UNIVERSIDADE FEDERAL DO CEARÁ

TÓPICO 09

PROGRAMAÇÃO ORIENTADA A OBJETOS

Prof. Marcos Vinicius de Andrade Lima
E-mail: marcos.vinicius@ufc.br

Olá!

Sou Marcos Vinicius

No tópico passado nós aprendemos mais coisas sobre os segredos da agregação com Java...

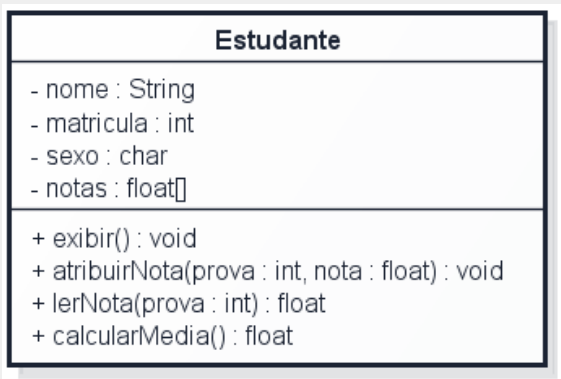
Neste tópico aprenderemos como trabalhar com o mecanismo de **Herança**!

> **Mesmo desacreditado por todos, não posso desistir, pois para mim, vencer é nunca desistir** (Albert Einstein)

Herança na OO

INTRODUÇÃO

- Imagine que você precise criar uma classe para representar estudantes. Essa classe terá algumas propriedades e ações que são comuns a todos os estudantes, como por exemplo:

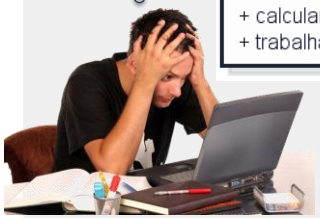


INTRODUÇÃO

- Agora, como faríamos para **criar outros tipos de estudantes**, semelhantes ao estudante que definimos anteriormente, mas com algumas poucas diferenças?
- Poderíamos imaginar, por exemplo, **estudantes tutores** (que realizam tutoria na universidade), e **estudantes estagiários** (que desenvolvem estágio em alguma empresa).
- **Esses novos tipos de estudante** possuem as mesmas propriedades e realizam as mesmas operações que o estudante comum, porém, **possuem novas propriedades e realizam novas operações**.

Olha que Lindo... Novas Classes!

Qual é a pegadinha ?



EstudanteEstagiario

- nome : String
- matricula : int
- sexo : char
- notas : float[]
- empresa : String
- bolsaSalario : float

+ exibir() : void
+ atribuirNota(prova : int, nota : float) : void
+ lerNota(prova : int) : float
+ calcularMedia() : float
+ trabalhar() : void

EstudanteTutor

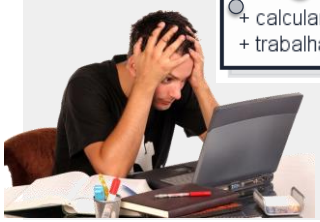
- nome : String
- matricula : int
- sexo : char
- notas : float[]
- disciplina : String
- bolsa : float

+ exibir() : void
+ atribuirNota(prova : int, nota : float) : void
+ lerNota(prova : int) : float
+ calcularMedia() : float
+ tirarDuvidas() : void
+ auxiliarProfessor() : void

Novas Classes!

Tenho que definir **todos** os **atributos** e **métodos** comuns em cada uma das classes...

Já sei! Vou usar a salvação "**Copy & Paste**"... Yeah! **Eu sou demais!!!**

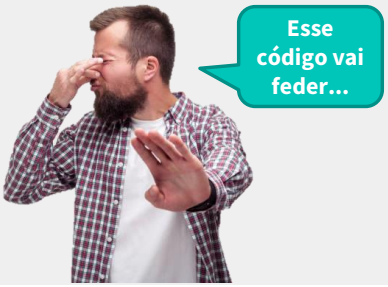


EstudanteTutor

+ exibir() : void
+ atribuirNota(prova : int, nota : float) : void
+ lerNota(prova : int) : float
+ calcularMedia() : float
+ tirarDuvidas() : void
+ auxiliarProfessor() : void

Houston, There is a Problem!

