Programação Orientada a Objetos
Classes e Objetos

Alexandre Mello

Fatec Campinas

2024

Roteiro

Introdução

- Na POO você só se preocupa com o que o objeto expõe, não como o mesmo é implementado
- Um objeto nunca deve manipular diretamente os dados internos de outro objeto
  - Esse manipulação deve ser feita somente via métodos
  - Isso garante o princípio de encapsulamento

Introdução

- O mecanismo que torna e reutilização de código efetiva é a herança (junto com polimorfismo)
- Em Java, diz-se que uma classe estende (**extends**) a outra
- Toda classe em Java já estende automaticamente uma "classe base cósmica" chamada de **Object**.

Objeto

- Uma coisa apresentada ou capaz de ser apresentada
- Um objeto normalmente contém:
  - Atributos que o descrevem
  - Um estado, que reflete o conteúdo dos seus atributos em determinado momento
  - Comportamentos, que são previamente definidos para o objeto, mas que podem se alterar, de acordo com o estado que o objeto assume
- Ex.: um relógio

Objeto

- Entidades que
  - Possuem características próprias (atributos)
  - Desempenham funções (conjunto de tarefas) específicas em um contexto ou ambiente
- Normalmente comparados a componentes ou partes de um cenário
- Em OOP, interessa-nos o que um objeto oferece, ou expõe, e não como foi internamente implementado

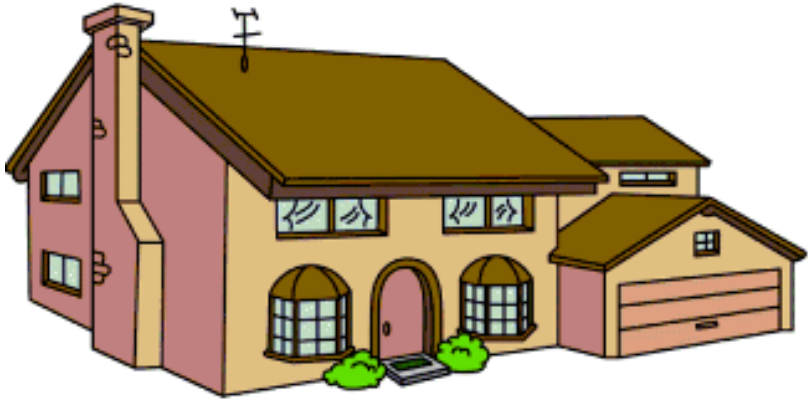
Objeto

- Um objeto sempre estará associado ao:
  - **Estado:** definido pelas propriedades e pelos valores atuais
  - **Comportamento:** definido pela forma como reage em termos de mudança de estado e relacionamento com os demais objetos
  - **Identidade:** é a propriedade pela qual ele se distingue dos demais

Classes

- De forma geral, uma classe define o modelo ou gabarito a partir do qual o objeto é criado
- Um objeto é criado pela **instância** de uma classe, ou seja, pela alocação de memória conforme o modelo da classe
- Modelo: estruturas de dados internos (**atributos**) + **métodos** para acesso aos atributos
- Todos os objetos instanciados da mesma classe têm a mesma estrutura interna
- Classes abstratas: objetos nunca são instanciados diretamente

Classes versus objetos



= **Objeto**

= **Classe**

Atributos

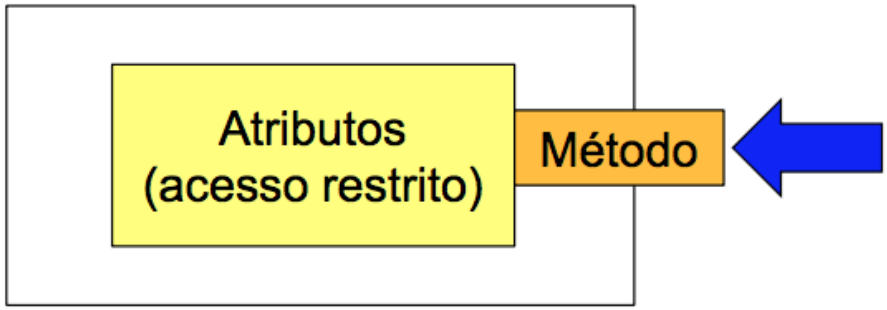
- Os atributos devem ser manipulados exclusivamente por serviços associados à classe
- Diferentes objetos de uma mesma classe não compartilham dos mesmos atributos, cada um possui sua própria cópia do atributo

