

Introdução ao paradigma de programação: Orientado a Objetos

Profa. Kamila Rios

25/03/2015

Roteiro

- Paradigma de construção de software
- Orientação a Objetos
- Conceitos de Orientação a Objetos (OO)
- Classe, Objeto e Mensagem
- Os pilares da Orientação a Objetos (OO)
- Reuso de Implementação

Paradigma

- Paradigma é a filosofia adotada na construção de software:
 - **Imperativo ou Procedural** (C, Fortran, etc...);
 - **Lógico** (Prolog, etc...);
 - **Orientado a Objetos** (Java, C++, SmallTalk, etc...);
 - **Orientado a Aspectos** (AspectJ, AspectC++, etc...).

Paradigma Orientado a Objetos (1/2)

- Sugere a **diminuição** da distância entre a **modelagem computacional** e o **mundo real**.
- Surgiu na tentativa de solucionar **problemas complexos** existentes através do desenvolvimento de softwares **menos complexos, confiáveis** e **com baixo custo** de desenvolvimento e manutenção.

Paradigma Orientado a Objetos (2/2)

- Permite que **objetos** do **mundo real** sejam mapeados em Objetos no computador, pressupondo que o mundo é **composto por objetos**.
- Os **sistemas** são modelados como um **conjunto de objetos** que **interagem entre si**.

Por que programar Orientado a Objetos?

- Permite **alta reutilização de código**;
- **Reduz tempo de manutenção** de código;
- **Aumenta qualidade e produtividade** → oferece mais facilidade ao desenvolvedor;
- Adoção (aceitação) comercial **crescente**.

Armadilhas da Orientação a Objetos

- Pensar no paradigma OO simplesmente como **uma linguagem**;
- Aversão a reutilização;
- Programação Egoísta:
 - É preciso documentar!

Orientação a Objetos (OO)

- É uma forma de **entender** e **representar sistemas complexos** como estruturas hierárquicas de objetos que se **relacionam**.

Conceitos da Orientação a Objetos

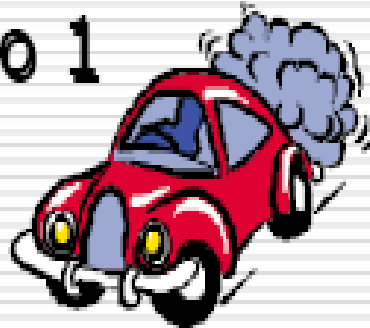
- Classe
- Objeto (Instância)
- Mensagem
- Encapsulamento
- Herança
- Polimorfismo
- ...

Classe (1/3)

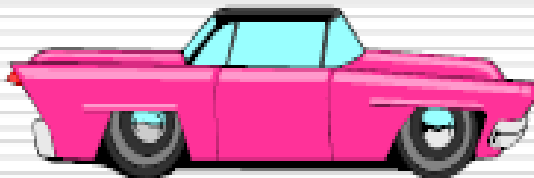
- A classe é a implementação de tipo abstrato de dados (TAD) no paradigma orientado a objetos.
- Uma classe Java é um molde para a criação de objetos. A classe define as propriedades (atributos) e os comportamentos (métodos).
- Além disso, uma classe Java define como produzir (instanciar) objetos a partir dela.

Classe (2/3)

Objeto 1



Objeto 2



Classe
Automovel

numeroPortas
cor
fabricante
ano
placa

Classe (3/3)

