ECM251 - Linguagens I

Herança

Prof. Murilo Zanini de Carvalho
Prof. Tiago Sanches da Silva

Antes de começar!

Clone seu repositório do Github

- Lembre-se sempre antes de iniciar uma aula, clonar seu repositório remoto e realizar as atividades nele.
- Para cada atividade desenvolvida, criar um novo diretório.

Retirado de (https://miro.medium.com/max/4000/0*MZMl76wKo2FQLqG0.png), em 07/03/2021

Herança

Analisar é viver!

Retirado de (https://vignette.wikia.nocookie.net/liberproeliis/images/5/5b/Endeavor_-_Uniforme.png/revision/latest?cb=20180618013320&path-prefix=pt-br), em 12/04/2020

+

Retirado de (https://vignette.wikia.nocookie.net/bokunoheroacademia/images/e/ed/Todoroki_Rei_Anime.png/revision/latest?cb=20180615145703&path-prefix=pt-br), em 12/04/2020

=

Retirado de (https://vignette.wikia.nocookie.net/myheroacademianovaera/images/4/47/Shoto_Todoroki_Hero_Costume_Profile.png/revision/latest/top-crop/width/360/height/450?cb=20190717174923&path-prefix=pt-br), em 12/04/2020

Analisar é viver!

Retirado de (https://conteudo.imguol.com.br/c/entretenimento/16/2017/06/27/naruto-1498593686428_v2_1920x1080.png), em 12/04/2020

+

Retirado de (https://criticalhits.com.br/wp-content/uploads/2019/03/hinata-byakugan.jpg), em 12/04/2020

=

Retirado de (https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS6JVs-8n2ER0h6s5emWLkLQCIygSiFQ68cYiMat5yi4eD4YqFC5A&s), em 12/04/2020

Analisar é viver!

Como uma primeira aproximação, podemos dizer que a herança é uma forma dos pais passarem características para os filhos.

Retirado de (https://img.memecdn.com/rmx-yamcha-and-gohan-anyone-else-realize_o_361927.jpg), em 12/04/2020

Um problema

Um problema

Existem algumas categorias para classificação de algumas profissões, como mecânicos, engenheiros, desenvolvedores, por exemplo. Vamos estudar a classificação dos NINJAS para avaliarmos o 3 e 4 pilares da orientação a objetos.

A classificação leva em consideração os níveis que os ninjas podem atingir durante sua carreira.

Retirado de (https://lh3.googleusercontent.com/proxy/8vVkWN2M_XPkX4KMJpapOMULYZDMmGebuf[_i4ZmsCzMiAg_Lilfc8M6BpaD699S86YAMMNbK0SIjsqWWgBXTLgsGYoN698JkUi7M3ar6OEAFLiybuGi2XpNAXOyPcGK7kUqVCQD3h27e_WpocdrIwopBskpw), em 12/04/2020

Um problema

Modelar as classes para representar cada uma das categorias dos ninjas. Ainda ficar no modelo da classe, não realizar as implementações.

AcademicStudent	Genin	Chuunin	Jounin
name : String family : String jutsus : String []	name : String family : String jutsus : String [] mission : String	name : String family : String jutsus : String [] mission : String	name : String family : String jutsus : String [] mission : String
train() play()	train() goToMission()	train() goToMission()	train() goToMission()

Um problema

Retirado de (https://static2.cbrimages.com/wordpress/wp-content/uploads/2019/07/Naruto-meme-header.jpg), em 12/04/2020

Analisando a proposta das classes, é possível perceber que bastante do código entre elas é semelhante, em muitos casos igual.

Será que não existe nenhuma forma melhor de fazer essa implementação?

AcademicStudent	Genin	Chuunin	Jounin
name : String family : String jutsus : String []	name : String family : String jutsus : String [] mission : String	name : String family : String jutsus : String [] mission : String	name : String family : String jutsus : String [] mission : String
train() play()	train() goToMission()	train() goToMission()	train() goToMission()

Entra a Herança!

