Programação Orientada a Objetos (PO24CP)

Aula #03 - Introdução ao Paradigma de Programação Orientado a Objetos

Profª Luciene de Oliveira Marin
lucienemarin@utfpr.edu.br

Introdução ao Paradigma de Programação Orientada a Objetos

Princípio básico: Abstração do problema

Retirar do domínio do problema detalhes relevantes e representá-los na linguagem da solução.

- A evolução das linguagens de programação influenciaram a forma de como os problemas são atacados
- A tecnologia existente em cada época delimitou como os problemas eram atacados e ao longo dos tempos surgiram diversos **paradigmas de programação**

Paradigma

Forma de como atacar um problema.

Paradigma Orientado a Objetos

Biologicamente inspirado

Surgiu da idéia que todo sistema de software funcionasse como um ser vivo

- Cada célula do sistema poderia interagir com outras células, através do envio de mensagens e cada célula consistiria ainda em um **sistema autônomo**

Todo o sistema é visualizado como um conjunto de células interconectadas, denominadas **objetos**. Cada objeto possui uma tarefa específica e através da comunicação entre os objetos é possível realizar uma tarefa computacional completa.

- **Tal paradigma é ideal para o desenvolvimento de softwares complexos**
  - Extensão do projeto de forma fácil e simplificada

