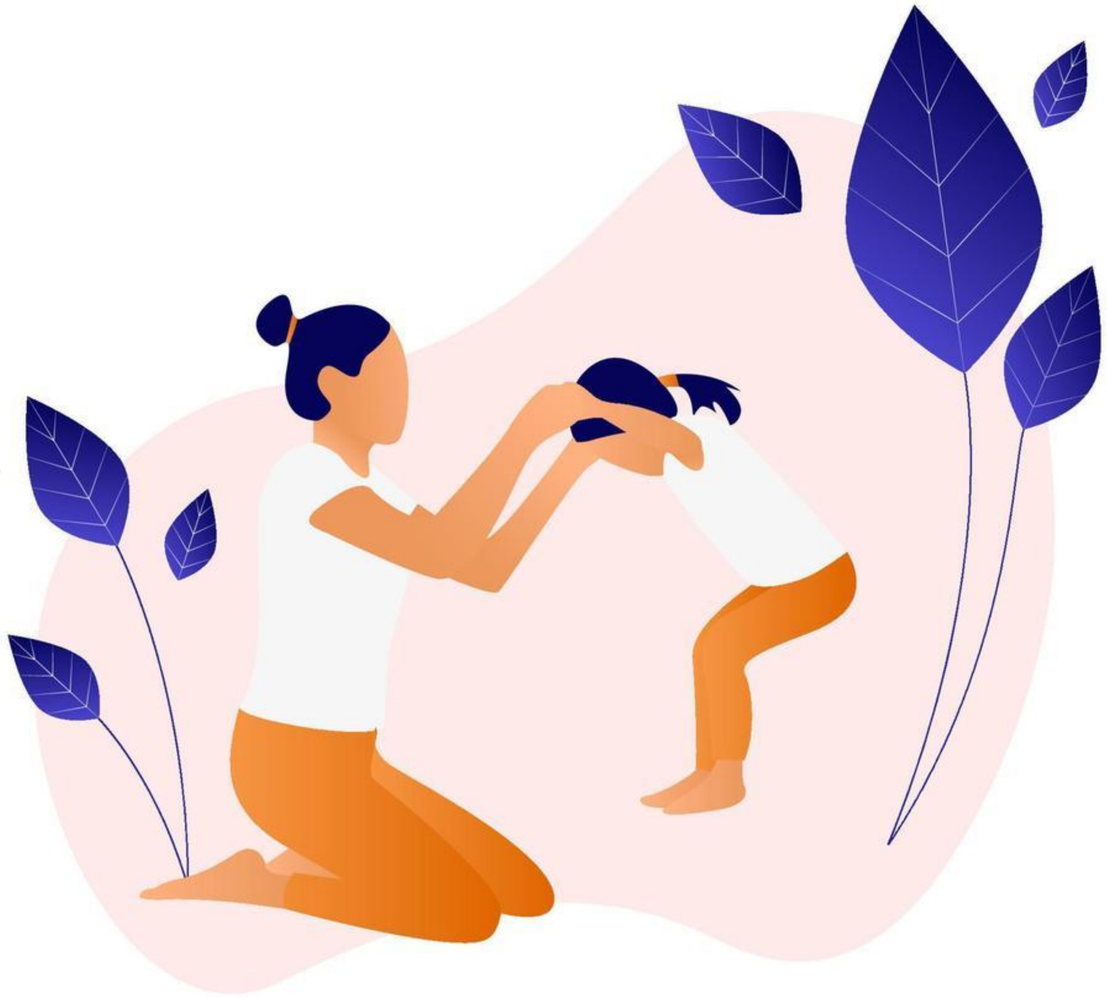
vnt/school
powered by venturus

Herança

Mãe e filhos.

Normalmente os filhos trazem características e comportamentos da mãe.

Este é um conceito de Herança.



Herança

As **Classes** são consideradas mães que podem gerar filhas.

As filhas podem herdar **características** e **comportamentos** das mães.