Métodos

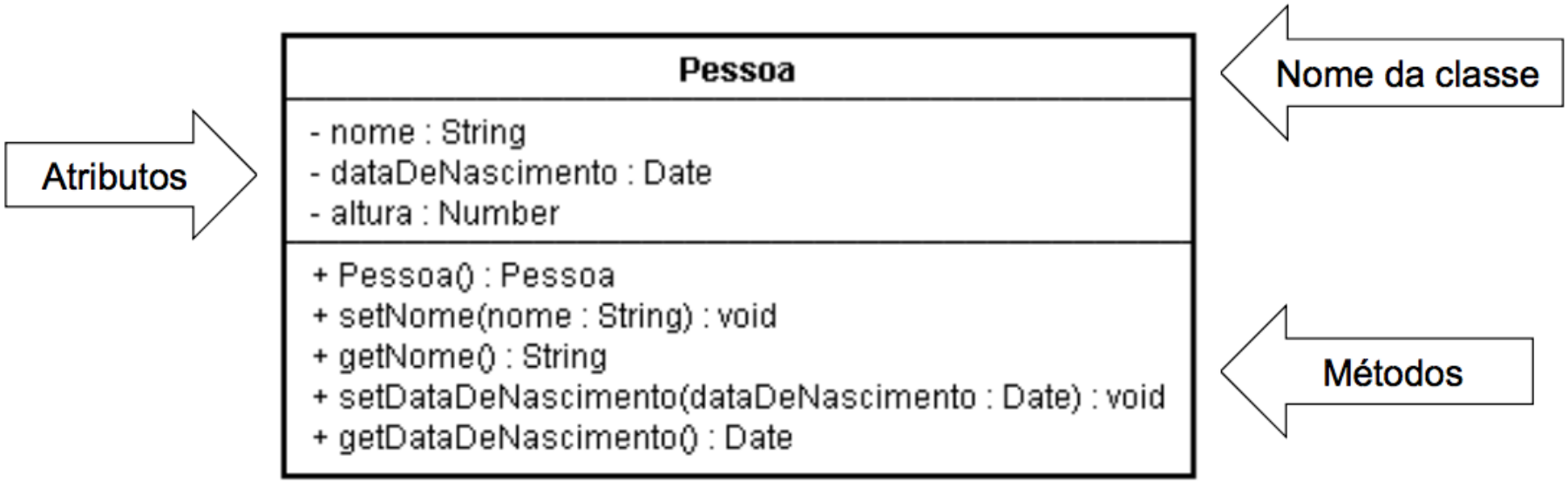
- Os métodos são operações que podem ser executadas pelos objetos
- É um comportamento específico, residente no objeto, que define como ele deve agir quando exigido

