universidade de aveiro
theoria poiesis praxis

esan

Herança

Prof. Dr. Tiago Araújo

tiagodavi70@ua.pt

esan

Material

SCC0504 - Programação Orientada a Objetos (2018)

Luiz Eduardo Virgilio da Silva

Herança

No mundo real, por meio da Genética, é possível herdarmos certas características de nossos ancestrais

- Atributos: cor dos olhos, cor da pele, doenças, etc.
- Comportamentos?

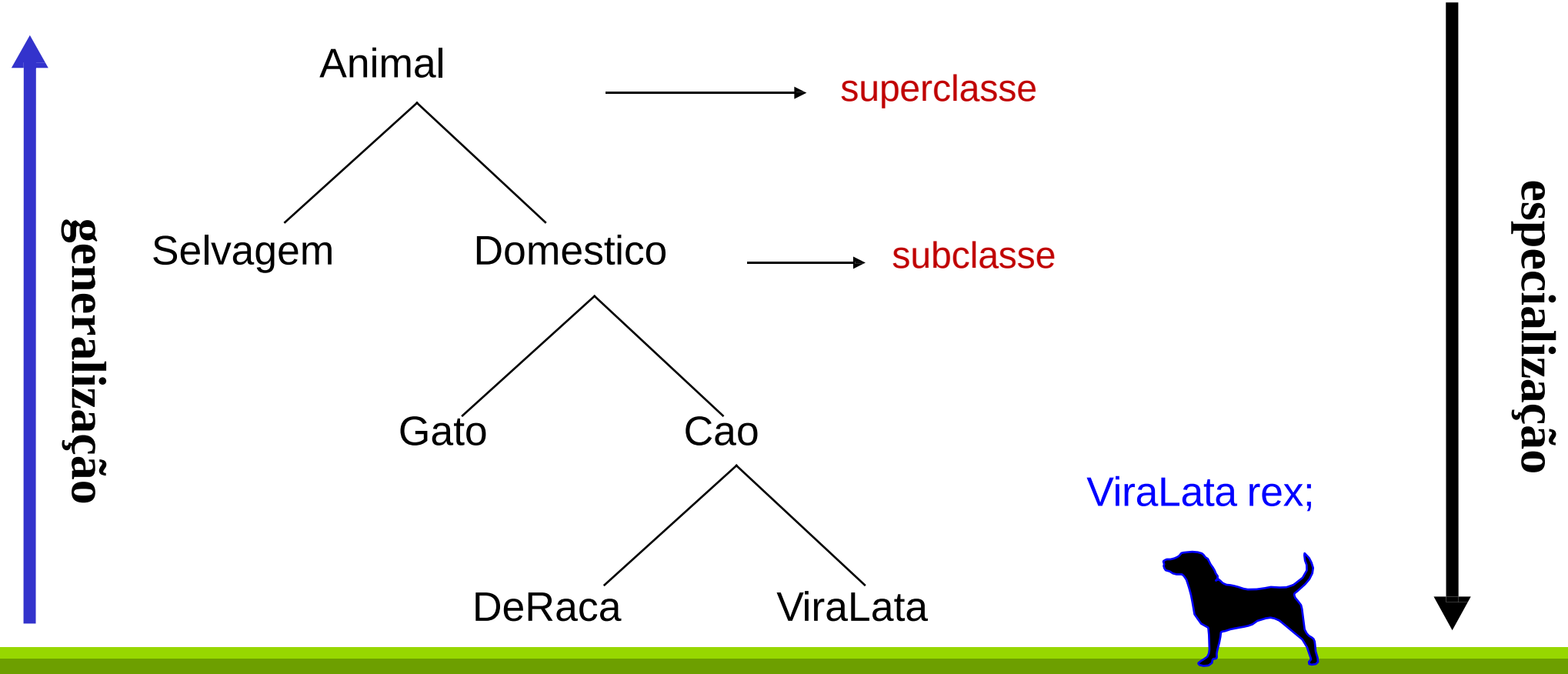
De forma similar, em POO as classes podem herdar

- Atributos (propriedades)
- Métodos (comportamento)