classe Pessoa



objeto Maria



objeto Pedro

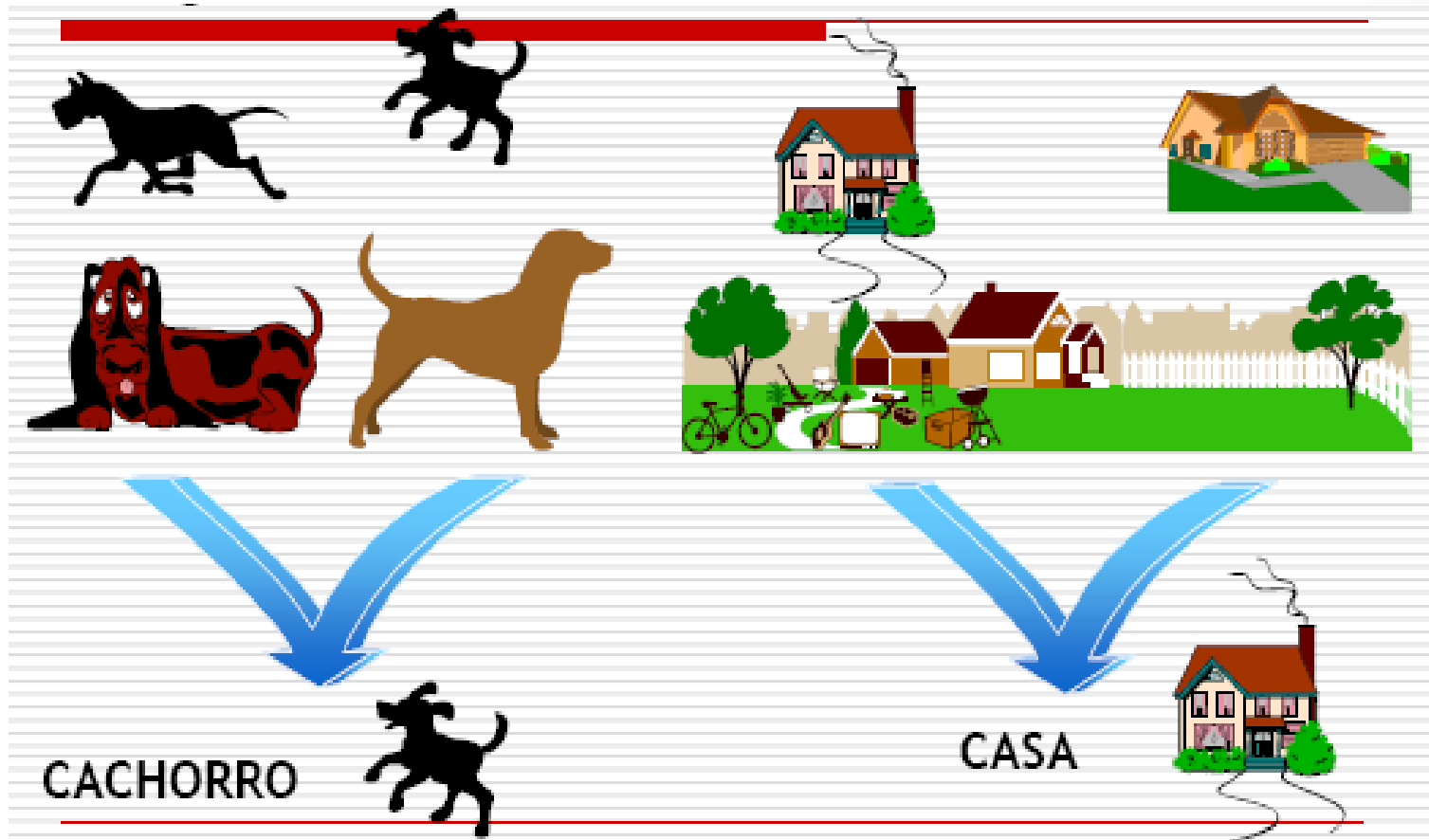
Classe
Pessoa

nome
idade
altura
peso
sexo

Objeto (1/3)

- Um objeto é uma construção de software que encapsula **estado** e **comportamento**, através respectivamente de **propriedades** (atributos) e **operações** (métodos);
- **Estado de um Objeto**: composto por suas **propriedades** e seus respectivos **valores**;
- **Comportamento**: a maneira como o objeto reage quando o seu estado é **alterado** ou quando uma mensagem é **recebida**.

Objeto (2/3)



Objeto (3/3)

☐ Um objeto possui operações:



- ☐ Ligar;
- ☐ Desligar;
- ☐ Travar portas;
- ☐ Acelerar;
- ☐ Frear...

Mensagens

- Mecanismo através do qual os objetos se **comunicam**, invocando as **operações desejadas**;
- Um objeto (**Emissor**) envia uma mensagem a outro (**Receptor**) que executará uma tarefa.

Os pilares da OO

- Os pilares da OO são mecanismos fundamentais que garantem a filosofia de Orientação a Objetos.
 - Encapsulamento;
 - Herança;
 - Polimorfismo.