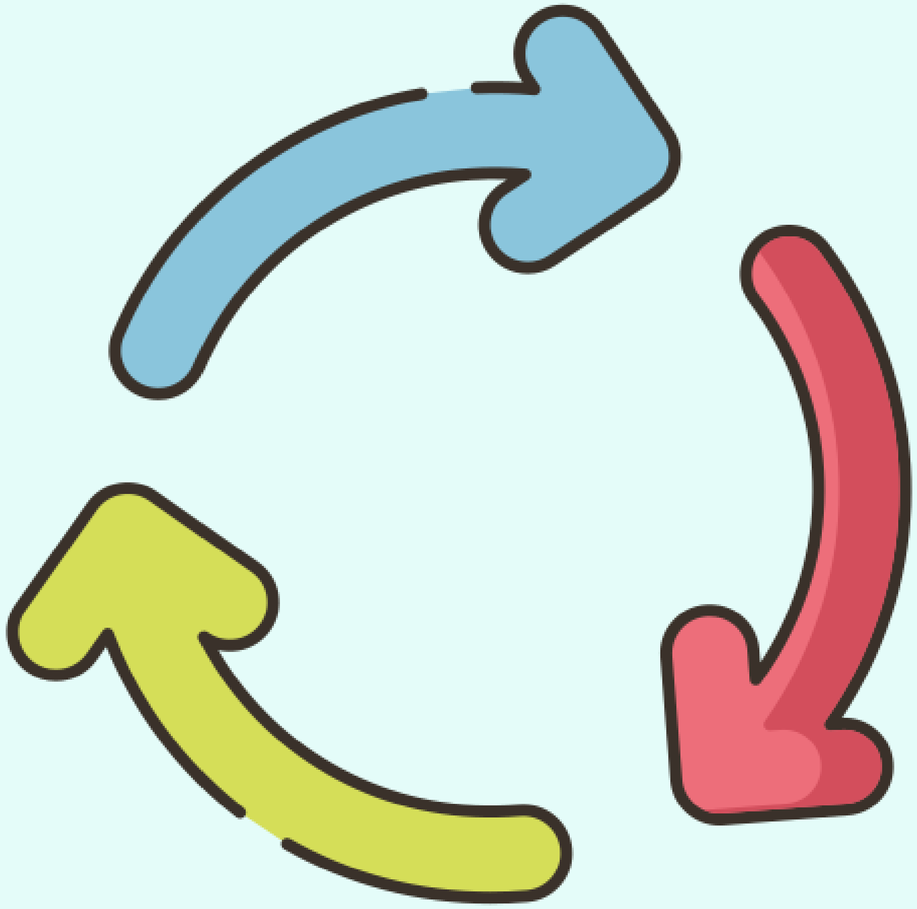
vnt/school
powered by venturus

Herança

Permite basear uma nova classe na definição de uma outra classe previamente existente.

Vantagens:

* Reuso
* Polimorfismo



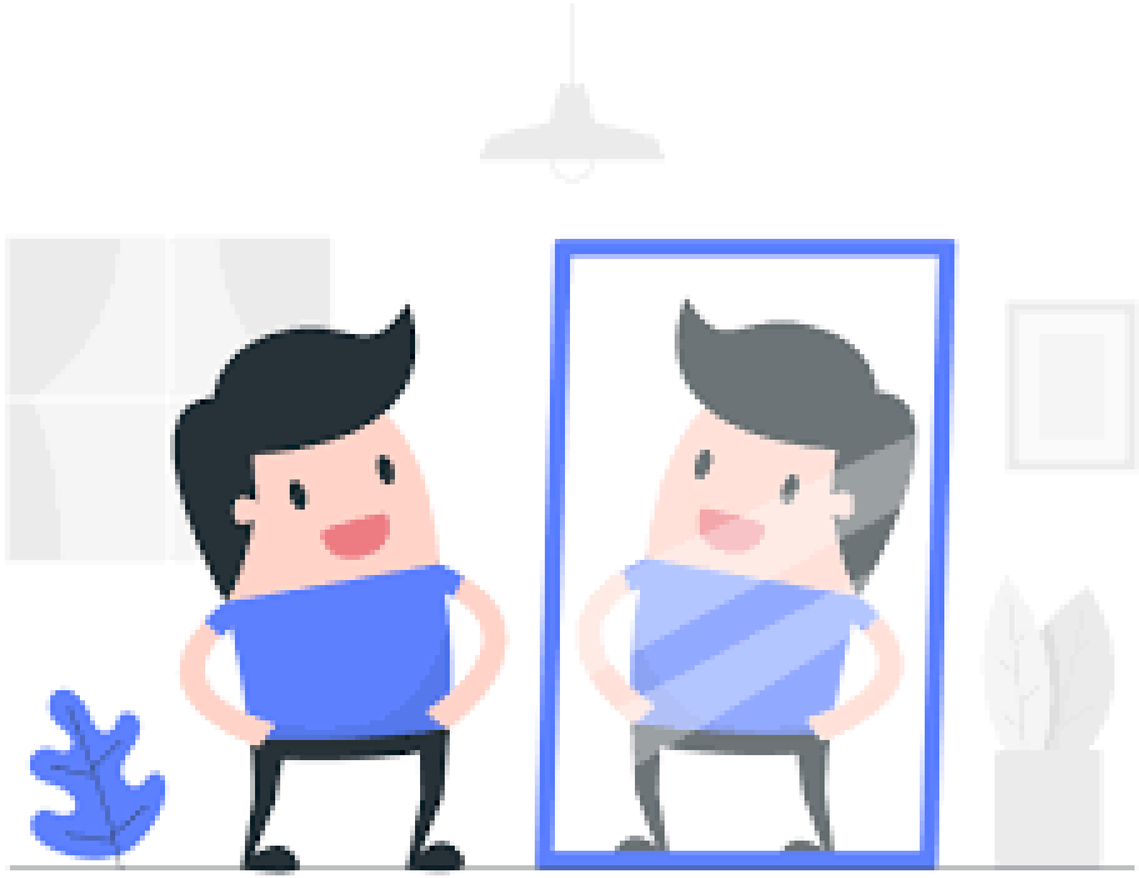
vnt/school
powered by venturus

Herança

A **herança** será aplicada tanto para as **características** quanto para os **comportamentos**.

Sintaxe:

class A extends B



vnt/school
powered by venturus

Herança

Imagine três pessoas, sendo um **aluno**, um **professor** e um **funcionário**.

Todas essas **pessoas**, podem apresentar **características** e **comportamentos** semelhantes.

Herança

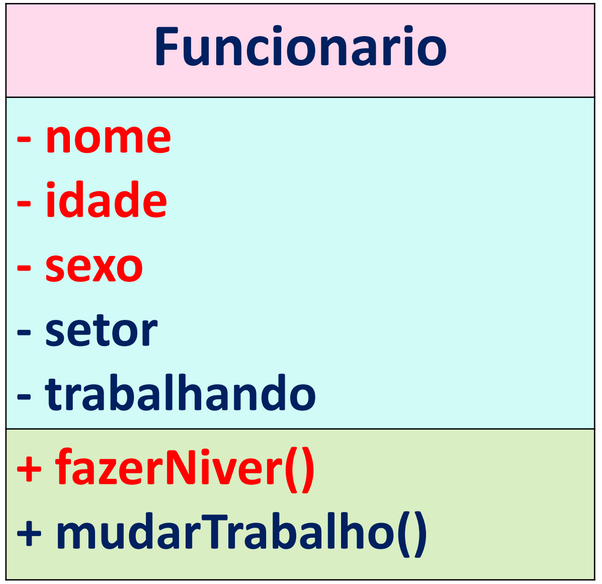
Todas as pessoas apresentam:

Atributos

Métodos

Em vermelho está destacado os atributos e métodos que ambos compartilham.