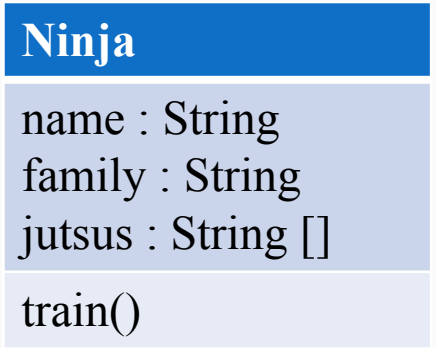
Herança

Existe um jeito, em Java, de relacionarmos uma classe de tal maneira que uma delas **herda** tudo que a outra tem. Isto é uma **relação** de **classe mãe** e **classe filha**.

No Java fazemos isso com a palavra chave **extends**.

Herança

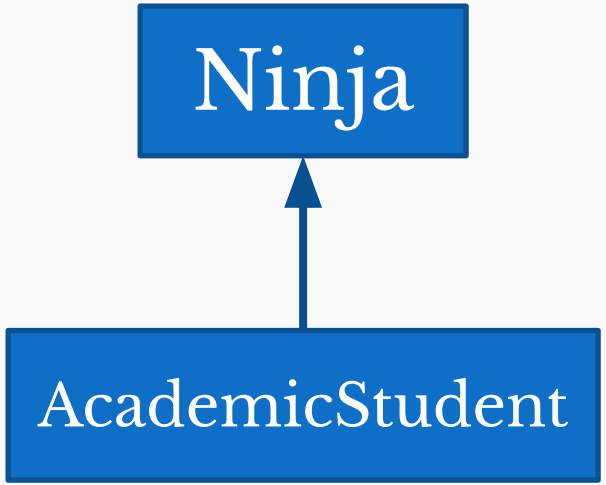
Para a nossa aplicação, podemos agrupar todas as propriedades comuns entre os projetos das classes e propor uma classe base. Assim nossas outras classes seriam uma EXTENSÃO da classe base.



Herança

Em todo momento que criarmos um objeto do tipo AcademicStudent, este objeto possuirá também os atributos definidos na classe Ninja, pois um AcademicStudent **é um** Ninja.

Dizemos que a classe AcademicStudent **herda** todos os atributos e métodos da classe mãe, no nosso caso, a Ninja.



Herança

Super e Sub Classe

- A nomenclatura mais encontrada é que Ninja é a **superclasse** de AcademicStudet, e AcademicStudent é a **subclasse** de Ninja.
- Dizemos também que todo AcademicStudent **é um** Ninja.
- Outra forma é dizer que Ninja é a classe Mãe de AcademicStudent e AcademicStudent é a classe filha de Ninja.