Encapsulamento

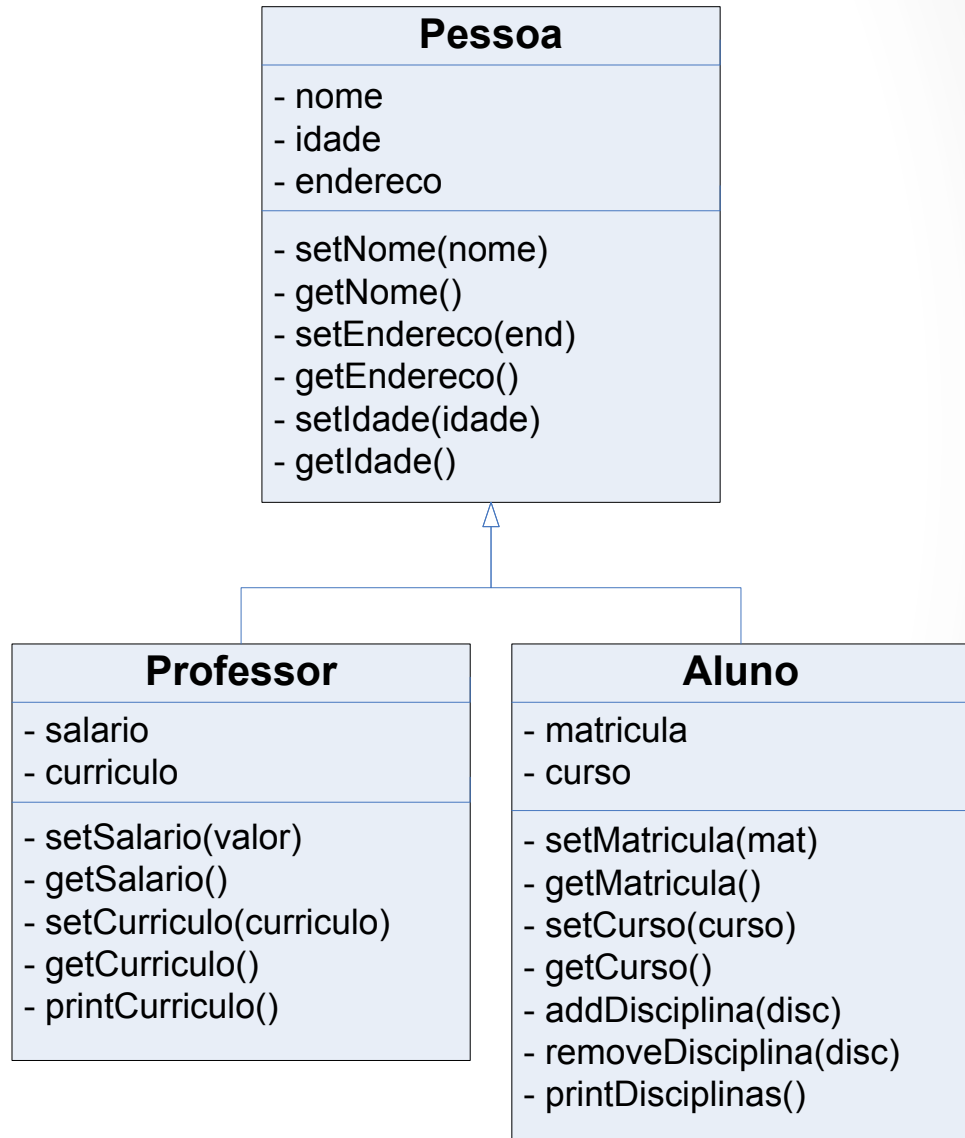
- **Resumindo:** “Não mostre as cartas de seu baralho”
- **Objetivos:**
 - Ocultar do mundo externo ao objeto os detalhes de implementação e restringir o acesso às propriedades e aos métodos;
 - Permitir a criação de programas com **menos erros** e **mais clareza**.
- **Vantagens:**
 - Segurança no acesso ao objeto;
 - Melhor consistência no estado interno, pois evita **alterações incorretas** nos valores das propriedades.

Herança (1/3)

- **Resumindo:** “Filho de peixe, peixe é” .
- Permite definir **novas classes** (subclasses) a partir de uma classe já existente (superclasse).
- A subclasse herda as **propriedades comuns** da superclasse e pode ainda **adicionar novos métodos** ou **reescrever métodos herdados**.
- **Objetivo:** evitar que classes que possuam atributos ou métodos **semelhantes** sejam **repetidamente criados**.

Herança (1/3)

- Pode ser: **Simples**



Herança (3/3)

- Pode ser: **Múltipla**



Polimorfismo

- **Resumindo:** “Vamos nos adaptar” .
- Permite que referências de **tipos de classes mais abstratas** representem o comportamento **das classes concretas que referenciam**.
- É caracterizado quando duas ou mais classes distintas tem métodos de mesmo nome, assim, uma função pode utilizar um objeto de qualquer uma das classes polimórficas, sem necessidade de tratar de forma diferenciada conforme a classe do objeto.

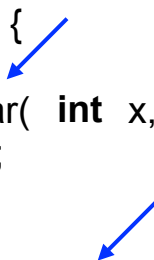
Tipos de Polimorfismo

- São dois os tipos de Polimorfismo:
 - Sobrescrita ou Redefinição de métodos (***Override***);
 - Sobrecarga de métodos (***Overload***).

Exemplo de Sobrecarga (*Overload*)

- Permite a existência de vários métodos de mesmo **nome**, porém com **assinaturas** levemente diferentes (número, tipo e qtd de parâmetros).

```
public class Soma {  
    public int somar( int x, int y) {  
        return x+y;  
    }  
    public double somar( double x, double y) {  
        return x+y;  
    }  
}
```



Exemplo de Sobrescrita (*Override*)

- Permite a existência de vários métodos com **assinaturas idênticas**, porém com implementações distintas.

```
01. public class Teste {  
02.  
03.     public void fazAlgo() {  
04.         System.out.println("Este é o método da super classe");  
05.     }  
06.  
07. }  
08.  
09. public class NovoTeste extends Teste {  
10.  
11.     @Override  
12.     public void fazAlgo() {  
13.         System.out.println("Este é o método foi sobrescrito");  
14.     }  
15.  
16. }
```