O que isso sugere?



vnt/school
powered by venturus

Herança

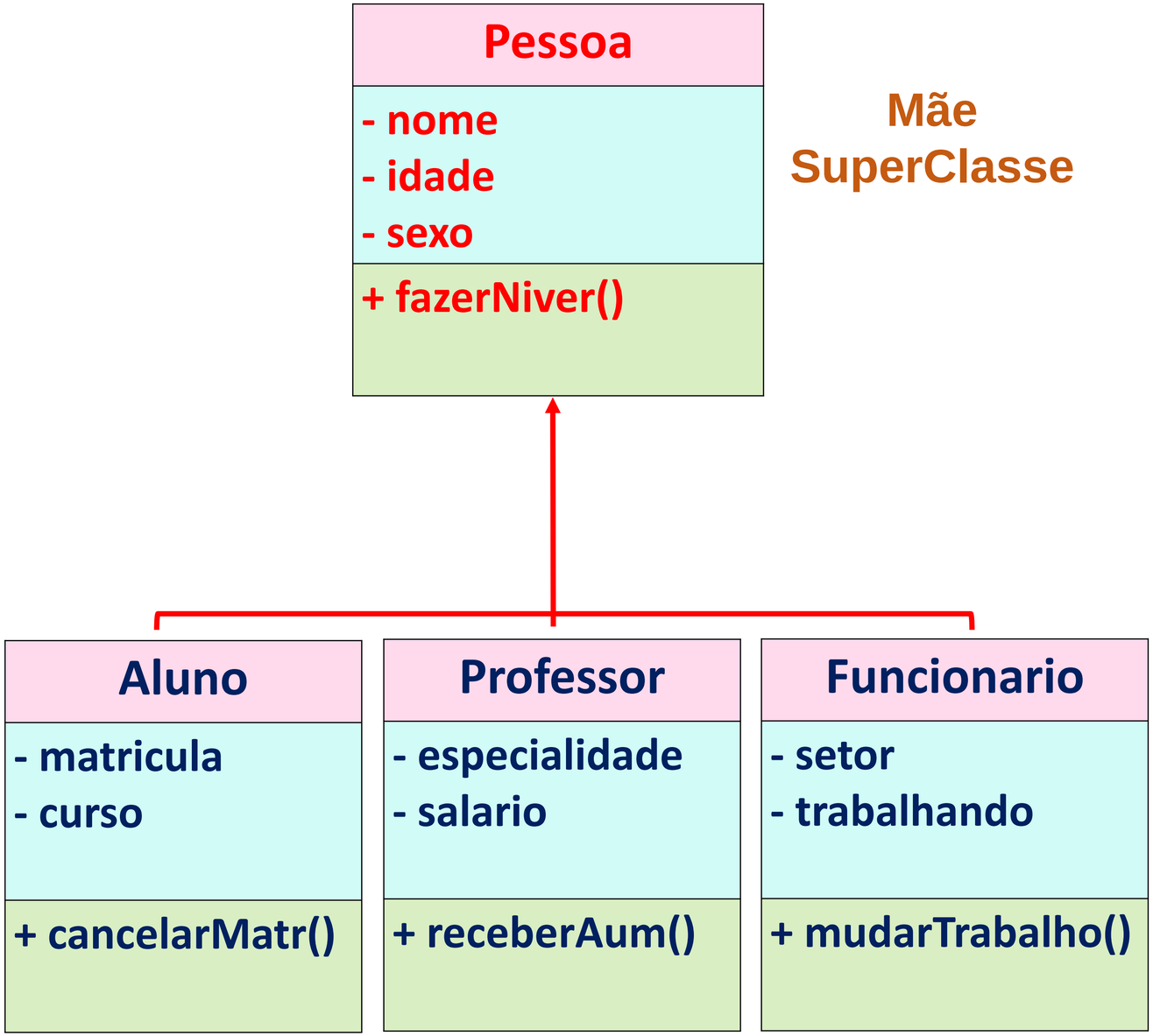
Os atributos e métodos comuns, sugerem que uma **classe mãe** pode ser construída.

As demais classes podem herdar tais características e comportamentos.

Todos **herdam características** de Pessoa.

Herança

O diagrama mostra que as classes Aluno, Professor e Funcionário, **herdam** características e comportamentos da Classe mãe (**Pessoa**).



vnt/school
powered by venturus

Herança

Inicialmente construímos a Classe mãe, **Pessoa**.

Inserimos também os **Gets** e **Sets**.

```
classe Pessoa
//Atributos
privado String nome
privado inteiro idade
privado String sexo
//Metodos
Publico metodo fazerNiver()
    (...)
fimMetodo
//Metodos Especiais
privado Metodo get...()
    (...)
FimMetodo
privado Metodo set...()
    (...)
FimMetodo
...
FimClasse
```

Herança

Criamos a classe **Aluno** com os seus devidos atributos e métodos.

Veja que a classe Aluno se **estende** a classe **Pessoa**.

```
classe Aluno estende Pessoa
//Atributos
privado inteiro matr
privado String curso
//Metodos
Publico metodo cancelarMatr()
    (...)
fimMetodo
//Metodos Especiais
privado Metodo get...()
    (...)
FimMetodo
privado Metodo set...()
    (...)
FimMetodo
...
FimClasse
```

Herança

Criamos a classe **Professor** com os seus devidos atributos e métodos.

Veja que a classe Professor se **estende** a classe **Pessoa**.

```
classe Professor estende Pessoa
//Atributos
privado String especialidade
privado Real salario
//Metodos
Publico metodo receberAum()
    (...)
fimMetodo
//Metodos Especiais
privado Metodo get...()
    (...)
FimMetodo
privado Metodo set...()
    (...)
FimMetodo
...
FimClasse
```

Herança

Criamos a classe **Funcionário** com os seus devidos atributos e métodos.

