Características das Linguagens de Programação I

Conteúdo: Introdução à programação orientada a objetos

Prof. Dsc. Giomar Sequeiros
giomar@eng.uerj.br

Paradigmas de Programação

Paradigmas de Programação

Paradigma de programação **estruturada**

Paradigma de programação **orientada** a **objetos**

Programação Estruturada

Consiste no mapeamento do problema do mundo real a ser resolvido em um modelo computacional.

Caracteriza-se por:

- Criar um conjunto de funções ou procedimentos (algoritmos) para resolver o problema
- Os comandos de um programa são executados sequencialmente
- Encontrar modos apropriados de armazenar os dados
- Seus códigos ficam em um mesmo bloco
- Ex. C, Pascal, Fortran

Programação Estruturada

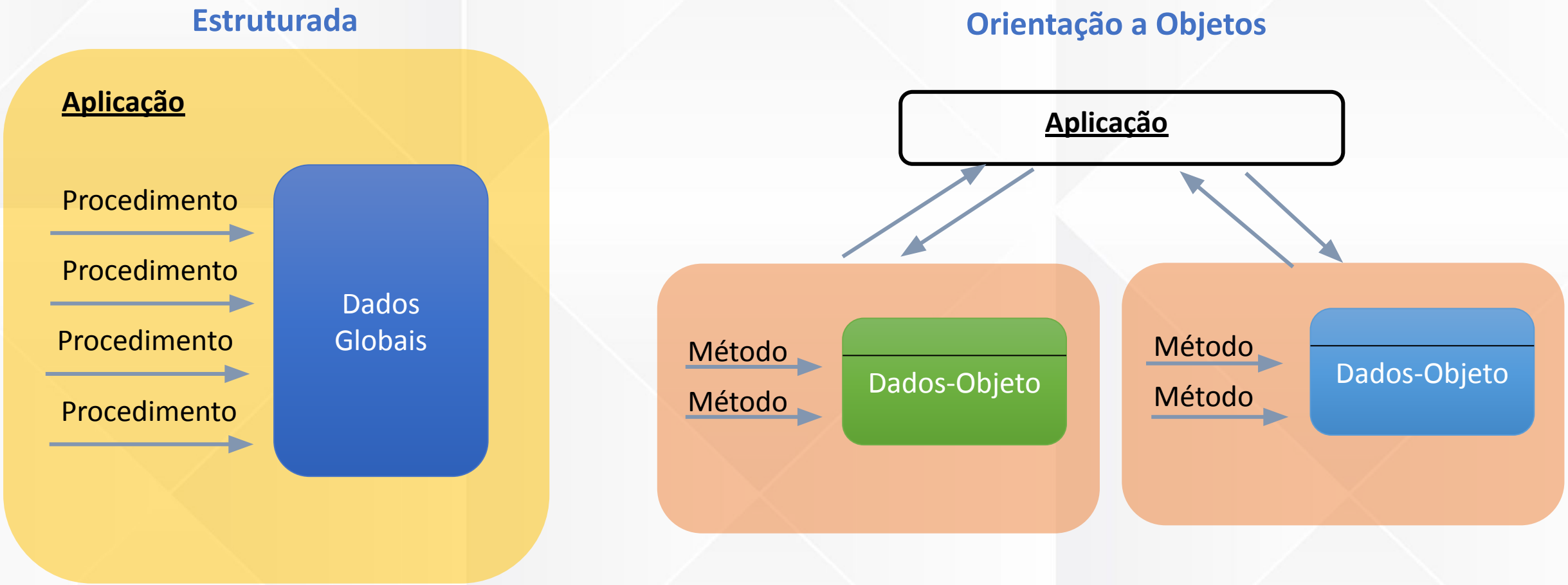
- Problemas
  - Decomposição funcional, leva o desenvolvedor a decompor o sistema em partes menores (funções), criando um emaranhado de inúmeras funções que chamam umas às outras.
  - Geralmente não há separação de conceitos e responsabilidades, causando dependência enormes no sistema, dificultando futuras manutenções no código do programa
  - Não existe muito reaproveitamento de código, ao contrário, muitas vezes se tem muito código duplicado

Programação orientada a objetos

- Composição de programa:
  - A funcionalidade do programa é agrupada em objetos
  - Os dados do programa são agrupados em objetos
  - Os objetos **agrupam dados em funções** correlacionadas

- Fluxo de execução:
  - Os objetos colaboram entre si para solução dos objetivos
  - A colaboração se realiza através de chamadas de rotina
  - A comunicação é feito através do envio e recebimento de mensagens
  - Ex. Java, C++, C#, Delphi, Ruby, Python...

Programação estruturada X Programação orientada a objetos



Vantagens e desvantagens da programação estruturada

- Vantagens
  - É fácil de entender.
  - Ainda muito usada em cursos introdutórios de programação.
  - Execução mais rápida.

- Desvantagens
  - Baixa reutilização de código
  - Códigos confusos: Dados misturados com comportamento

Vantagens e desvantagens da programação orientada a objetos

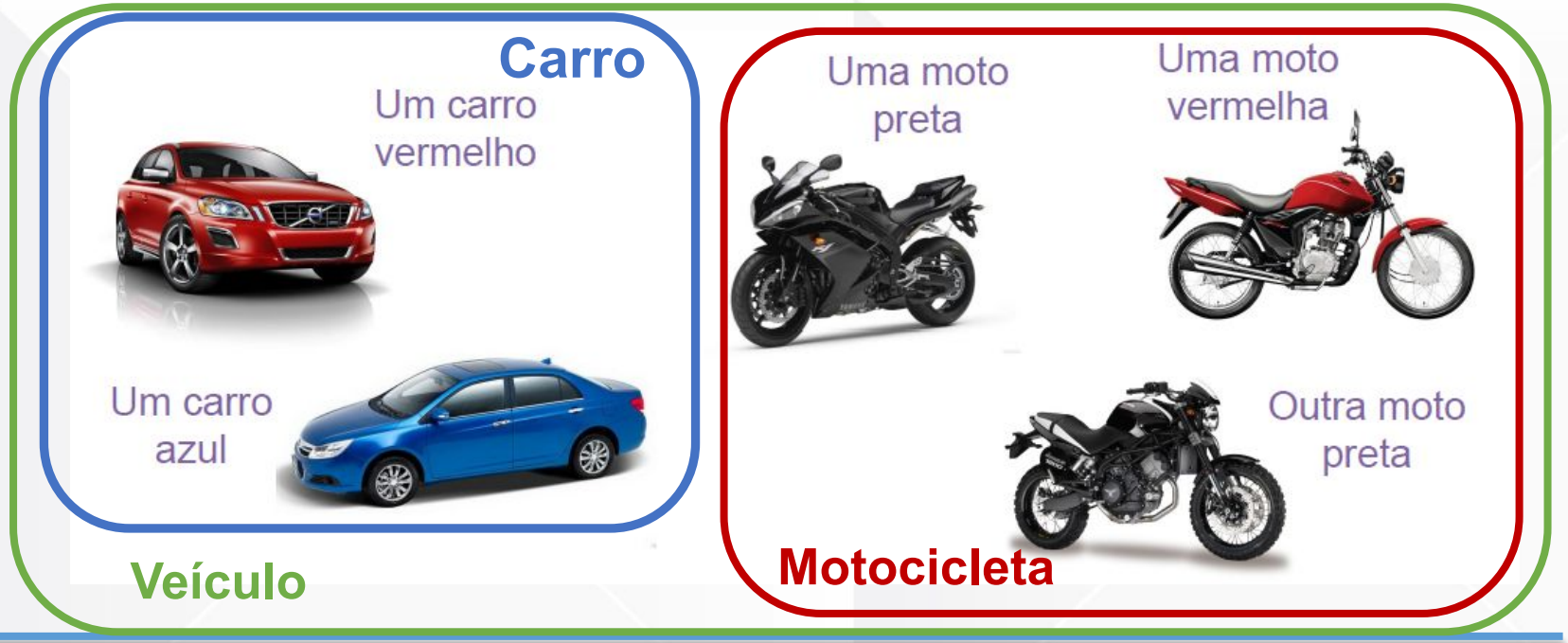
- Vantagens
  - Melhor organização do código
  - Bom reaproveitamento de código
  - Modularidade

- Desvantagens
  - Desempenho mais baixo que o paradigma estruturado
  - Mais difícil compreensão

Programação Orientada a Objetos

Introdução a objetos

- Surgiu no fim dos **anos 60**
- O ser humano se relaciona com o mundo através do conceito de **objetos**.
- Estamos **identificando** sempre objetos ao nosso redor.Para isso:
  - Atribuímos nomes
  - Classificamos em grupos



Objeto

- Um objeto é a representação computacional de um elemento ou processo do mundo real
  - Um carro específico no estacionamento
  - Um aluno específico na sala de aula
  - Um professor específico da Faculdade

Características e comportamento de um objeto

- Objetos possuem **características** e **comportamentos**
- Uma característica descreve uma propriedade de um objeto.
- O comportamento descreve uma ação de um objeto.