- Em uma **programação tradicional**, teríamos que definir diversas vezes os métodos que são comuns as classes **Estudante**, **EstudanteTutor** e **EstudanteEstagiário**.
- O reaproveitamento de código nesse caso se basearia no **Copy & Paste.**



Qual o Problema de Copy & Past em Classes?

- Quando algum desses métodos precisar ser alterado, todas as classes correspondentes precisarão ser alteradas.

Manutenção ➔ Difícil!

Herança de Classe

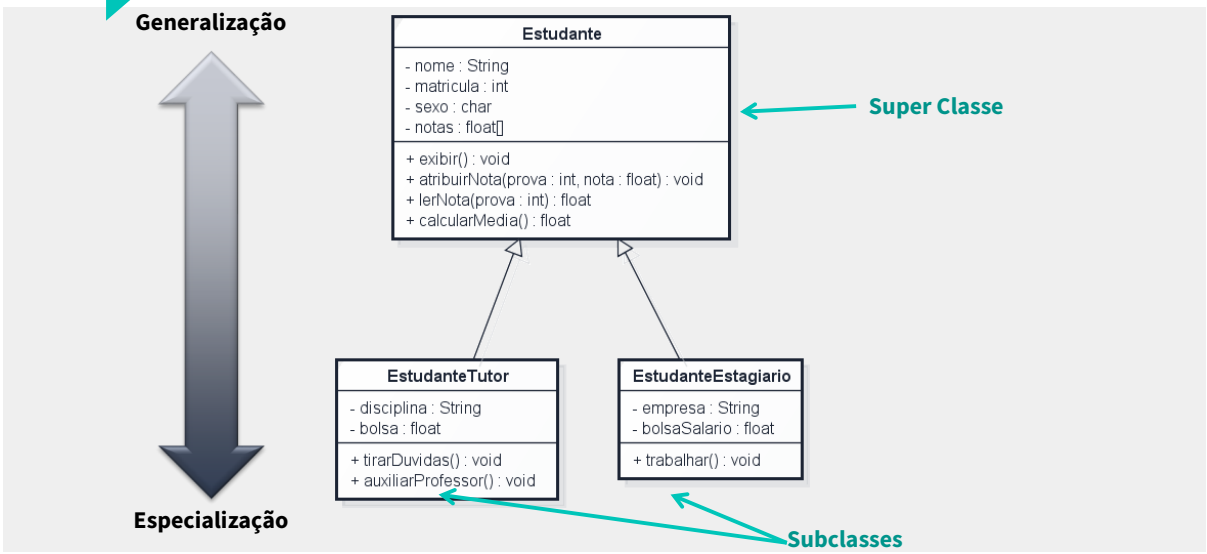
- **Herança de Classe** é usada na POO para definir novas classes a partir de uma classe existente.
- A nova classe herda **todos** os membros (atributos e métodos) da classe antiga sem precisar escrever o código novamente.
- **A nova classe é chamada de:**
  subclasse, classe derivada, classe filha, etc.
- **A classe antiga é chamada de:**
  superclasse, classe base, classe mãe, classe pai, etc.

A **subclasse** pode (e deve) **acrescentar novo comportamento** (métodos) e/ou **propriedades** (variáveis de instância), e, em algumas situações, **até mesmo mudar o comportamento herdado**.

HERANÇA DE CLASSE

- Herança é utilizada caracteristicamente no processo de **especialização** de funcionalidades mais gerais para situações mais específicas, evitando que se tenha que iniciar a implementação do zero.
- A **generalização** segue o processo inverso, isto é, parte de situações mais específicas para mais gerais.

Representação da Herança na UML... Lembra?