Veja que a classe Funcionário se **estende** a classe **Pessoa**.

```
classe Funcionario estende Pessoa
//Atributos
privado String setor
privado Logico trabalhando
//Metodos
Publico metodo mudarTrabalho()
    (...)
fimMetodo
//Metodos Especiais
privado Metodo get...()
    (...)
FimMetodo
privado Metodo set...()
    (...)
FimMetodo
...
FimClasse
```

Herança

Após desenvolver todas as Classes e estender as classes filhas à Classe mãe.

O que pode ser feito?

Podemos instanciar os **objetos**.

vnt/school
powered by venturus

Herança

Instanciamos os **objetos**.
Qual o erro?

P1(pessoa) não recebe aumento.

P2 (Aluno) não pode mudar de trabalho.

P4 (Funcionário) não pode cancelar matrícula.

```
//Programa Principal
P1 = new Pessoa( )
P2 = new Aluno( )
P3 = new Professor( )
P4 = new Funcionario( )

P1.setNome("Pedro")
P1.receberAum(450)
P2.setNome("Maria")
P2.mudarTrabalho( )
P3.setSalario(3500)
P4.setSetor("Estoque")
P4.cancerlarMatr( )
```

vnt/school
powered by venturus

Herança

Vamos Praticar!!

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

.

vnt/school
powered by venturus

Herança

```
package aula19;
public class Exemplo1 {
    public static void main(String[] args) {
        // Programa Principal
        Pessoa p1 = new Pessoa();
        Aluno p2 = new Aluno();
        Professor p3 = new Professor();
        Funcionario p4 = new Funcionario();

        p1.setNome("José");
        p2.setNome("Maria");
        p3.setNome("João");
        p4.setNome("Eloá");
```

??

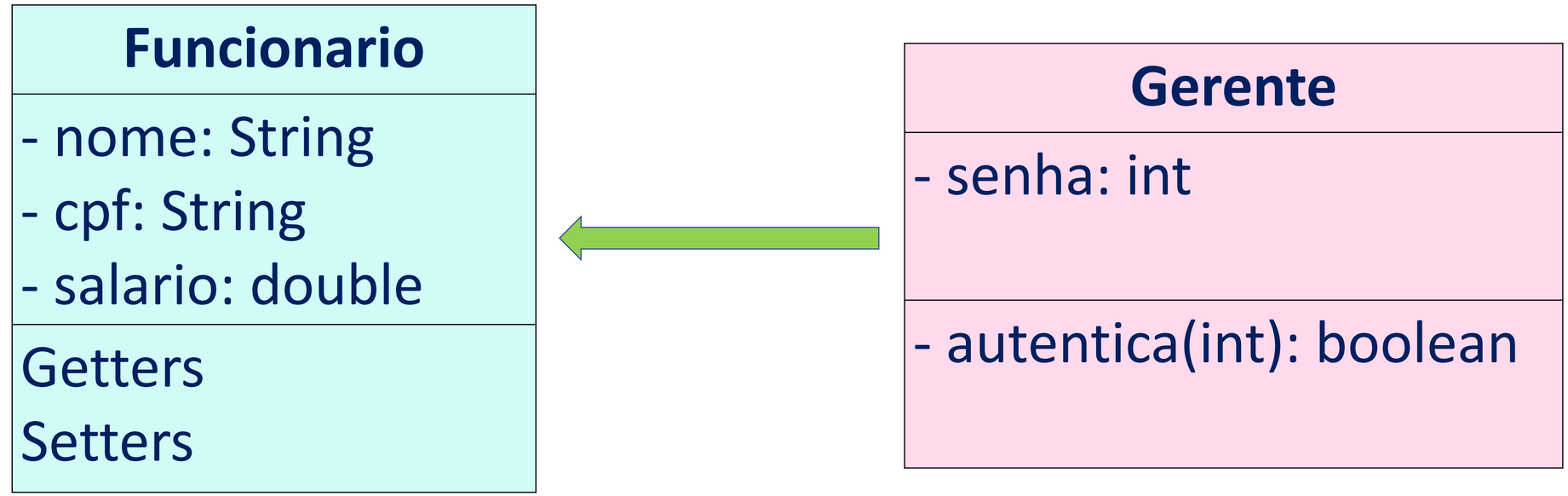
Exercício

Herança

vnt/school
powered by venturus

Atividade 1

Crie uma programa que siga as instruções conforme o diagrama de classe ao lado. Crie uma lógiga para a senha e instancie objetos do tipo funcionário e do tipo Gerente. Veja que a todo momento que for criado um objeto do tipo Gerente, este objeto possuirá também os atributos definidos na classe Funcionario, pois Gerente é um Funcionario!



vnt/school
powered by venturus

Atividade 1

```java
public boolean autentica(int testarSenha){
    if(testarSenha == senha){
        System.out.println("Acesso Permitido!!");
        return true;
    }else{
        System.out.println("Acesso Negado!!");
        return false;
    }
}
```