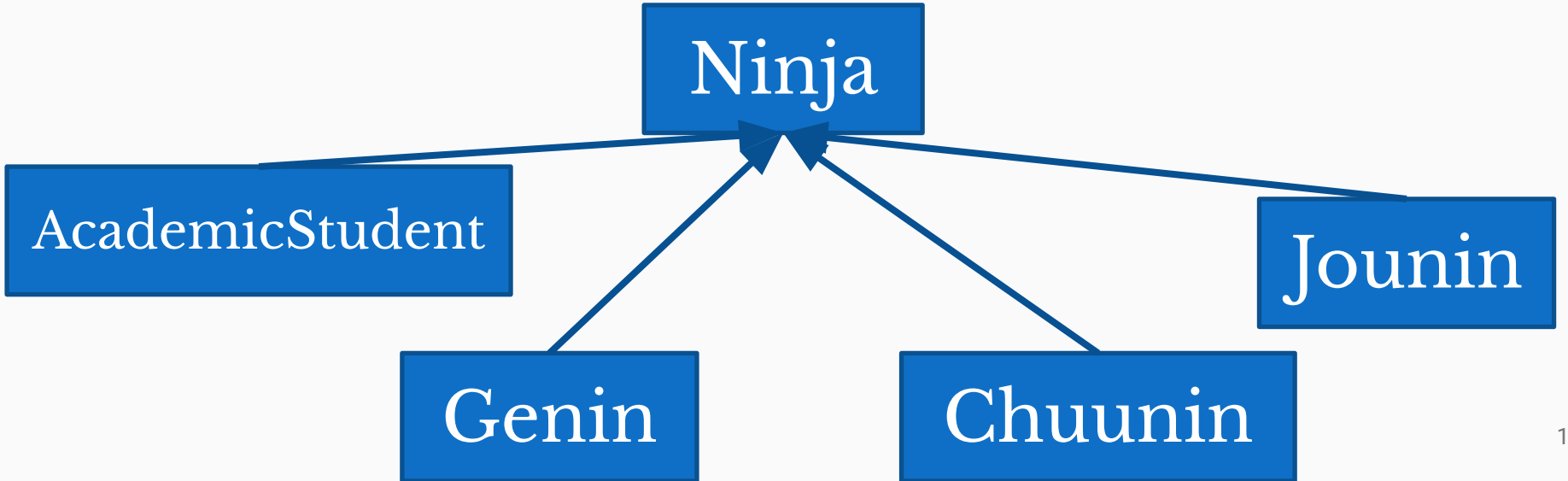
Herança

A forma como a herança vai ser implementada também depende do conjunto de regras que estão sendo implementadas. Deve ficar claro que pode que essa é uma decisão de negócio.

Se por exemplo, um Diretor vai estender de Gerente ou não, vai depender se para você, Diretor é um tipo de Gerente.

Herança

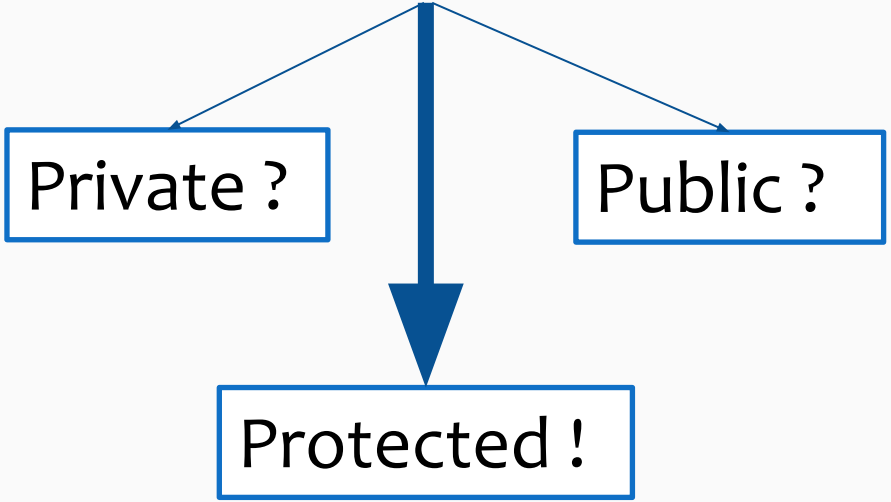
Uma classe pode ter várias filhas, mas pode ter apenas uma mãe, é a chamada herança simples do Java.



Herança e Modificadores de Acesso

Herança e Modificadores de Acesso

A sub classe também herda os atributos e métodos privados, porém não consegue acessá-los diretamente. E se precisamos acessar os atributos que herdamos?



Herança e Modificadores de Acesso

Um atributo protected só pode ser acessado (visível) pela própria classe e por suas subclasses.