Objeto: Carro

Características:
- **Marca**
- **Cor**
- **Modelo**
- **Peso**

Comportamentos:
- **Andar**
- **Acelerar**
- **Parar**
- **Estacionar**

Características e comportamento de um objeto

- Exemplos:
  - Cachorros
    - ✔ **Características**: nome, raça, cor
    - ✔ **Comportamentos**: latir, correr
  - Bicicleta
    - ✔ **Características**: marca, marcha atual, velocidade atual
    - ✔ **Comportamentos**: trocar marcha, aplicar freios

As características que descrevem um objeto são chamadas de atributos

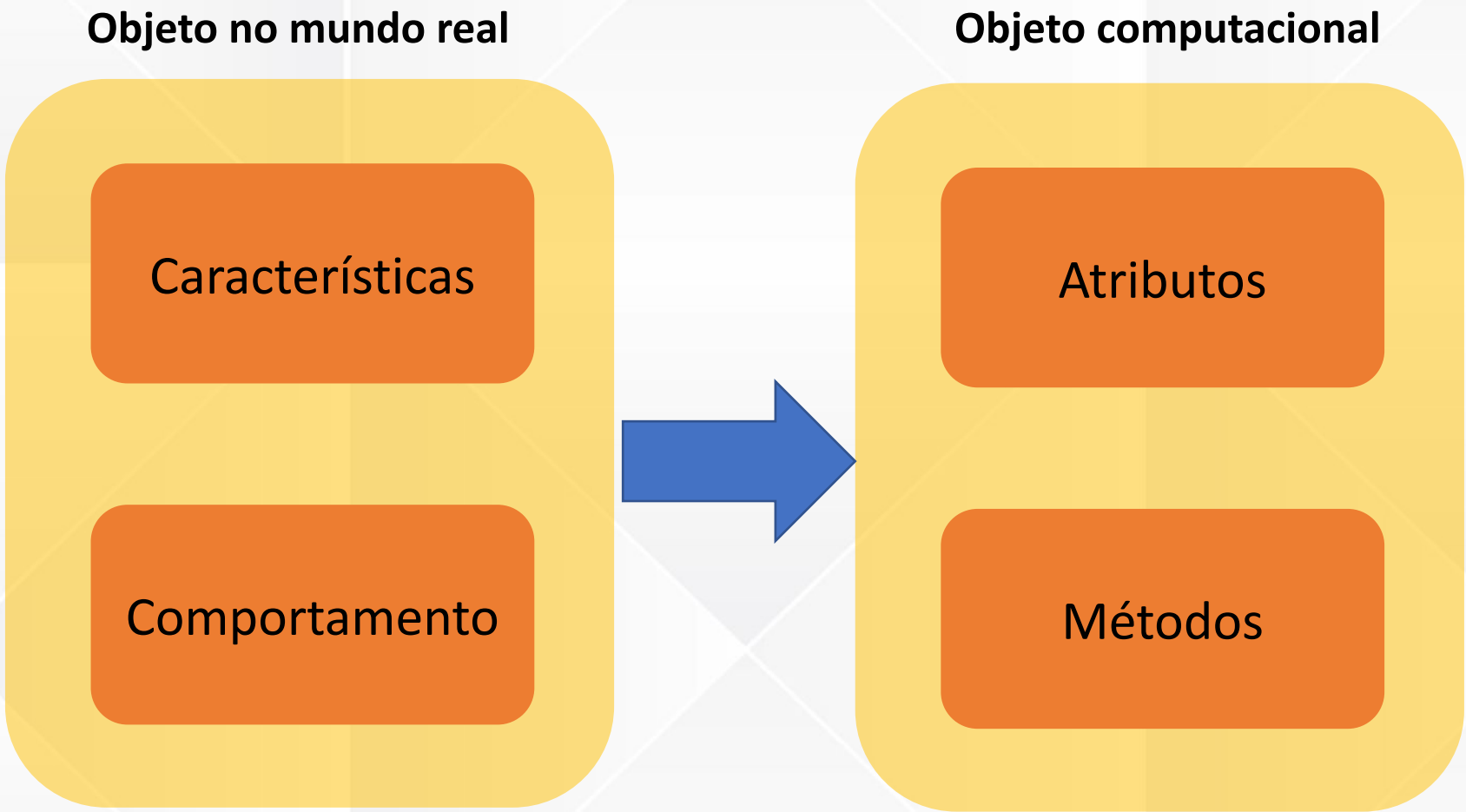
As ações que um objeto pode executar são chamadas de métodos ou serviços

Objetos: exemplos

Cachorro	
Nome	Pluto
Idade	2 anos
Comprimento dos pelos	Compridos
Cor dos pelos	Café
Cor dos olhos	Castanhos
Peso	8kg

Cachorro	
Nome	Snoopy
Idade	4 anos
Comprimento dos pelos	Curtos
Cor dos pelos	Marrão
Cor dos olhos	Castanhos
Peso	5kg

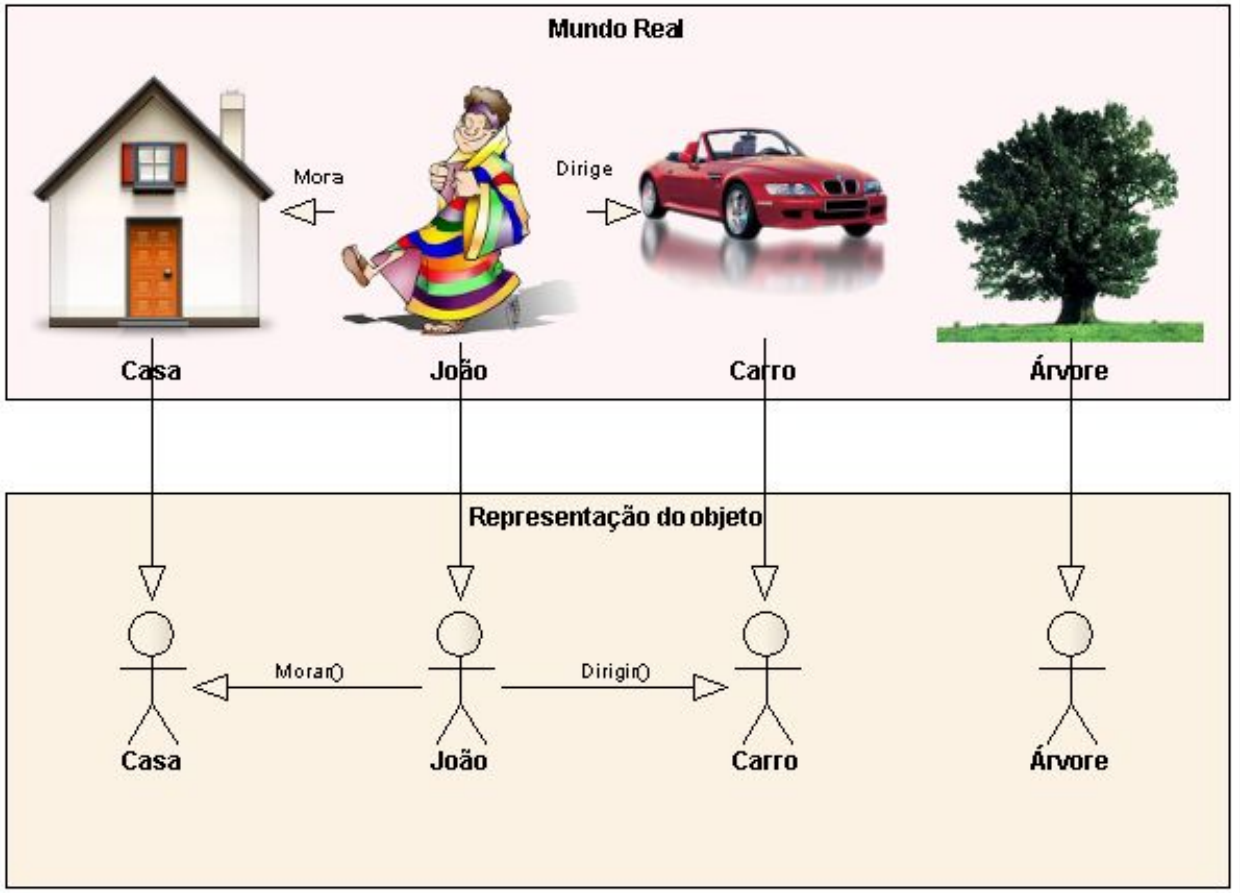
Mapeamento entre objetos reais e computacionais



Orientação a objetos

Objetos não são considerados isoladamente

- Um processo natural é identificar características e comportamentos semelhantes entre objetos
- Objetos com características e comportamentos semelhantes são agrupados em classes
- Ex. de representação na UML