Exemplos: Smalltalk, C++, Java, Python

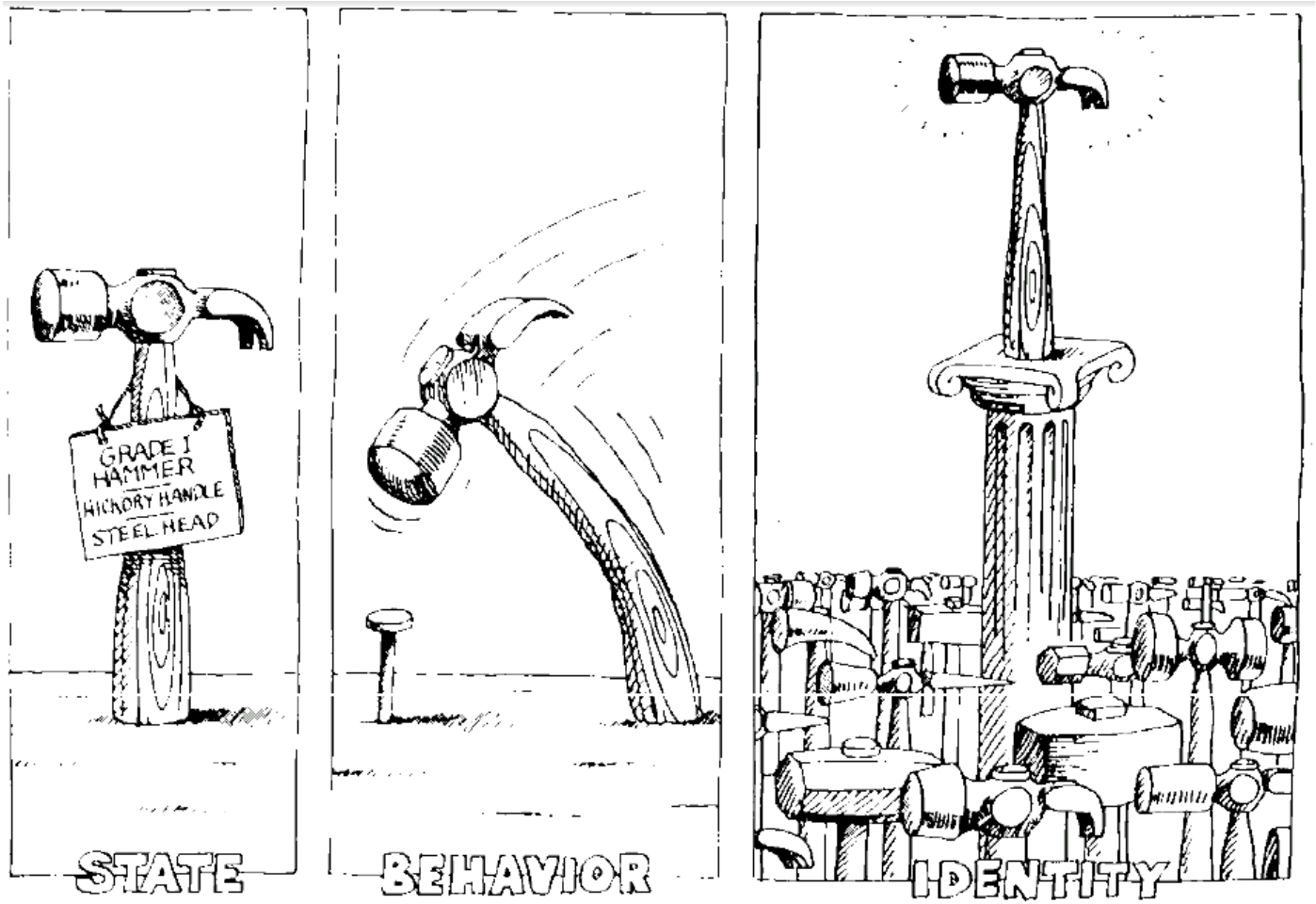
Conceitos da Orientação a Objetos

A Programação Orientada a Objetos fundamenta-se sobre 5 conceitos:

- Objetos
- Classes
- Mensagens
- Herança
- Polimorfismo

Objeto - definição

- Um **objeto** é um item **identificável** e é composto por **estado** e **comportamento**



Objeto - definição

Estado

- **O estado de um objeto representa as características deste**
- **Um carro possui como características uma cor, modelo, potência, velocidade atual, marcha atual, etc.**

Comportamento

- **Representa as funções (operações) que este objeto é capaz de executar**
- **Um carro pode trocar de marcha, acelerar, frear, etc.**

Regra de ouro da orientação a objetos

Identificar os **estados** e **comportamentos** de objetos do mundo real é um grande passo para se começar a pensar em termos de programação orientada a objetos

Objetos em sistemas computacionais

Objetos de *software* são semelhantes aos objetos reais

Um objeto armazena seu estado em **atributos** (ou variáveis) e seu comportamento se dá através de **operações** (funções ou métodos, depende da linguagem de programação).

- Em Java, os métodos de um objeto são invocados para realizar uma determinada computação e potencialmente para modificar os atributos deste objeto.

programador:	Qual a tua velocidade atual?
objeto:	20 km/hora
programador:	Diminua a velocidade em 10%
objeto:	Ok

Encapsulamento (1/5)

Um princípio importante do **paradigma de orientação a objetos**

Definição

Processo de esconder todos os detalhes de um objeto que não contribuem para as suas características essenciais.
Ex: uma caixa preta

- A **interação entre objetos** se dá através da **troca de mensagens**
- O **emissor da mensagem** não precisa conhecer como o **destinatário processará** a mensagem, ao emissor só importa receber a resposta
- Exemplo: `System.out.println("Ola mundo");`
  - Mensagens são compostas por três partes
    1. Objeto: `System.out`
    2. Nome do método: `println`
    3. Parâmetros: `"Ola mundo"`

Encapsulamento (2/5)

Princípio

O emissor das mensagens precisa conhecer quais operações o destinatário é capaz de realizar ou quais informações o destinatário pode fornecer

Interface de um objeto

corresponde ao que ele conhece e ao que ele sabe fazer, sem no entanto descrever como ele conhece ou faz