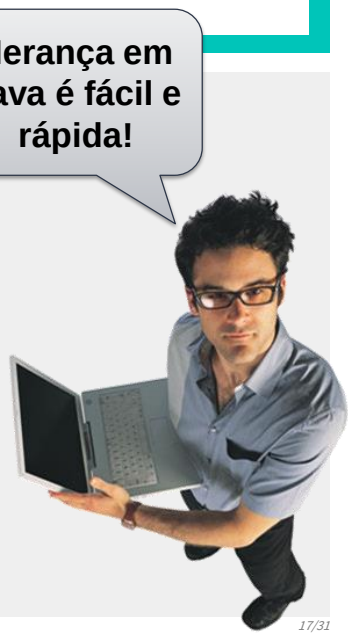


Além da herança, existe outro mecanismo fundamental de reuso de código, chamado **agregação**.

Enquanto a **herança define uma relação** do tipo "**é um**" (ou relação superclasse-subclasse), a **agregação define uma relação** do tipo "**tem um**" (ou relação todo-parte).

Aplicando Herança em Java

- A superclasse é especificada na cláusula **`extends`** no cabeçalho da subclasse.
- Se esta não for especificada, a classe herdará implicitamente da classe **`java.lang.Object`**, que é a classe mãe de todas as classes.
- **A subclasse especifica apenas membros novos ou modificados**, sendo o resto da definição herdada da superclasse.



Relembrando a Classe Estudante

```
public class Estudante{
  // variaveis de instancia
  private int matricula;
  private String nome;
  private char sexo;
  private float notas[] = new float[4];

// metodos de instancia
  public void exibir() { ... }
  public void atribuirNota(int prova, float nota)
  { ... }
  public float lerNota(int prova) { ... }
  public float calcularMedia() { ... }
}
```

USANDO A HERANÇA COM *EXTENDS*

Para estender da classe Estudante, cria-se um novo arquivo chamado **EstudanteTutor.java** com o seguinte código:

```
public class EstudanteTutor extends Estudante{
  // novas variáveis de instância
  private double bolsa;
  private String disciplina;

  // novos métodos de instância
  public void tirarDuvidas() { ... }
  public void auxiliarProfessor() { ... }
}
```

USANDO A HERANÇA COM *EXTENDS*

Para representar o estagiário, cria-se um novo arquivo chamado **EstudanteEstagiario.java** com o seguinte código:

```
class EstudanteEstagiario extends Estudante{
  // novas variáveis de instância
  private double salario;
  private String empresa;

  // novo método de instância
  public void trabalhar() { ... }
```

Vantagens do Uso da Herança

Permite reuso de código

Facilita a manutenção

Melhora o entendimento

Acaba com o "Copy & Paste"

Recordar é Viver! Se Liga!

- Existem dois tipos de **Herança de Classes** na POO:

  Herança Simples ➔ uma subclasse só pode ter uma superclasse imediata;

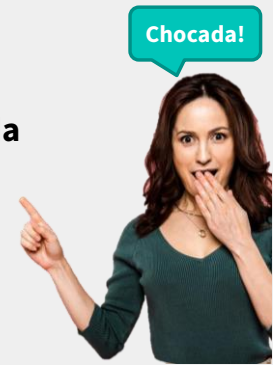
  A herança simples de classes também é chamada de herança linear de implementação ou herança simples de implementação.

  Herança Múltipla ➔ uma classe pode ter várias superclasses imediata.

Em Java, uma classe só pode herdar (estender) de uma única classe, ou seja, uma classe só pode ter uma única superclasse imediata!

No Mecanismo de Herança...

- Os atributos e métodos privados são herdados, **mas não podem ser acessados diretamente.** Deve-se usar métodos acessadores (*getters*) e modificadores (*setters*).
- **Uma instância de uma subclasse é uma instância de todas as suas superclasses.**
- **Objetos da subclasse comportam-se como objetos da superclasse.**
- **Objetos da subclasse podem ser usados no lugar de objetos da superclasse**.



Mecanismo da Herança em Ação

Uma instância de uma subclasse é uma instância de todas as suas superclasses.

```
...
EstudanteTutor et = new EstudanteTutor();
Estudante e = et;
Estudante e2 = new EstudanteTutor();
...
```


Mecanismo da Herança em Ação

Objetos da subclasse comportam-se como objetos da superclasse.

```
...
EstudanteTutor et = new EstudanteTutor();
et.auxiliarProfessor();
et.atribuirNota(1, 7.5f);
et.lerNota(1);
...
```

operações herdadas da superclasse (comportamento da superclasse)

Mecanismo da Herança em Ação

Objetos da subclasse podem ser usados no lugar de objetos da superclasse.