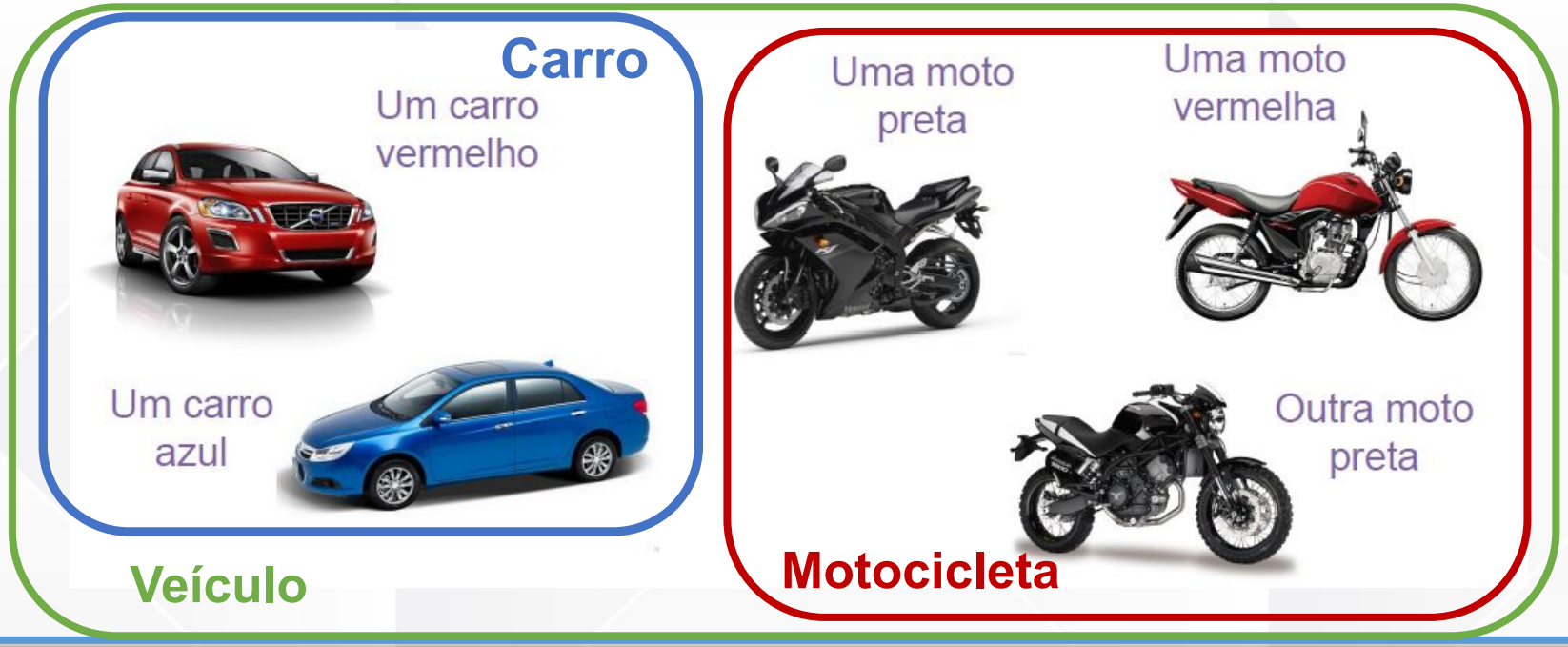


Classes

- Uma classe representa um conjunto de objetos que possuem características e comportamentos comuns
- Classes são as unidades fundamentais na construção de programas orientados a objetos.
- Um objeto é uma instância de uma única classe.
- Uma instância de um objeto é uma unidade de programação que é armazenada em uma variável

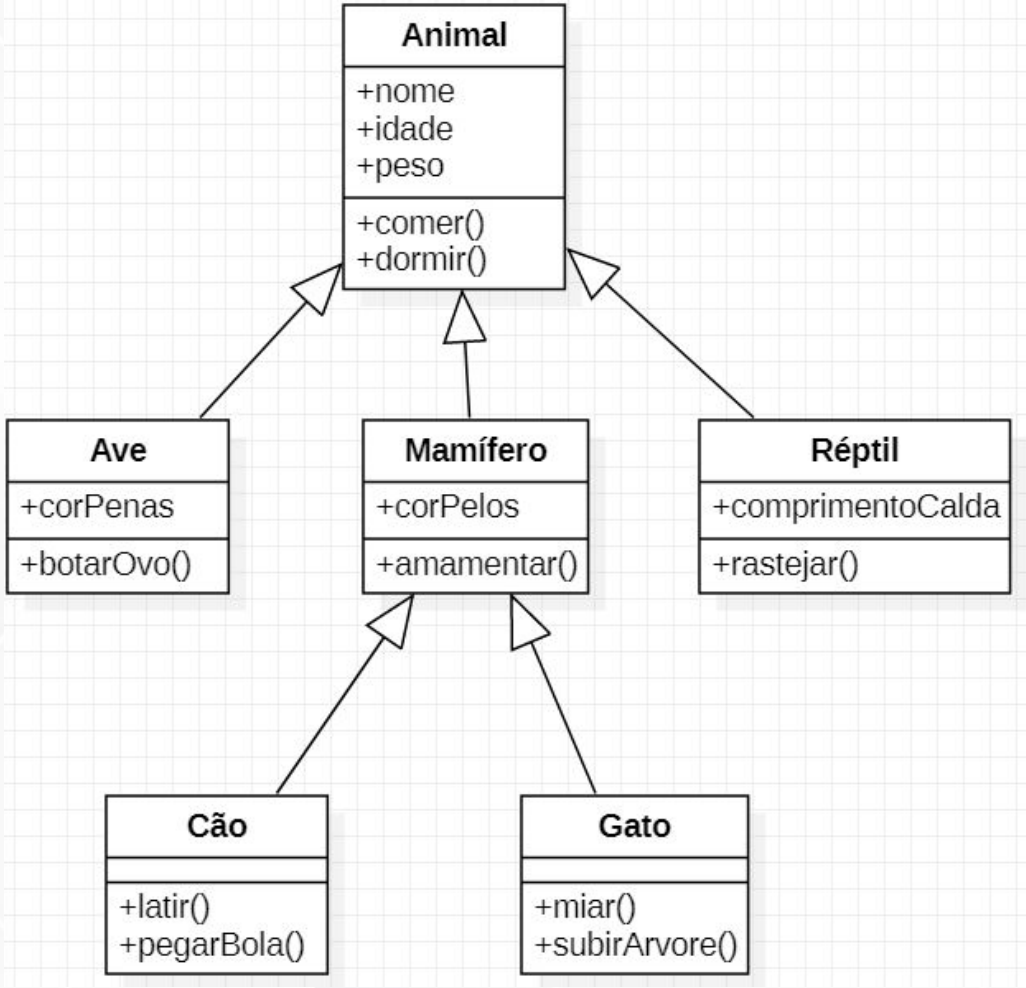
Classes

- Muitos objetos do mundo real possuem características comuns e podem ser agrupados de acordo com elas.
- Ex. As classes Carro e Motocicleta possuem algumas características e métodos comuns. Surge então o conceito de subclasse e superclasse



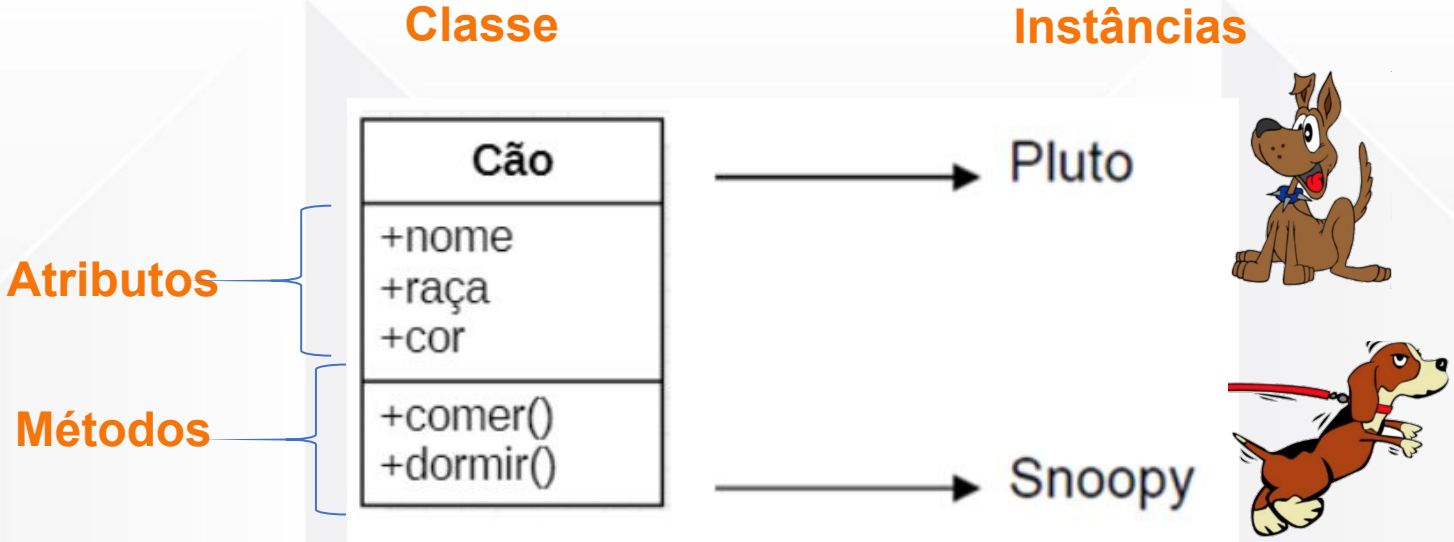
Classes

- Chamamos de “ancestrais” às classes das quais as outras dependem e de “descendentes” as classes originadas a partir de outra.
- No exemplo,
  - Ancestral da classe mamíferos animais
  - Descendentes da classe mamíferos cães e gatos.
  - Os descendentes descrevem somente atributos e métodos que apresentarão alterações em relação à descrição do ancestral