Chamamos este processo de herança

Herança

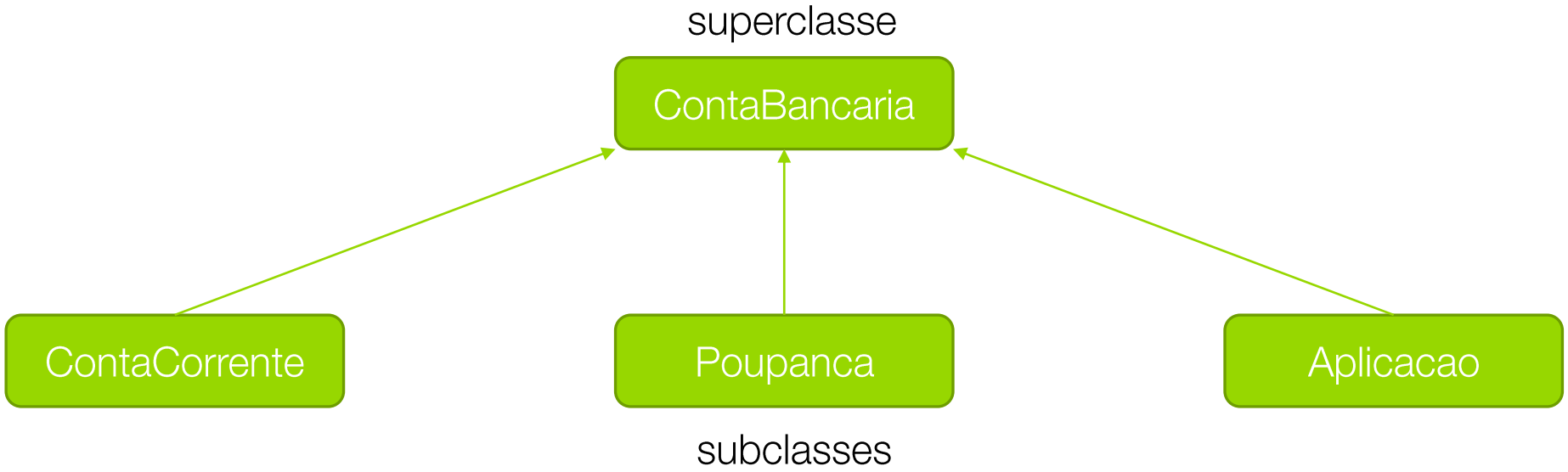
A herança pode ser repetida em cascata, criando várias gerações de classes



Herança

Classe mãe, superclasse, classe base: A classe mais geral, a partir da qual outras classes herdam membros (atributos e métodos)

Classe filha, subclasse, classe derivada: A classe mais especializada, que herda os membros de uma classe mãe



Herança

Herança permite a criação de classes com base em uma classe já existente

- Proporcionar o reuso de software
- Não é preciso escrever (e degubar) novamente
- Especialização de soluções genéricas já existentes

A ideia da herança é “ampliar” a funcionalidade de uma classe

Todo objeto da subclasse também é um objeto da superclasse, mas NÃO o contrário

Encapsulamento

Subclasse herda todos os membros da superclasse

Construtores não são membros da classe
- Contudo, da subclasse é possível chamar um construtor da superclasse

Membros privados (-): ocultos na subclasse
- Acessíveis apenas por métodos

Membros protected (#): acessíveis na subclasse (e outras classes do mesmo pacote)

Membros internal: acessíveis se a subclasse estiver no mesmo pacote da superclasse

Membros public (+): acessíveis na subclasse (e por qualquer outra classe)

Os membros herdados visíveis podem ser usados diretamente, como os membros da própria classe

Encapsulamento

É possível declarar um campo na subclasse com o mesmo nome de um campo da superclasse

- Mesmo que os tipos sejam diferentes
- Ocultamento de campo (não recomendado)

É possível sobrescrever um método da superclasse, declarando um método com a mesma assinatura

- Polimorfismo

É possível declarar novos campos e métodos na subclasse

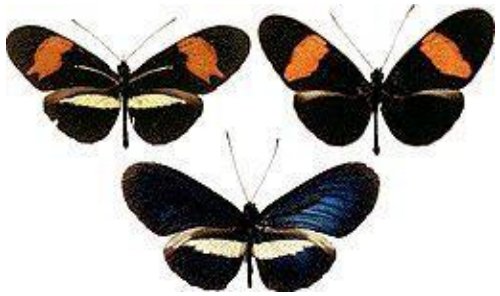
- Especialização

Polimorfismo

Polimorfismo é um conceito que vem da biologia

É a capacidade de indivíduos ou organismos de uma mesma espécie apresentarem diferentes formas