coffee
time

Árvore Hierárquica

Tipos de Herança

vnt/school
powered by venturus

Árvore Hierárquica

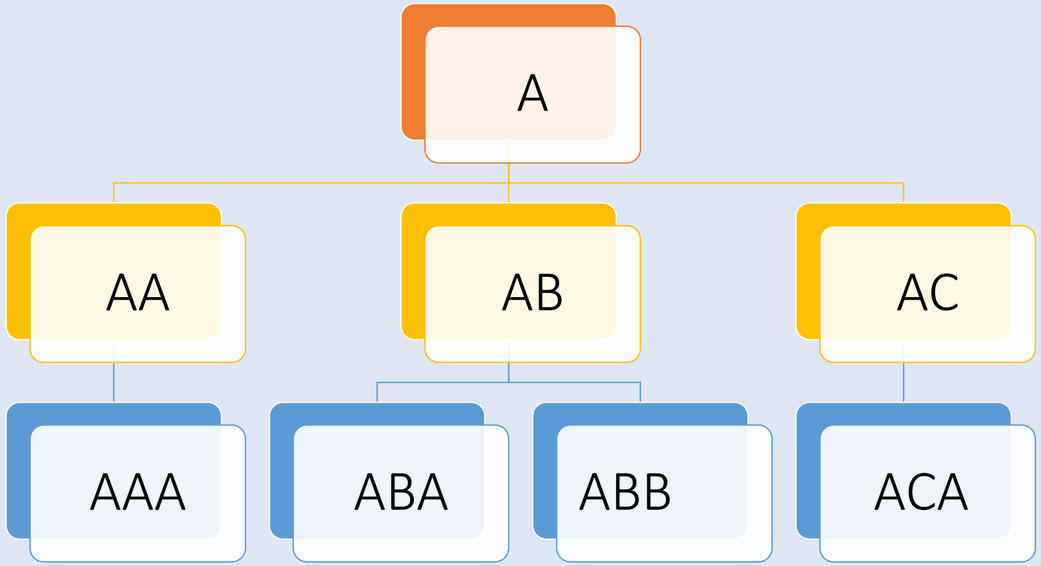
Podemos criar uma árvore de heranças. Lo go, é possível navegar pela herança.

“**A**” -> é uma **superClasse** ou Progenitora.

“**ABA**” -> é uma **subClasse**.

“**AB**” -> é **superClasse** e **subclasse**.

Depende do referencial.



vnt/school
powered by venturus

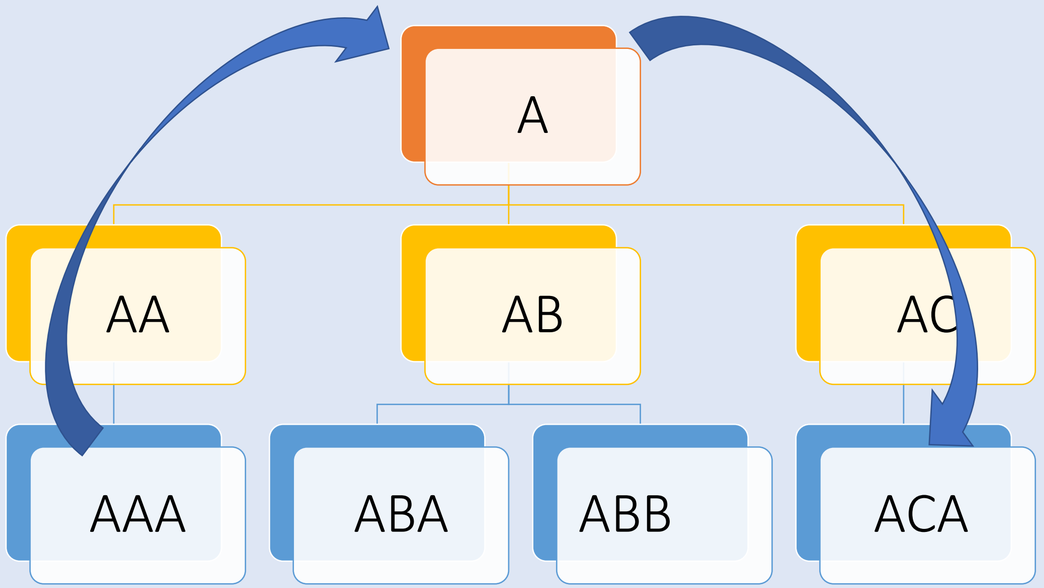
Árvore Hierárquica

Raiz: a Classe origem de tudo.

Folha: a última classe da árvore. Não existe outras classes após ela.

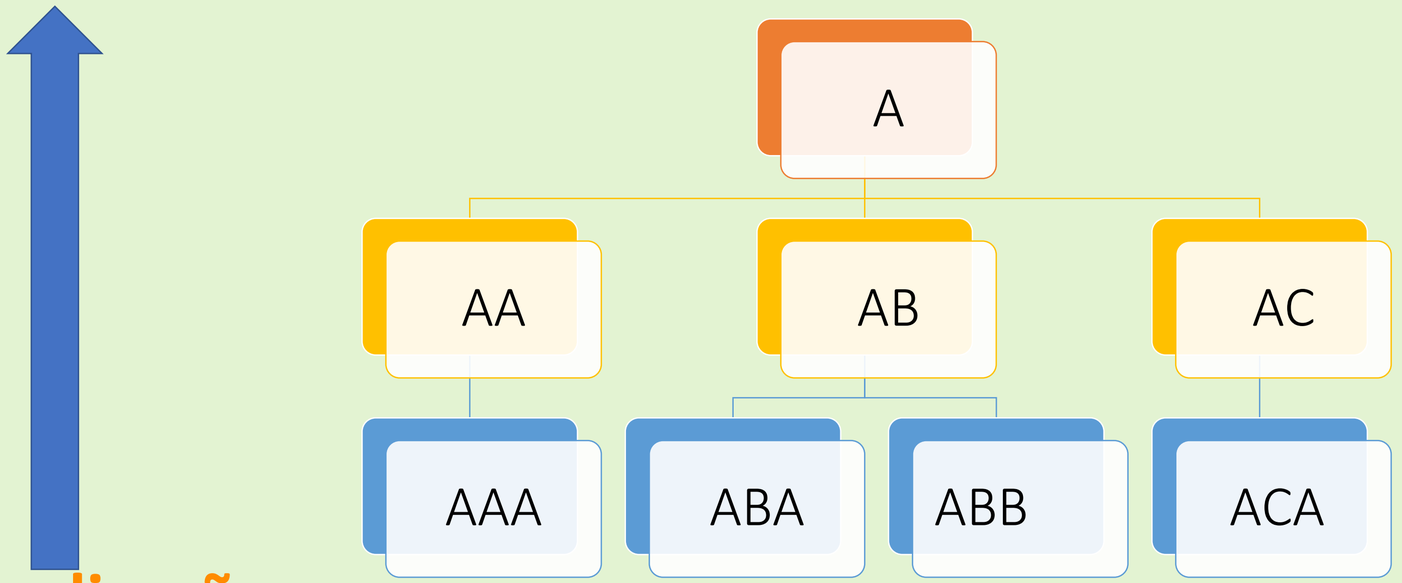
Ancestral: Classes a partir da hierarquia “netos”. Portanto, filhas não são ancestrais.