Classes e instâncias

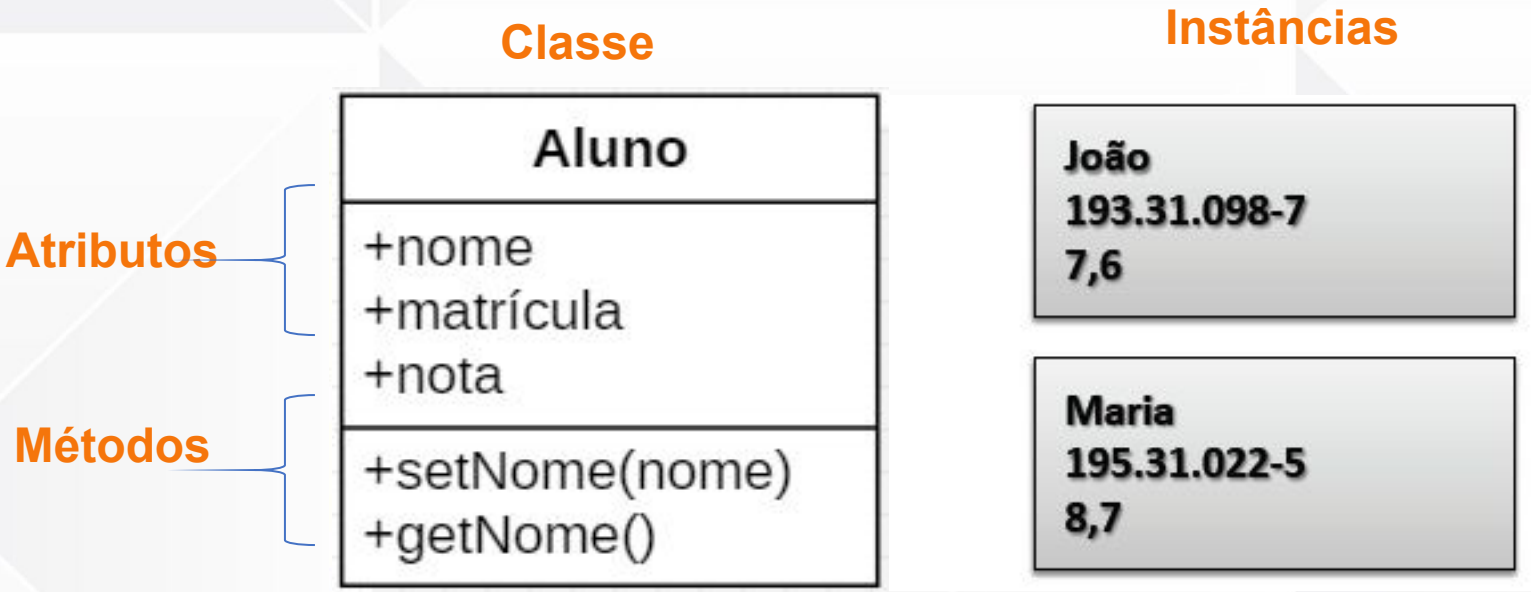
- As classes são definições de como os objetos devem ser e não existem na realidade. Somente os objetos têm existência.
  - Exemplo: quando vamos mostrar nosso cachorro a alguém, não dizemos “esse é um cão”, e sim “esse é o pluto”, ou “snoopy”. O que se pode ver não é uma classe de seres, mas um cachorro específico, ou seja um objeto.



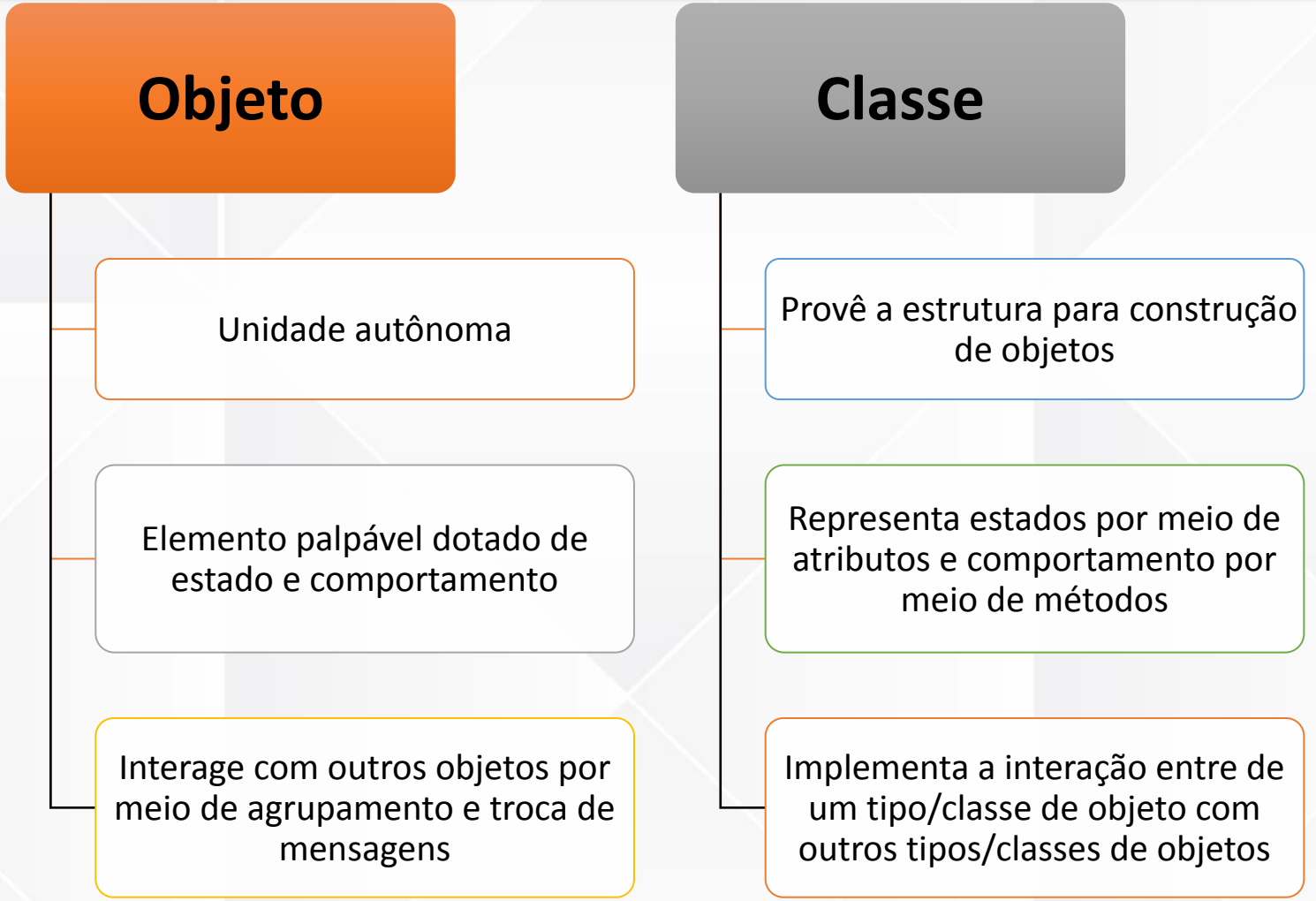
As classes provêem a estrutura para a construção de objetos estes são ditos instâncias das classes

Classes e instâncias

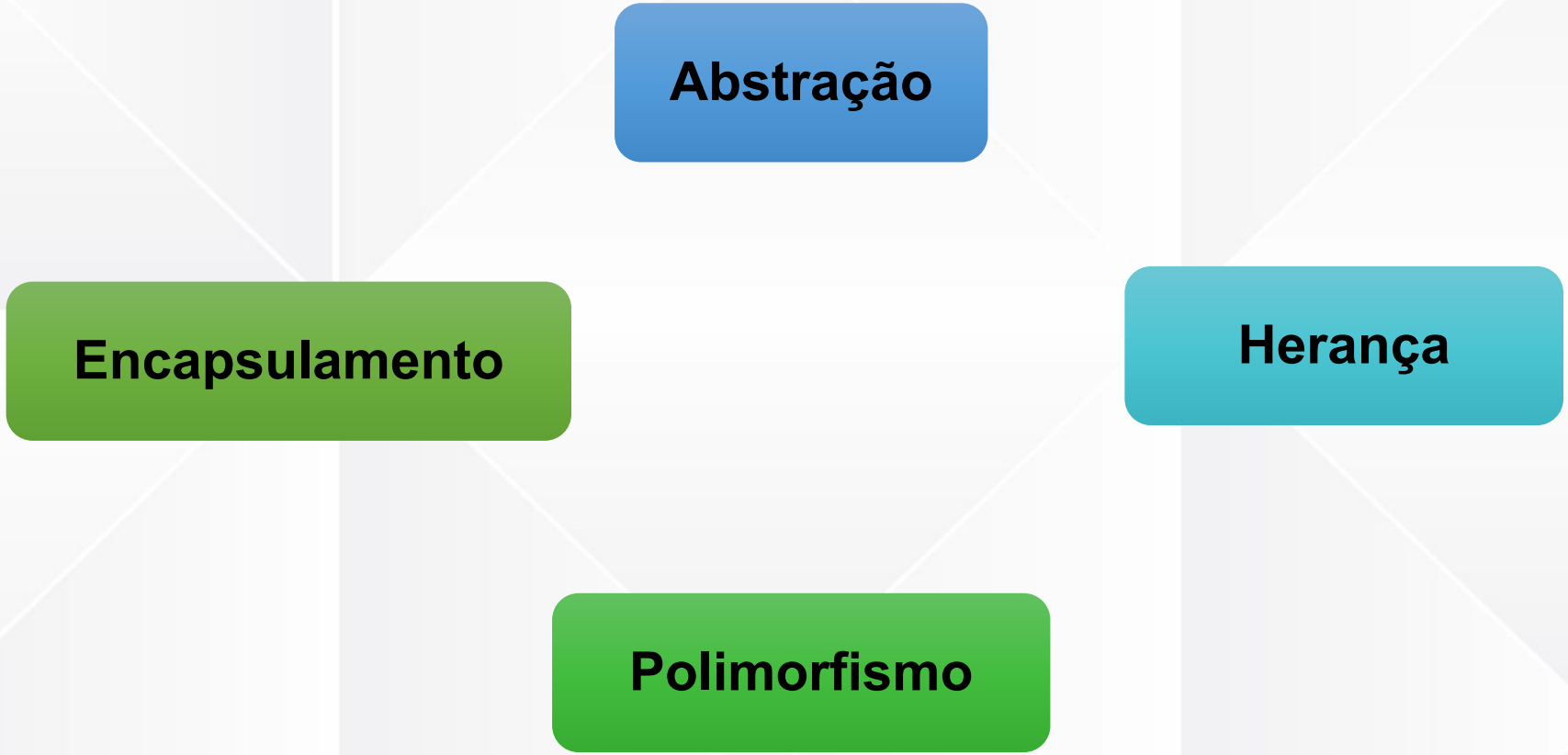
- As classes provêem a estrutura para a construção de objetos estes são ditos instâncias das classes