Protected !

```
public class Funcionario {
    protected String nome;
    protected String cpf;
    protected double salario;
    // métodos devem vir aqui
}
```

Formalizando a Herança

Formalizando a Herança

Herança é um mecanismo que permite que características comuns a diversas classes sejam fatoradas em uma classe base, ou superclasse. A partir de uma classe base, outras classes podem ser especificadas.

Cada classe derivada ou subclasse apresenta as características (estrutura e métodos) da superclasse e acrescenta a elas o que for definido de particularidade para ela.

Formalizando a Herança

Herança é a capacidade de reutilizar código pela especialização de soluções genéricas já existentes.

```
// SuperClass.java
public class SuperClass {
...
}

// SubClass.java
public class SubClass extends SuperClass {
...
}
```

Formalizando a Herança

1. **Extensão**: subclasse estende a superclasse, acrescentando novos membros (atributos e/ou métodos). A superclasse permanece inalterada, motivo pelo qual este tipo de relacionamento é normalmente referenciado como herança estrita.

2. **Especificação**: a superclasse específica o que uma subclasse deve oferecer, mas não implementa nenhuma funcionalidade.

3. **Combinação de extensão e especificação**: a subclasse herda a interface e uma implementação padrão de (pelo menos alguns de) métodos da superclasse.

Hierarquia de Classes

Hierarquia de Classes

Superclasse direta:

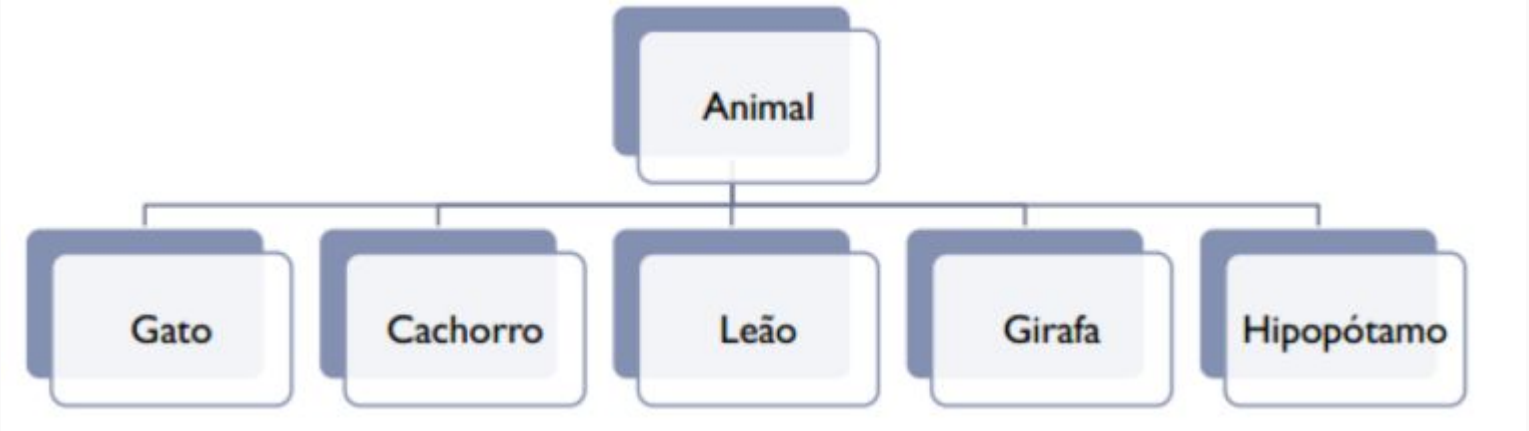
- Herdada explicitamente (um nível acima da hierarquia).

Superclasse indireta:

- Herdada de dois ou mais níveis acima da hierarquia.