Descendentes: Classes a partir da hierarquia “netos”. Portanto, filhas não são descendentes.



vnt/school
powered by venturus

Árvore Hierárquica



Generalização

Especialização

vnt/school
powered by venturus

Tipos de Herança

Herança de Implementação.

É o tipo de herança mais simples que existe, podemos também dizer que é a herança mais pobre que existe.

Herança para Diferença.

É o tipo de herança mais completa que existe. Iremos abordar em vários momentos do curso.

OBS: Neste curso definimos dois tipos de heranças. No entanto, isso não é uma solução definitiva. Você pode encontrar em outros cursos/livros, uma divisão de tipos de heranças maior.

Árvore Hierárquica

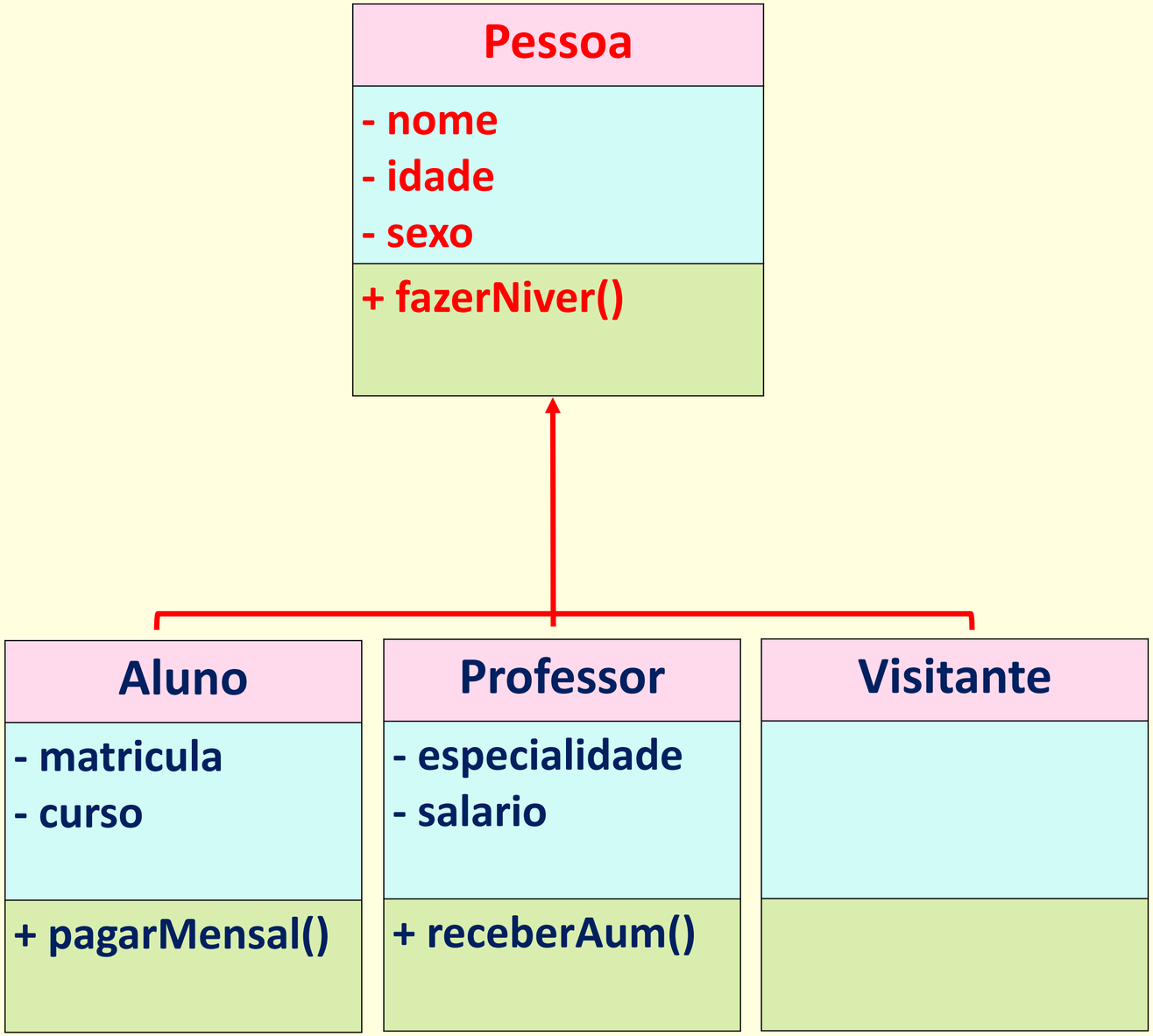
Imagine o seguinte exemplo: um **aluno**, um **professor** e um **visitante**.

Todas essas **pessoas**, podem apresentar **características** e **comportamentos** semelhantes.

Herança

A subclasse “Visitante” é o exemplo de tipo de “herança de implementação”.

Já as demais subclasses “Aluno” e “Professor” são denominadas “Herança para Diferença”.