Objetos x Classes

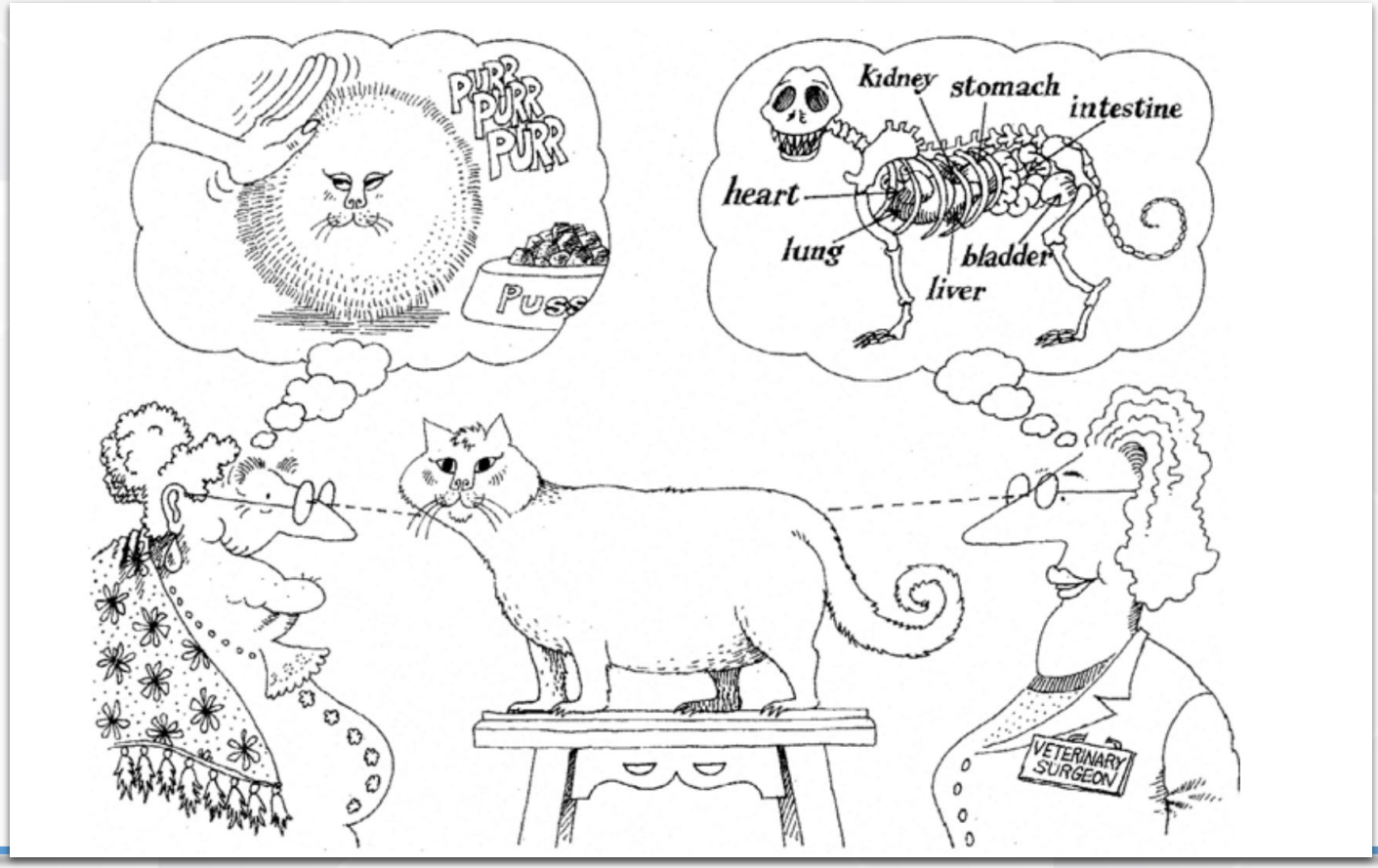


Pilares da programação orientada a objetos



Abstração

Definição, “ato de separar mentalmente um ou mais elementos de uma totalidade complexa (coisa, representação, fato), os quais só mentalmente podem subsistir fora dessa totalidade”.



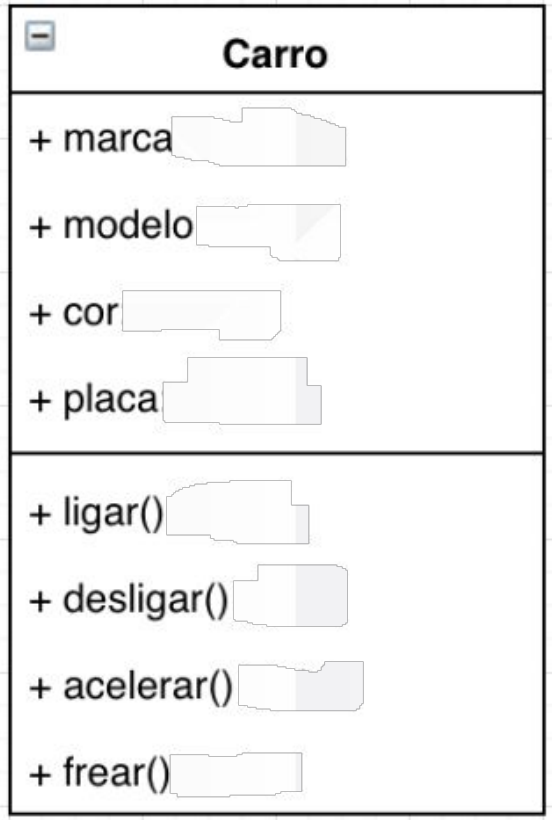
Abstração

- Possuímos o recurso da abstração como forma de entender problemas tidos como complexos. Assim, diante de um problema complexo, procuramos dividi-lo em problemas menores, e feito isso, resolvemos cada um deles até encontrar a solução do problema inteiro.

- Pelo princípio da abstração nós isolamos os objetos que queremos representar do ambiente complexo em que se situam e nesses objetos representamos só as características que são relevantes para o problema em questão.

  Por **exemplo**: toda pessoa tem um atributo para "**cor dos olhos**", mas em um sistema de folha de pagamento, essa informação não é relevante, portanto, ela não será incluída na relação de características de pessoas que queremos armazenar em nosso sistema.

Abstração: exemplo



Encapsulamento

- Consiste na separação dos aspectos internos e externos de um objeto;
- Com este mecanismo podemos ocultar detalhes de uma estrutura complexa, que poderiam interferir durante o processo de análise.
- A única maneira de conhecer ou alterar os atributos de um objeto é através de seus métodos.
- A vantagem é que o *encapsulamento* disponibiliza o objeto com toda a sua funcionalidade sem que você precise saber como ele funciona internamente, nem como armazena internamente os dados que você recupera.
- O que importa para poder haver interação entre dois objetos é que um conheça o conjunto de operações disponíveis do outro (interface) para que então envie e receba informação, ou mesmo ordene a realização de procedimentos.

Encapsulamento: exemplo 1

O objeto deve esconder seus dados e os detalhes de sua implementação

- Ninguém precisa conhecer **detalhes** dos circuitos de um **telefone** para utilizá-lo;
- Sua **carcaça encapsula** os detalhes e nos provê uma interface amigável.
- A interface: botões, monofone, sinais de tom.