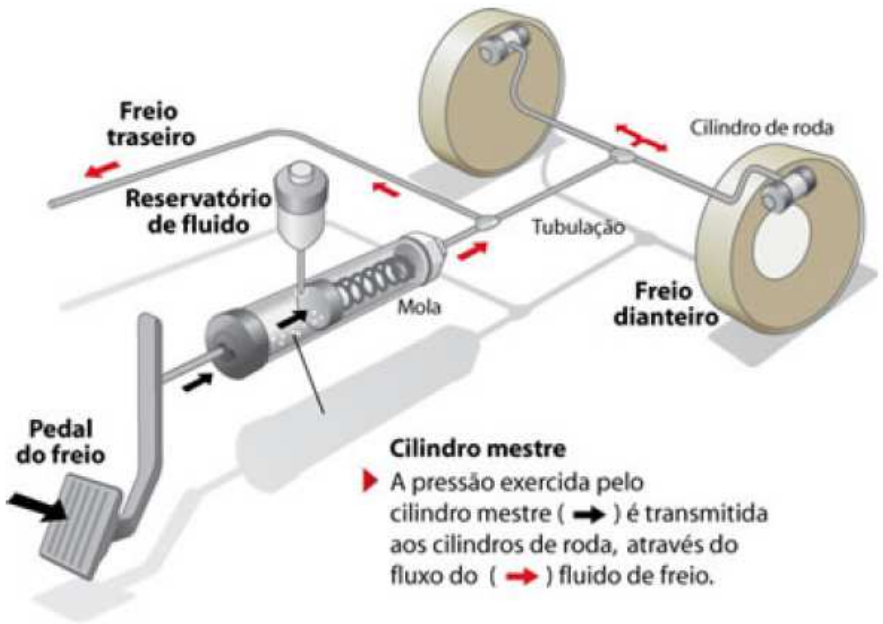
- Define as mensagens que ele está apto a receber e responder

Vantagem do encapsulamento

A implementação dentro de uma operação pode ser alterada sem que isso implique na alteração do código do objeto requisitante

Encapsulamento (3/5)

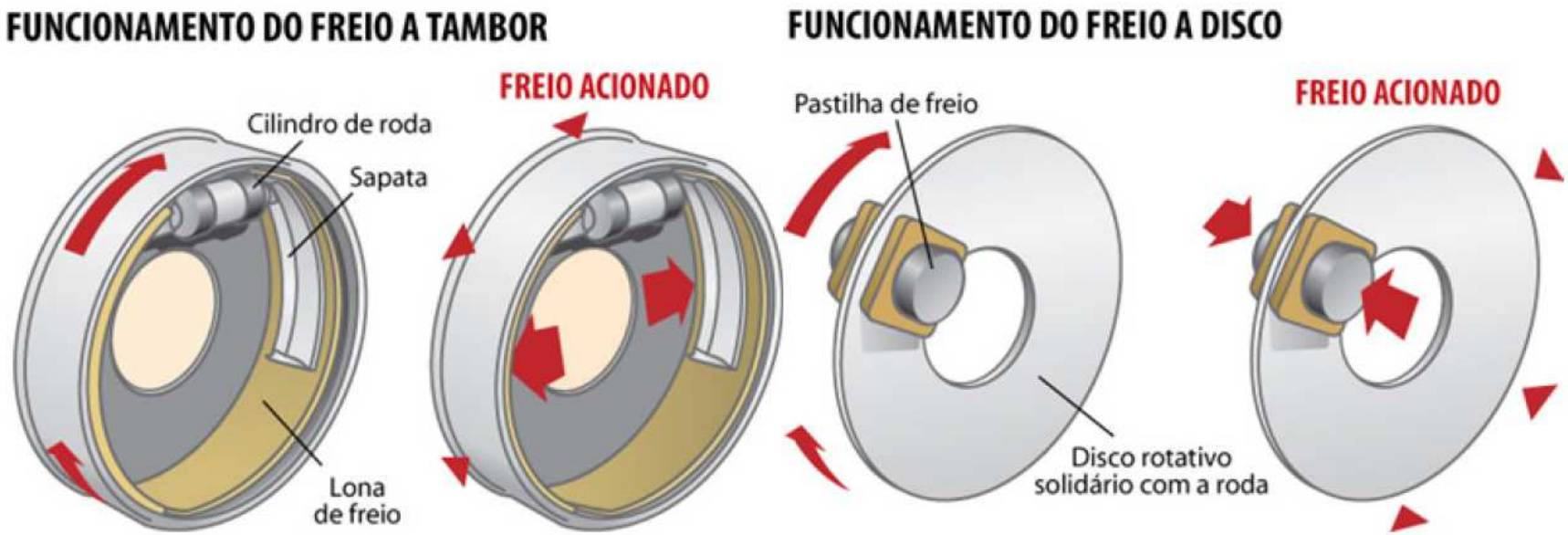
Exemplo do sistema de freio hidráulico:



- Freios funcionam através de um sistema de pistões e mangueiras por onde circula o fluído de freio
- Ao pisar no pedal de freio, aciona-se o cilindro mestre que irá pressurizar o fluído.
- Esse fluído transmite a pressão exercida no pedal até as rodas, acionando o freio.