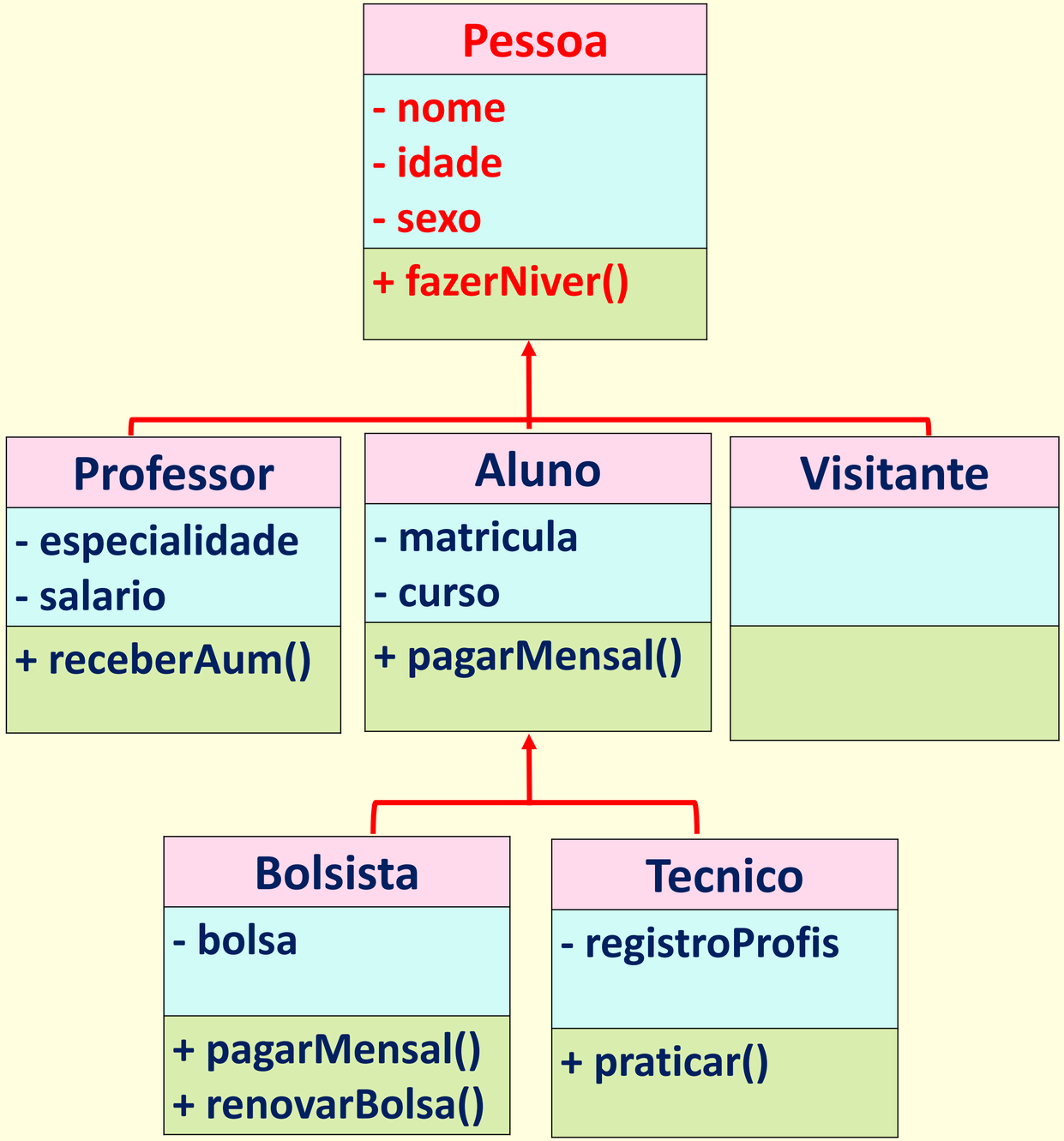


Herança

As classe “**Bolsista**” e “**Tecnico**” **herdam** do “Aluno” e “Pessoa” todos os atributos e métodos.

No “**Bolsista**” é implementado novamente o método “pagarMensalidade” porque ele terá uma ação diferente.

Será **sobrescrito**.



vnt/school
powered by venturus

Árvore Hierárquica

Classe Abstrata: Não pode ser instanciada. Só pode servir como progenitora.

Método Abstrato: Declarado, mas não implementado na progenitora.

Árvore Hierárquica

Classe Final: Não pode ser herdado por outra classe. Obrigatoriamente é uma folha.

Método Final: Não pode ser sobrescrito pelas suas subclasses. Obrigatoriamente herdado.

vnt/school
powered by venturus

Árvore Hierárquica

Pessoa é uma **classe abstrata**.
É uma superClasse.

As classes “**Visitante**” e “**Aluno**” estendem para a classe “**Pessoa**”.

```
classe abstrata Pessoa
privado String nome
privado inteiro idade
privado String sexo
Publico metodo final fazerNiver()
    (...)
fimMetodo
/FimClasse

classe Aluno estende Pessoa
privado inteiro matricula
privado String curso
Publico metodo pagarMensal()
    (...)
fimMetodo
/FimClasse
```

vnt/school
powered by venturus

Árvore Hierárquica

A classe “**Bolsista**” estende para classe “**Aluno**”.

O método “**pagarMensalidade**” é **sobreposto**. @sobrepor

@sobrepor é um tipo de polimorfismo.

```
classe Bolsista estende Aluno
privado inteiro bolsa
Publico metodo renovarBolsa()
    (...)
FimMetodo
@Sobrepor
Publico metodo pagarMensal()
    (...)
fimMetodo
/FimClasse
```

vnt/school
powered by venturus

Árvore Hierárquica

Instanciamos os **objetos**.
Qual o erro?

A classe “**Pessoa**” é **abstrata**.
Portanto, ela **não pode instanciar** um objeto.