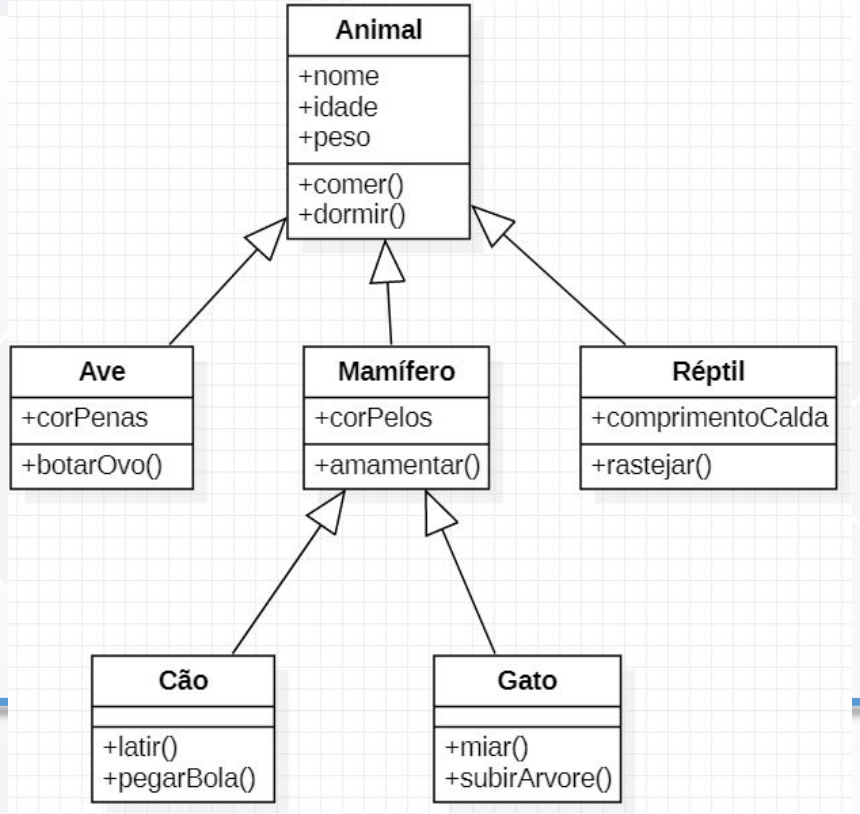
Encapsulamento: exemplo 2

- Se um banco reescrever o software (aperfeiçoando-o) ele não precisa avisar todos os clientes.

- A interface não mudou (o que mudou foram detalhes de implementação

Herança

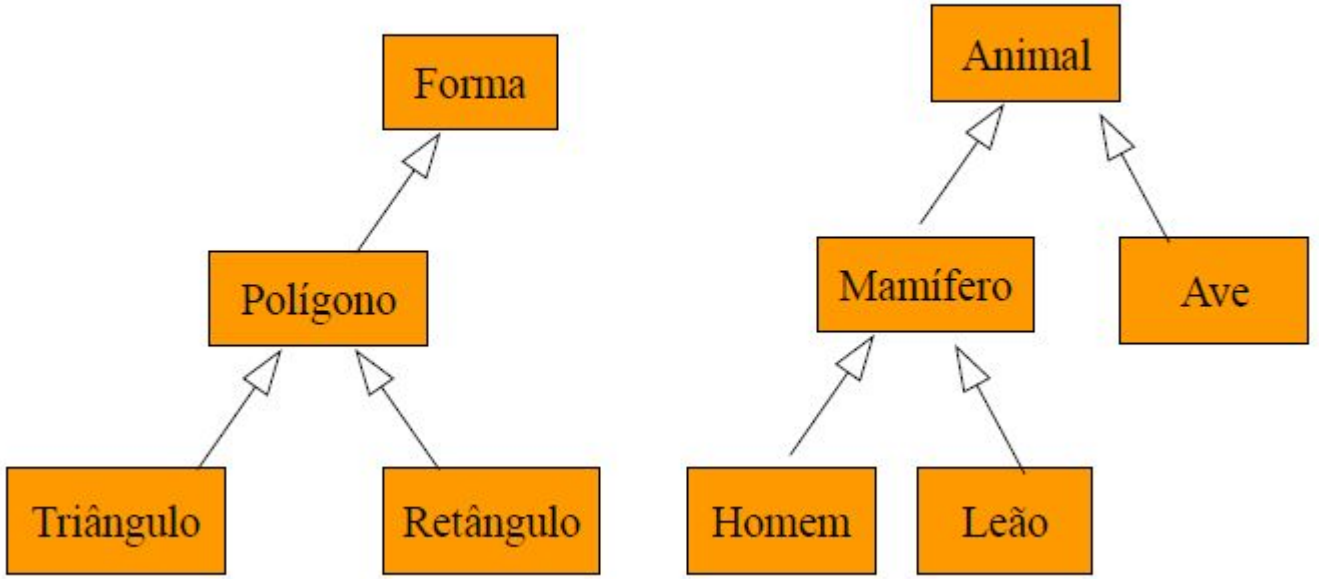
- Herança é o mecanismo pelo qual uma classe obtém as características e métodos de outra para expandi-la ou especializá-la de alguma forma, ou seja, uma classe pode "herdar" características, métodos e atributos de outras classes.
- A herança constitui um mecanismo muito inteligente de **aproveitar código**. É através da herança que os objetos podem compartilhar métodos e atributos.



"Herança significa que todos os atributos e métodos programados no ancestral já estarão automaticamente presentes em seus descendentes sem necessidade de reescrevê-los."