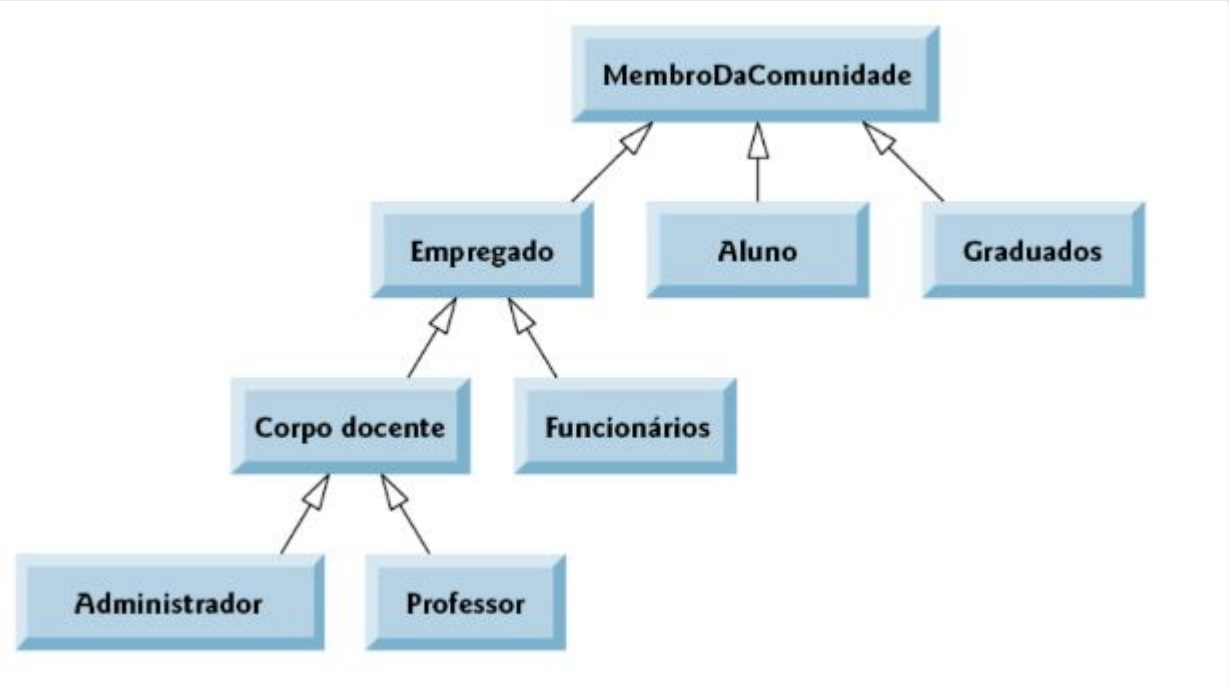
Hierarquia de Classes

Genérico e Específico?



Hierarquia de Classes

Genérico e Específico?



Construtores com Herança

Construtores com Herança

O construtor da superclasse, caso exista, sempre irá ser executado antes do construtor da subclasse. Pois podem existir variáveis da superclasse que precisam de inicialização.

Construtores com Herança

E quando o construtor da superclasse espera alguns parâmetros? Utilize super!

```
public class SuperClasse {

    private int atributo;

    SuperClasse ( int parametro ) {
        this.atributo = parametro;
    }
}
```

```
public class SubClasse extends SuperClasse {


    SubClasse ( int parametroParaSuper ) {
        super(parametroParaSuper);

        // Continuação do construtor da sub classe
    }
}
```

Sobreescrita

Sobreescrita

Um dos mecanismos fundamentais na programação orientada a objetos é o conceito de sobrescrita (no termo em inglês, overriding) de métodos em classes derivadas.

A redefinição ocorre quando um método cuja assinatura já tenha sido especificada recebe uma nova definição (ou seja, um novo corpo) em uma classe derivada.

O mecanismo de sobrescrita, juntamente com o conceito de ligação dinâmica, é a chave para a utilização do **polimorfismo**.

Aguarde as próximas aulas! ☺️

Perguntas?

Retirado de (https://cdn-icons-png.flaticon.com/512/1268/1268705.png), em 02/03/2022