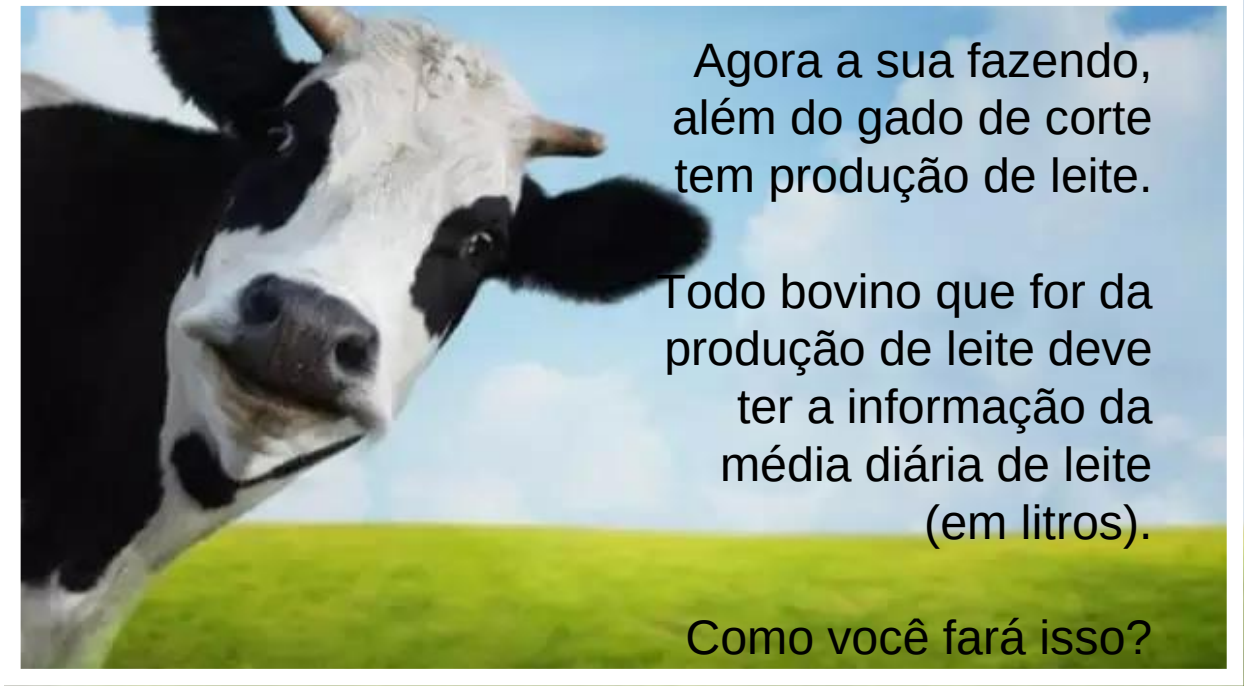
```
class Turma {
  String nome;
  Estudante estudantes []= new Estudante[5];
  private int numEstudantes = 0;
  public void matricular( Estudante e ){
    estudantes[numEstudantes] = e;
    numEstudantes++;
  } }
```


Mecanismo da Herança em Ação

Objetos da subclasse podem ser usados no lugar de objetos da superclasse.

```
class Programa{
public static void main(String args[]){
  EstudanteTutor et = new
EstudanteTutor("Pedro");
  Turma t = new Turma("POO");
  t.matricular( et );
}}
```


Vamos brincar um pouco?


Agora a sua fazendo, além do gado de corte tem produção de leite.
Todo bovino que for da produção de leite deve ter a informação da média diária de leite (em litros).
Como você fará isso?

Obrigado!

Mais alguma dúvida?

Acesse o **AME** para mais informações e treinamento do **NERDS**!

http://ame2.russas.ufc.br