Herança

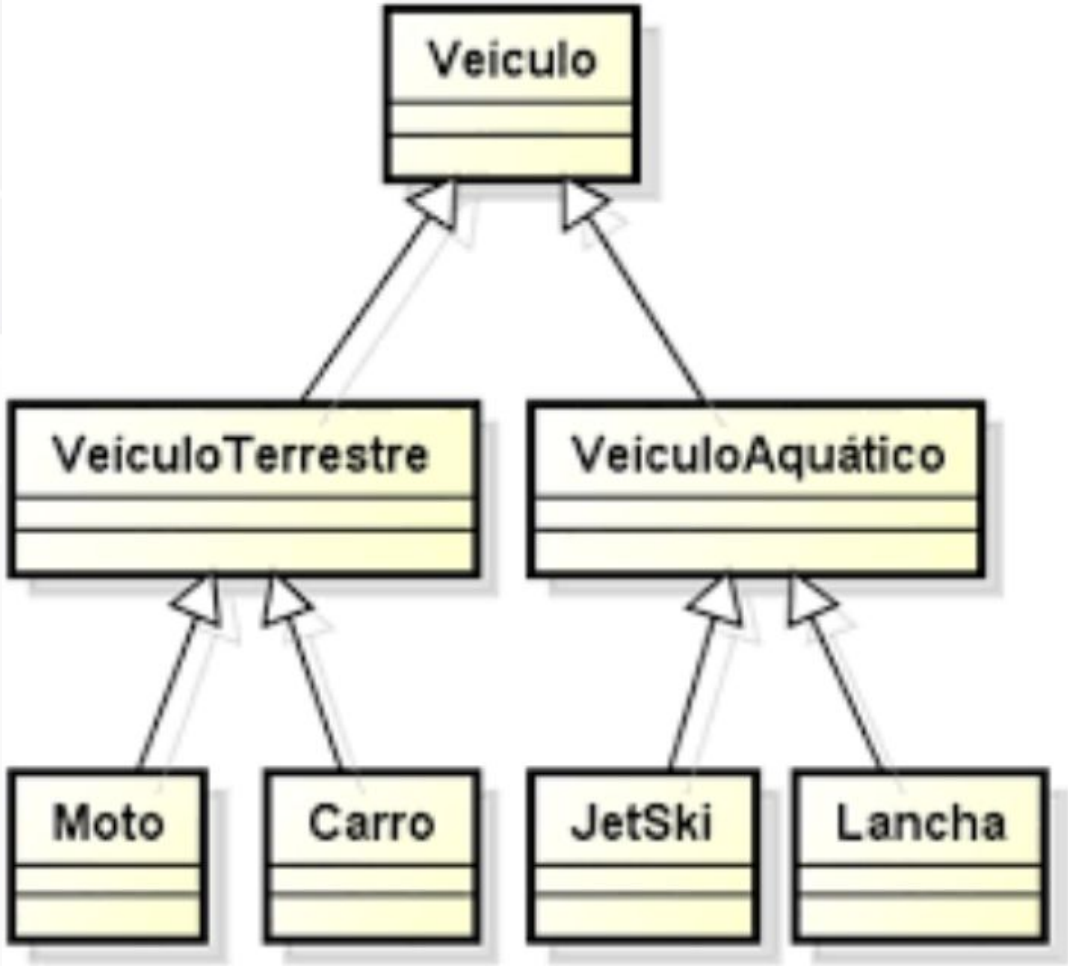
Define níveis de abstração



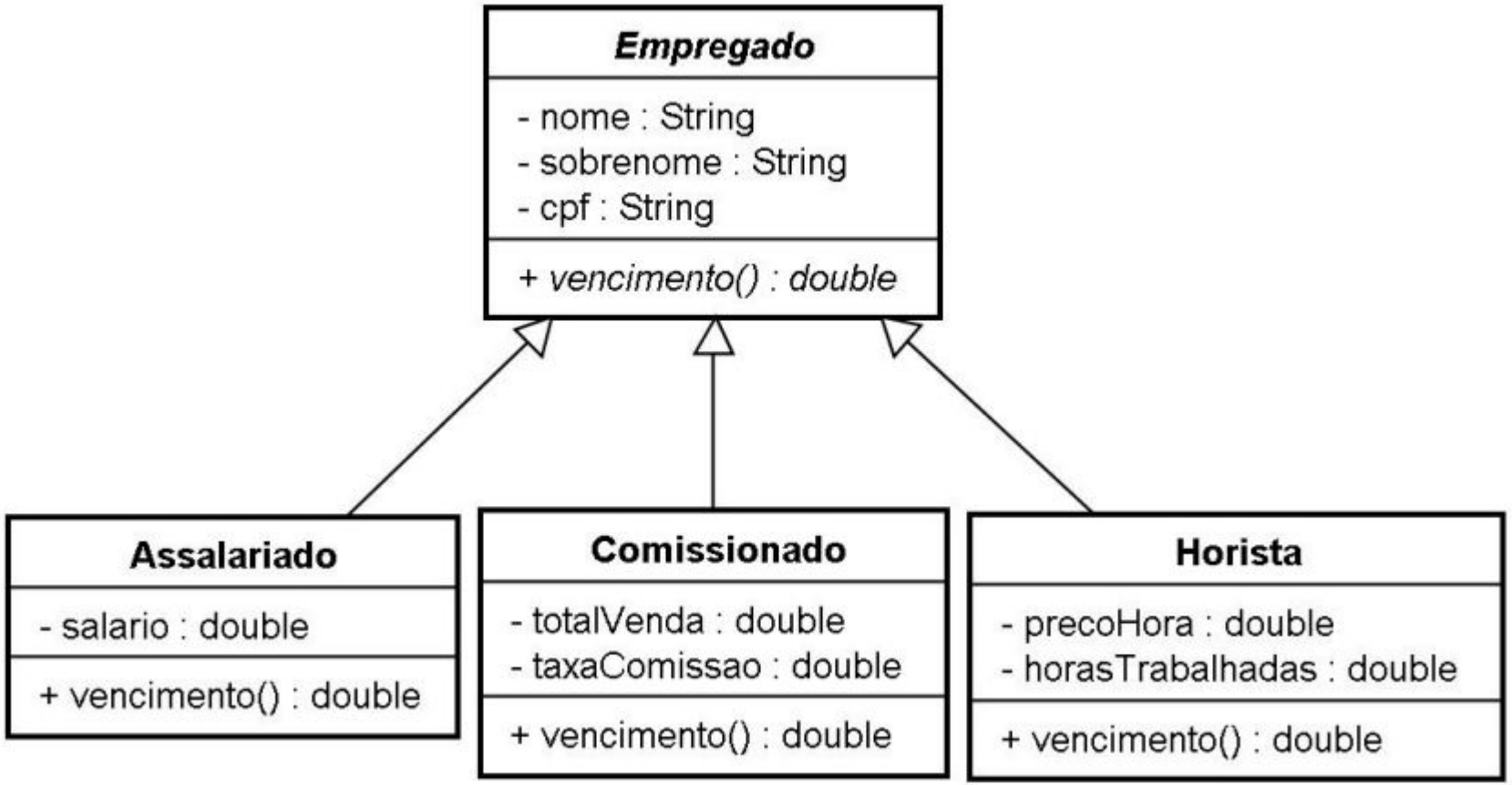
Aumenta abstração

Diminui abstração

Herança: exemplo

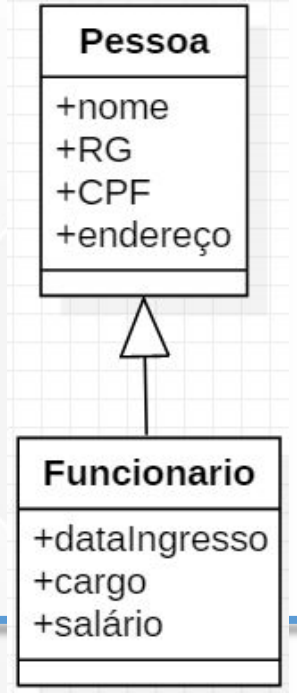


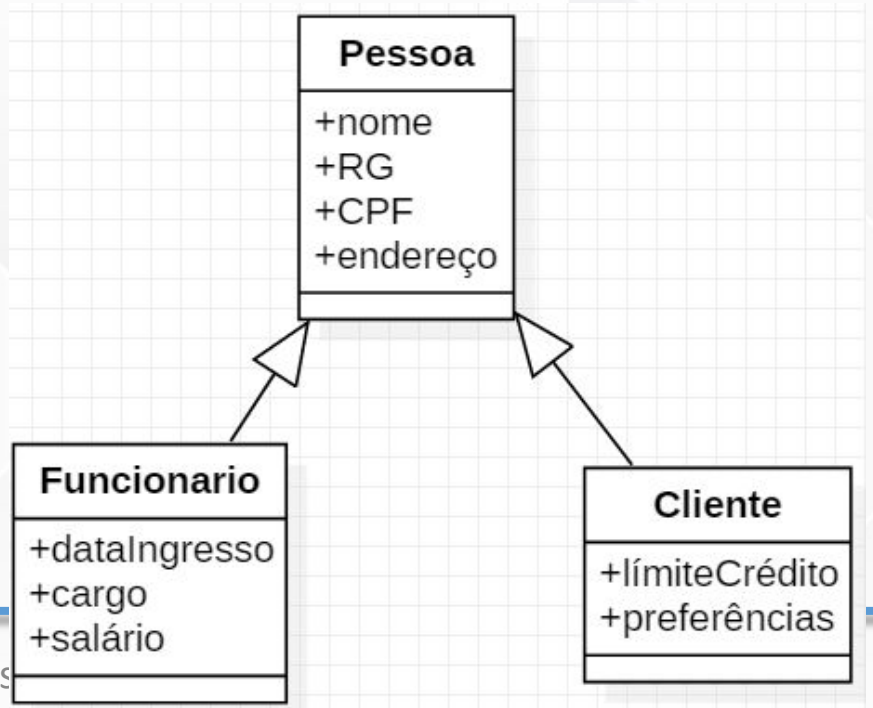
Herança: exemplo



Herança simples

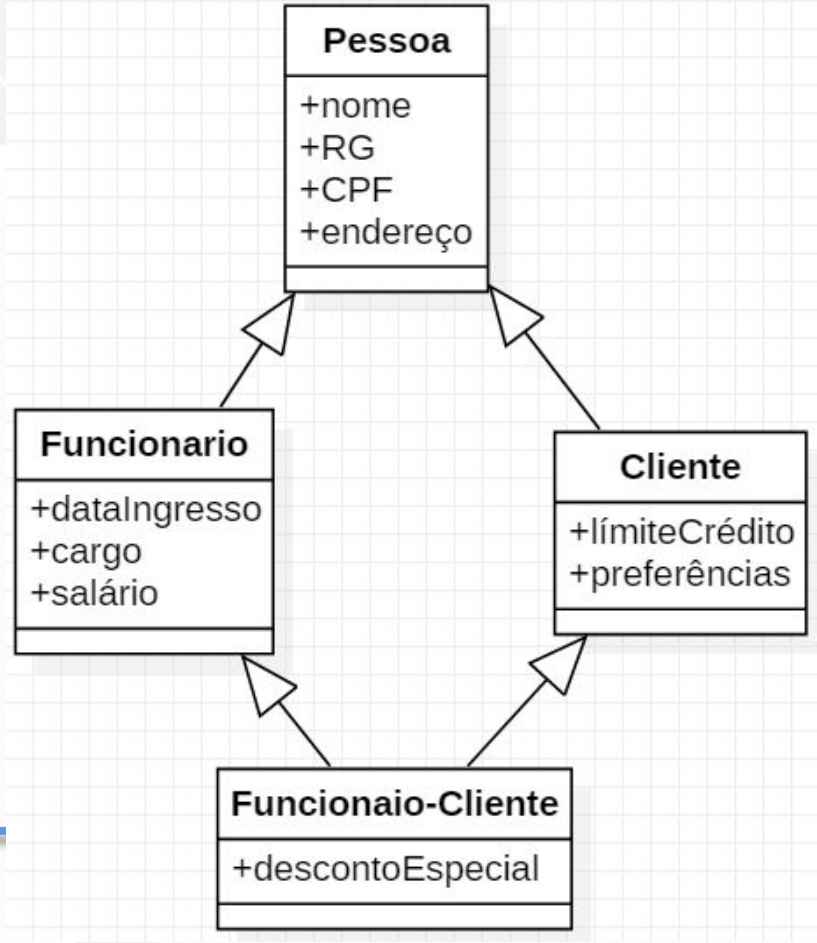
- A herança é denominada simples quando uma classe herda características de apenas uma superclasse.
- Por exemplo, podemos ter como superclasse uma classe chamada **Pessoa**, e dela derivar uma subclasse chamada **Funcionário**.
- Nada impede, entretanto, que a mesma superclasse gere mais de uma subclasse.