Encapsulamento (4/5)

Exemplo do sistema de freio hidráulico:



- Como você faz para frear um carro com o sistema de freio a tambor?
- Como você faz para frear um carro com o sistema de freio a disco?

Encapsulamento - exemplo (5/5)

- Objeto: Fusca
  - Para diminuir velocidade do carro basta pressionar o pedal do freio
    - Fusca possui mecanismo de freio a tambor
  - Não é necessário entender como o mecanismo do freio funciona, mas ao acionar o freio, o Fusca irá diminuir sua velocidade

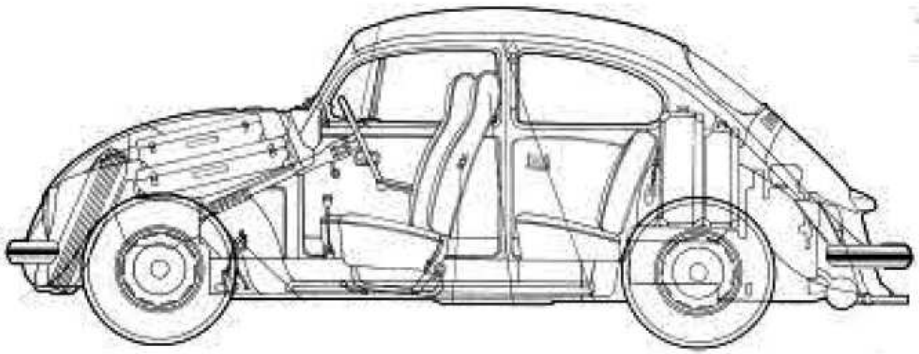
```
1 System.out.println("Acionando o freio do Fusca");
2 fusca.frear(10);
3 System.out.println("Acionando o freio da Ferrari");
4 ferrari.frear(10);
```