A primeira linha do programa principal estaria errado.

```
//Programa Principal
P1 = new Pessoa( )
A1 = new Aluno( )
V1 = new Visitante( )


P1.nome("Pedro")
A1.idade(25)
V1.nome("Maria")
A1.pagarMensal(1200 )
```

vnt/school
powered by venturus

Herança

Vamos Praticar!!

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

.

vnt/school
powered by venturus

Herança

```java
package aula19;
public class Exemplo2 {
    public static void main(String[] args) {
        Aluno a1 = new Aluno();
        a1.setNome("Eri");
        a1.setMatricula(0001);
        a1.setCurso("Data Science");
        a1.setIdade(35);
        a1.setSexo("M");
        a1.pagarMensalidade();
        System.out.println("-------------------------------");
```

??

Exercício

Herança

vnt/school
powered by venturus

Atividade 2

Faça uma classe “Animal” com um método “falar” e acrescente os atributos “Nome” e “Idade”. Faça as classes Homem, Cão, Gato e Papagaio, herdando de animal, redefinindo o método “falar” para retornar “Oi”, “Au au”, “Miau” e "Qué Café", respectivamente. Acrescente o atributo fome e o método comer(), caso o animal esteja com fome dê a comida adequada. Caso contrário, diga que não está com fome. Instancie todos os objetos possíveis. Ao instanciar apresente o objeto e o método “Falar” e "Comer".

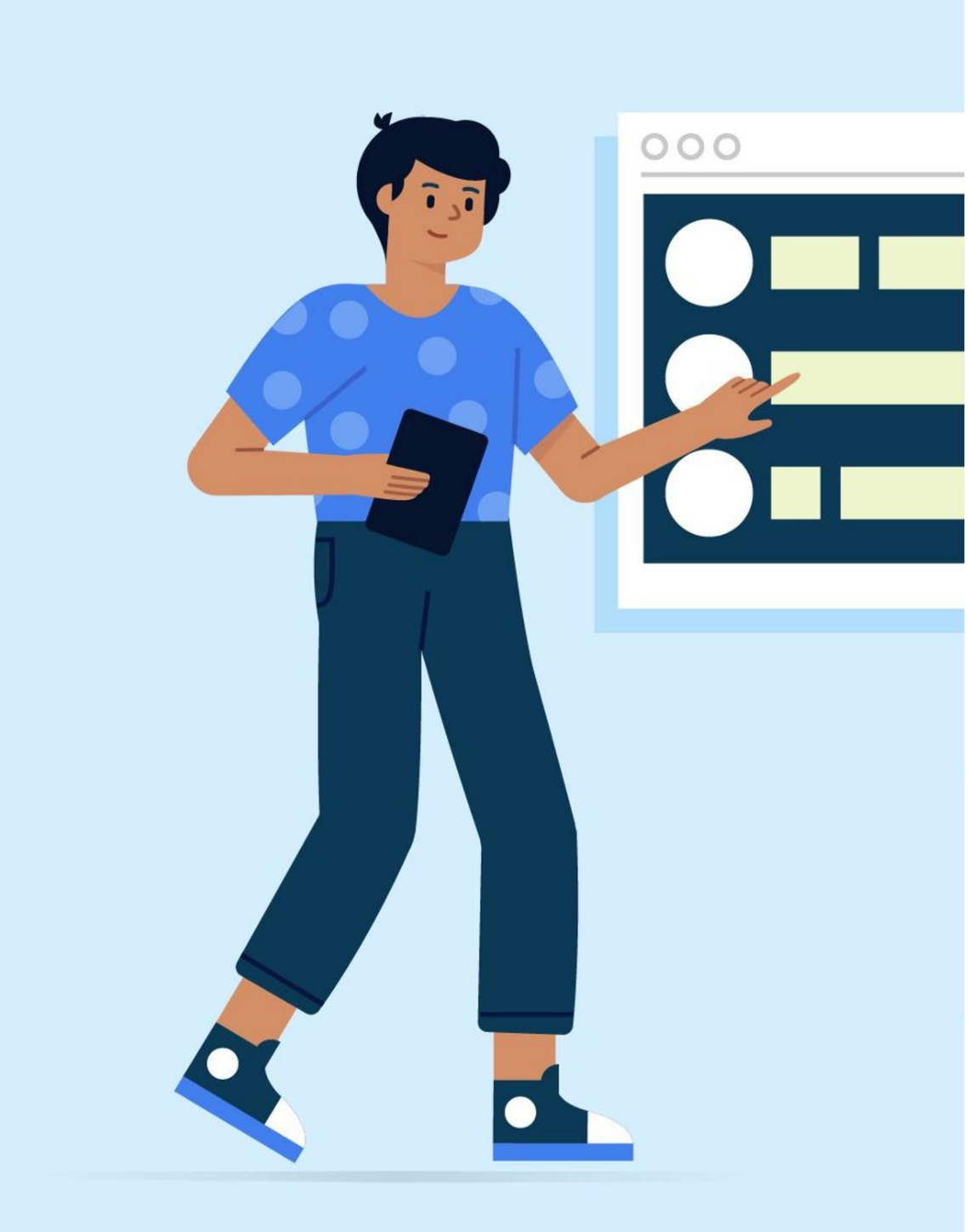
vnt/school
powered by venturus

Atividade 2

```
package aula19;
public class Atividade2 {
    public static void main(String[] args) {
        Gato g1 = new Gato();
         g1.setNome("Prince");
         g1.setIdade(3);
         g1.fome = true;
         g1.falar();
         g1.comer();
         System.out.println(g1.toString());
         System.out.println("##################################");
         System.out.println(" ");
```

??


Review
e
Preview



Comunidade VNT

Dica de hoje

O link abaixo apresenta uma boa leitura sobre o conceito de Herança. O texto aborda de forma prática alguns exemplos. Aproveite pra conhecer um pouco mais.

https://medium.com/caiquefortunato/heran%C3%A7a-em-java-teoria-e-pr%C3%A1tica-2ca7d9b0f3de

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/