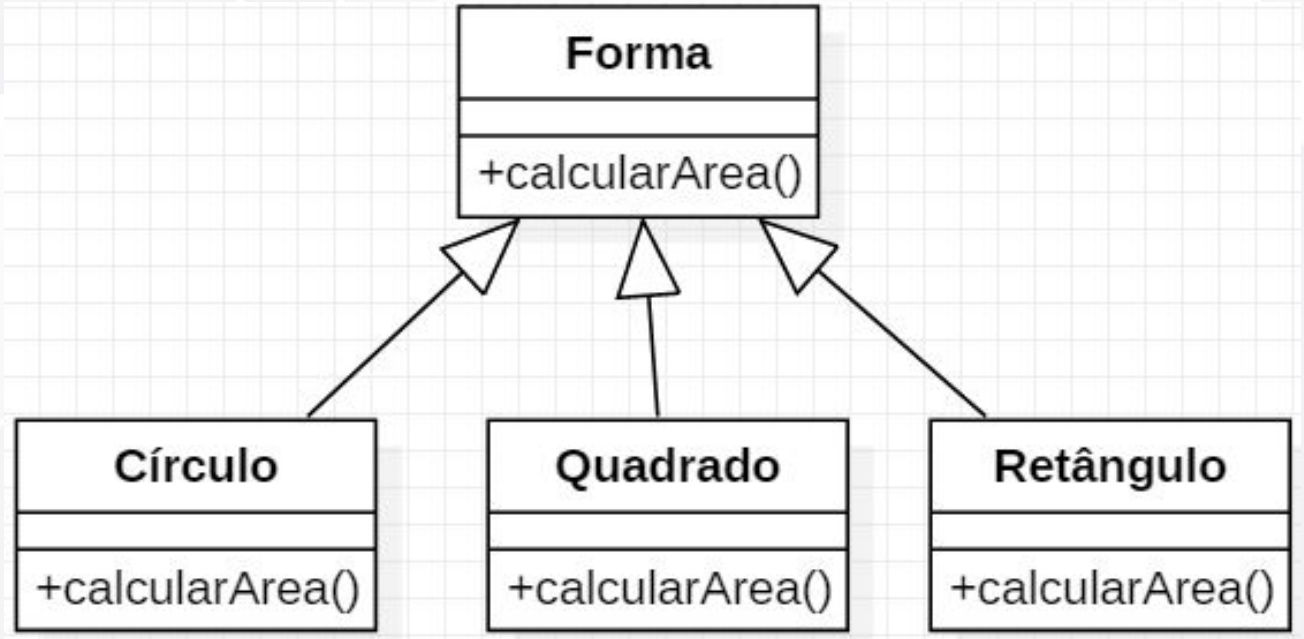
Herança múltipla

A herança é denominada múltipla quando uma classe herda características de duas ou mais superclasses.



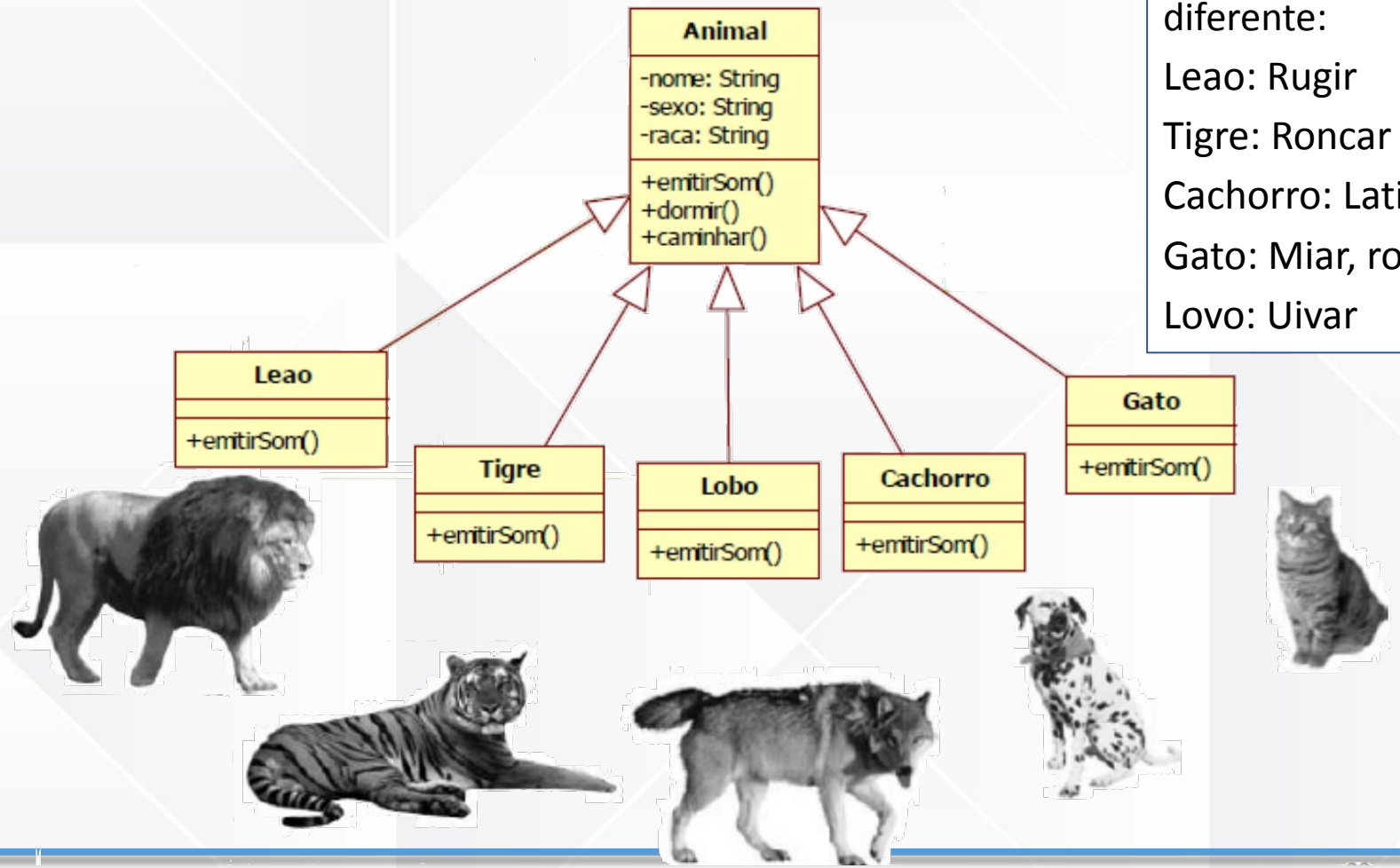
Polimorfismo

"O polimorfismo ocorre quando um método que já foi definido no ancestral é redefinido no descendente com um comportamento diferente."



Poli □ várias;
Morfos □ formas

Polimorfismo: exemplo



Cada animal emite sons diferente:

Leao: Rugir

Tigre: Roncar

Cachorro: Latir

Gato: Miar, rosnar

Lovo: Uivar

Vantagens da orientação a objetos

- Reusabilidade de código
- Escalabilidade de aplicações
- Multidesenvolvimento
- Facilidade de manutenção

Desvantagens da orientação a objetos

- Complexidade no aprendizado para desenvolvedores de linguagens estruturadas
- Maior uso de memória (heap)
- Maior esforço na modelagem de um sistema OO do que estruturado
- Dependência de funcionalidades já implementadas em superclasses no caso da herança, implementações espalhadas em classes diferentes

Orientação a objetos é sempre necessária

- Nem sempre
- Há situações onde o modelo de uma tarefa a ser executada é tão simples que a criação de uma classe para representá-lo torna o problema mais complicado.
- Exemplo: uma Equação

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Exercício

Para atender as necessidades de informação de uma biblioteca universitária foi proposto um sistema que deve atender as seguintes características:

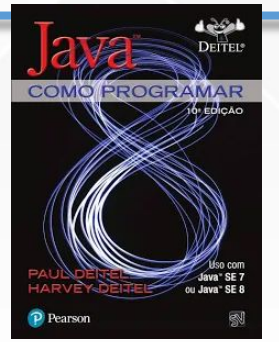
- O cadastro dos usuários da biblioteca com endereço completo. Os usuários podem ser classificados em três grupos: Professores, Alunos e Funcionários.
- O cadastro das obras da biblioteca, que podem ser classificadas em: Livros científicos, periódicos científicos, periódicos informativos, periódicos diversos, entretenimiento, etc.
- A língua em que se encontra o exemplar da obra.
- A mídia onde se encontra o exemplar da obra.
- Os autores da obra com o controle da nacionalidade do autor.
- As editoras dos exemplares com o ano de edição de cada exemplar.

Identifique os possíveis **objetos** com seus respectivos **atributos** e **métodos**.

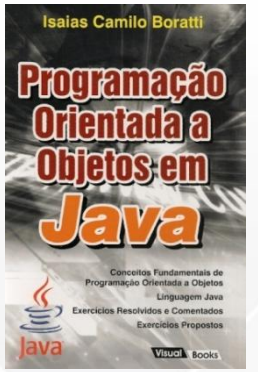
Referências

BIBLIOGRAFIA BÁSICA:

- DEITEL, Harvery M.. Java : como programar. 10ª ed. São Paulo: Pearson - Prentice Hall, 2017.



- BORATTI, Isaías Camilo. Programação Orientada a Objetos em Java : Conceitos Fundamentais de Programação Orientada a Objetos. 1ª ed. Florianópolis: VisualBooks, 2007.



- SIERRA, Kathy; BATES, Bert. Use a Cabeça! Java. 2ª ed. Rio de Janeiro: Alta Books, 2007.