Encapsulamento

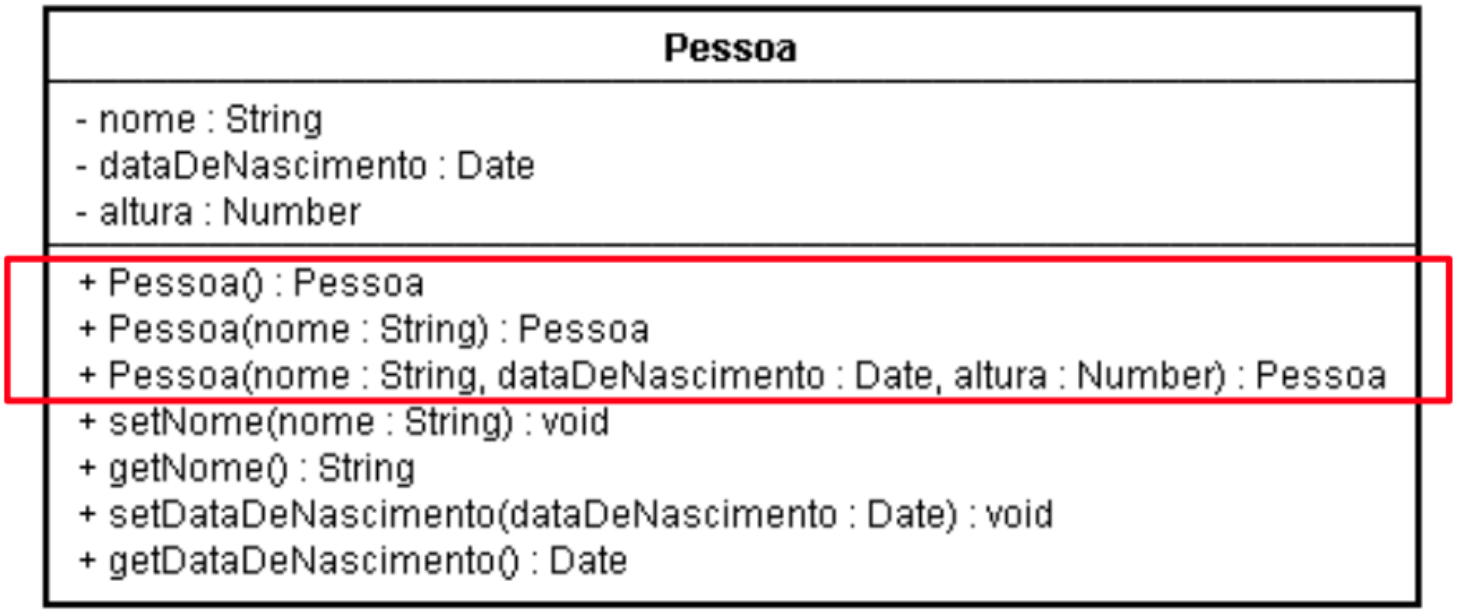
- Conjunto de operações e atributos que representam o estado e um tipo de objeto
- O estado é acessível ou modificável somente pela interface provida pelo encapsulamento
- Através do encapsulamente podemos deixar acessíveis apenas os métodos que manipularão os atributos do objeto (visão "caixa preta" do objeto)



Notação gráfica



Sobrecarga


Pessoa
- nome : String
- dataDeNascimento : Date
- altura : Number
+ Pessoa() : Pessoa
+ Pessoa(nome : String) : Pessoa
+ Pessoa(nome : String, dataDeNascimento : Date, altura : Number) : Pessoa
+ setNome(nome : String) : void
+ getNome() : String
+ setDataDeNascimento(dataDeNascimento : Date) : void
+ getDataDeNascimento() : Date

Visibilidade

- \- Privada (private): acessível somente pela própria classe
- \+ Pública (public): visível a instância de objetos de quaisquer outras classes
- \# Protegida (protected): visível a objetos da própria classe ou subclasse
- ~ Pacote (package): visível para objetos de classes de um mesmo pacote

Abstração

- Identificar artefatos de software na modelagem de um domínio ignorando aspectos não relevantes
- Classes são abstrações de conceitos