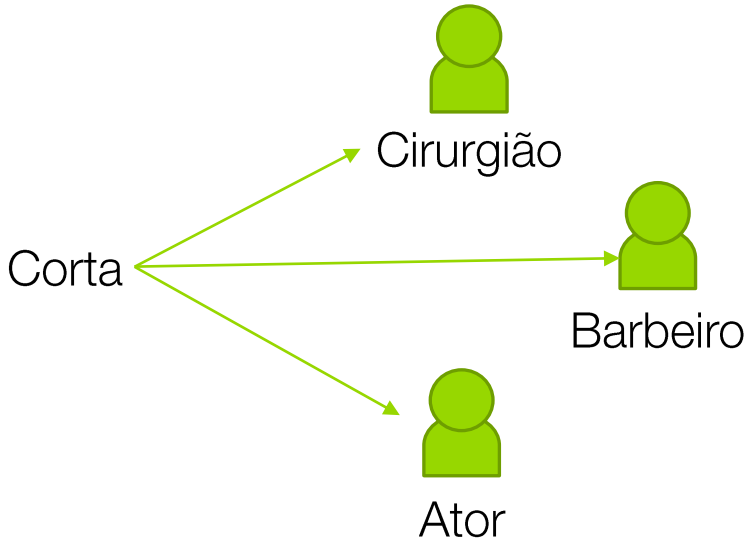
- Instâncias

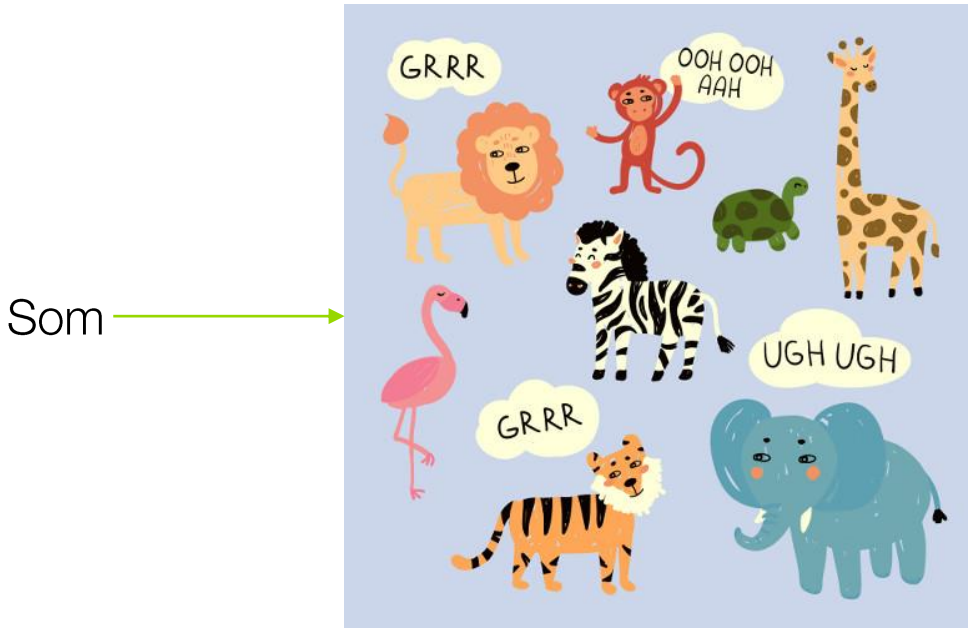


Polimorfismo

Estendendo para POO

- Capacidade de objetos responderem a um MESMA MENSAGEM de maneira diferente
- Mensagem -> chamada de método
- Obtido através da sobreposição (reescrita) de métodos





Polimorfismo

Quando um método de um objeto é chamado, a JVM procura a implementação mais especializada

- Hierarquicamente, de baixo (especializado) para cima (geral)

Se o método não foi definido na classe derivada (subclasse), procura-se pela implementação da classe base (superclasse)

Quando o método é sobrescrito na subclasse, ele passa a ser o comportamento padrão daquela classe

- Mas ainda é possível acessar o método da superclasse

Polimorfismo

A sobrescrita de métodos acontece quando um método da superclasse é redefinido na subclasse

- Mesma assinatura
- Mesmo tipo (ou subtipo) de retorno

Se quisermos aproveitar o comportamento definido pela superclasse, podemos chamar a implementação da superclasse

- Palavra-chave super
- Método da superclasse fica sobreposto (overriding)

Polimorfismo

A JVM chama o método correto de cada objeto, mesmo que eles estejam referenciado sob um tipo mais geral

Palavra-chave override

- Em geral, é uma boa prática anotar os métodos que foram sobrescritos quando herdamos de uma superclasse
- Se o método não existe em nenhuma superclasse, o compilador acusa erro
- Também ajuda no entendimento do código