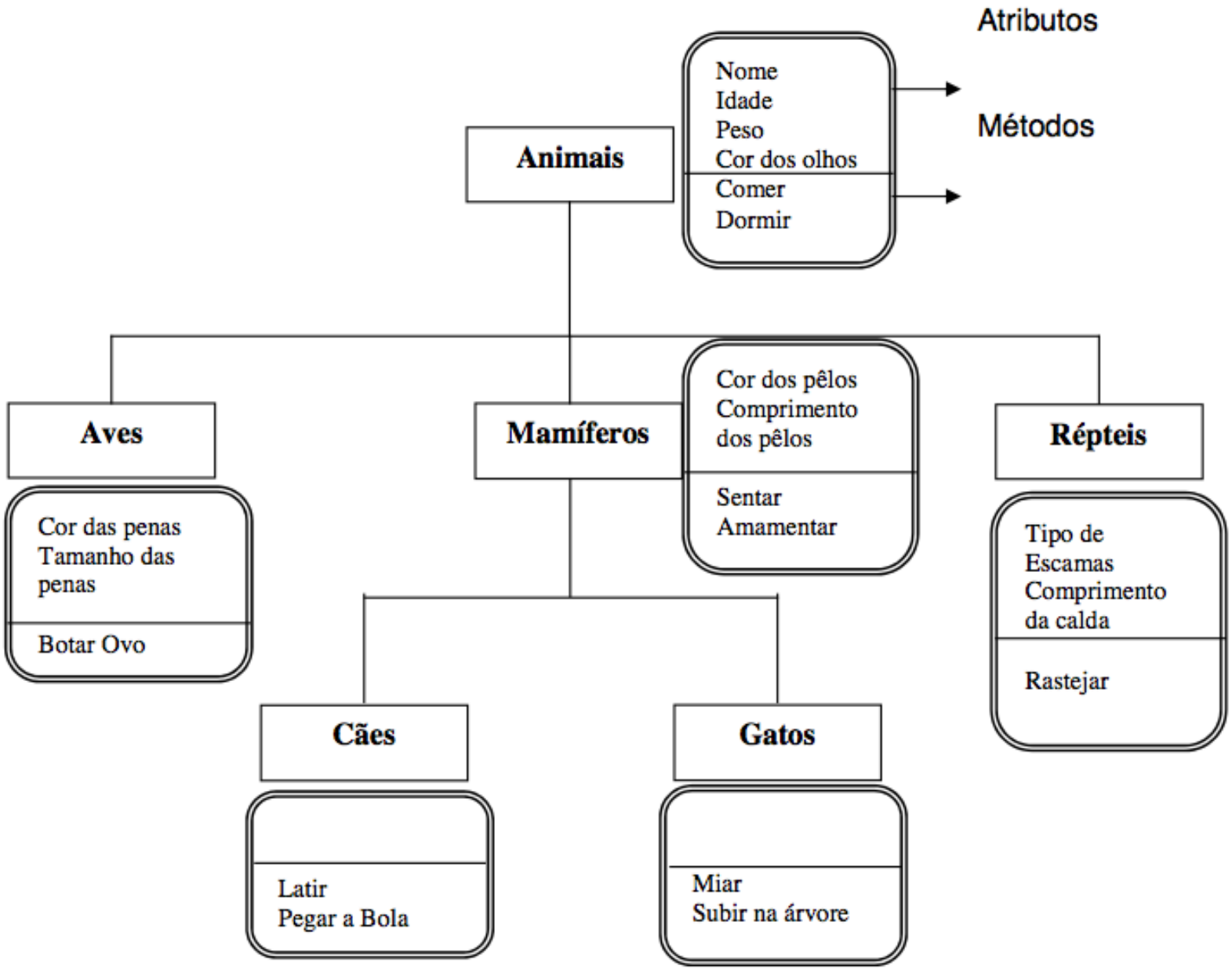
Encapsulamento

- Ideia principal: um sistema OO não deve depender de sua implementação interna, mas sim de sua interface
- Devemos utilizar os qualificadores de acesso (visibilidade) apresentados anteriormente
- O conteúdo dos atributos deve ser acessado por métodos públicos, como **get** e **set**

Herança

- Permite a hierarquização das classes
- Uma classe mais especializada (subclasse) herda as propriedades de uma classe mais geral (superclasse)
- Uma subclasse pode sobrescrever o comportamento de uma superclasse
- É um relacionamento "é-um(a)"
- Promove re-uso

Herança


Atributos
Métodos
Animais
Nome
Idade
Peso
Cor dos olhos
Comer
Dormir
Aves
Cor das penas
Tamanho das penas
Botar Ovo
Mamíferos
Cor dos pêlos
Comprimento dos pêlos
Sentar
Amamentar
Répteis
Tipo de Escamas
Comprimento da calda
Rastejar
Cães
Latir
Pegar a Bola
Gatos
Miar
Subir na árvore

Polimorfismo

- Possibilita que um objeto de uma determinada classe mais genérica (superclasse) possa assumir diferentes comportamentos, gerando objetos distintos
- Pressupõe a existência de herança entre as classes e redefinição de métodos (**overriding**)

Mensagens

- Objetos se comunicam por envio e recebimento de mensagens
- Uma mensagem contém alguma forma de informação
- Por exemplo: um objeto **gerente** pode necessitar enviar um email (método) já definido em um objeto **email**. Então, o objeto **gerente** solicita ao objeto **email** que faça isso por ele

Exercício

Identifique classes, objetos, atributos e métodos:
Biblioteca universitária com as caraterísticas

- Cadastro dos usuários (professores, alunos, etc)
- Cadastro de obras da biblioteca (livros, periódicos, etc)
- Língua e mídia de cada exemplar
- Controle de nacionalidade dos autores das obras
- Cadastro de editoras
- Histórico e gerenciamento de empréstimos