Classe - definição

é uma planta (projeto) que indica como os **objetos** deverão ser construídos

Exemplo: **Fusca**

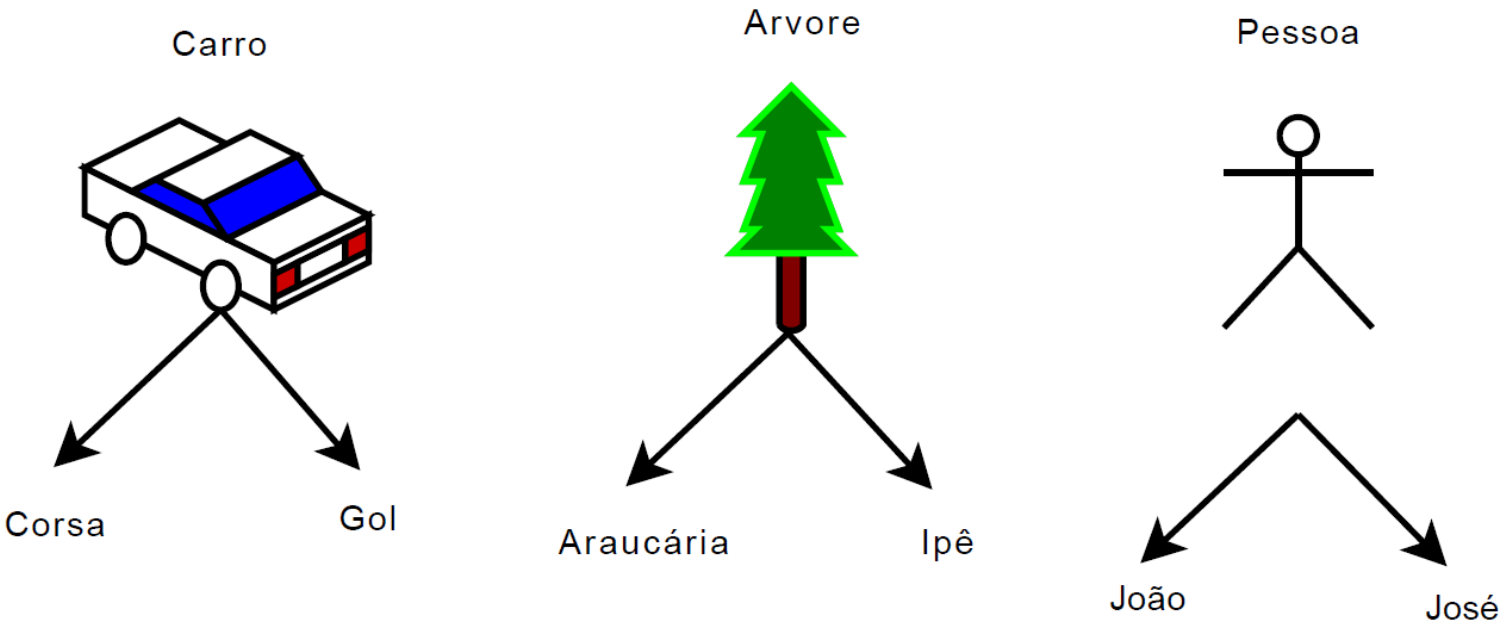
- Cada carro é construído com base em um mesmo projeto de engenharia e por consequência todos os carros possuirão os mesmos componentes



Classe

Objetos

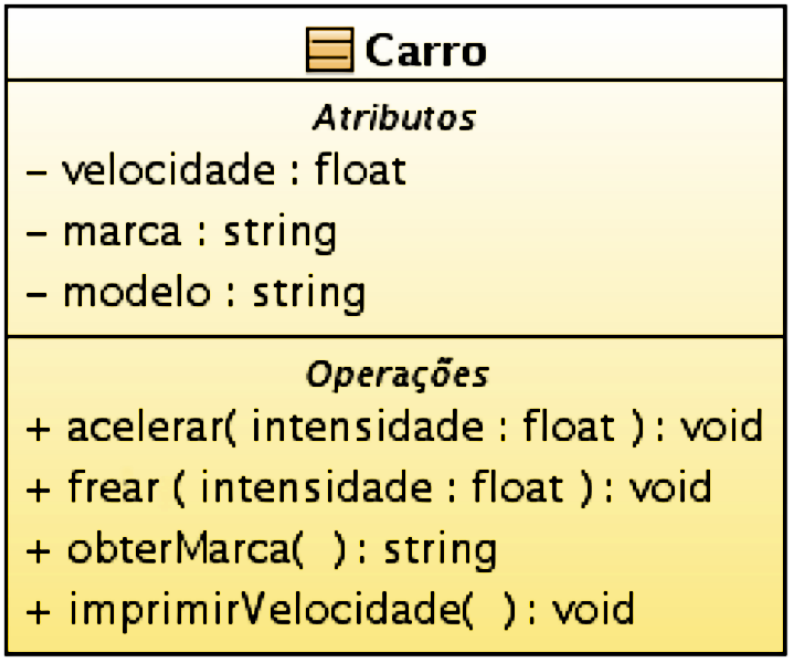
Classes e objetos



Uma classe em Java

```
5  public class Carro{
6   // atributos
7   private double velocidade;
8   private String marca;
9   private String modelo;
10  // metodos
11  public void acelerar(double intensidade){ ... }
12  public void frear(double intensidade){ ... }
13  public String obterMarca(){
14      return marca;
15  }
16  public void imprimirVelocidade(){
17      System.out.println("Velocidade: " + velocidade);
18  }
19 }
```

Representação gráfica em UML da classe `Carro`



Linguagem de modelagem unificada - UML

Uma linguagem **padrão** para a modelagem de sistemas, amplamente utilizada tanto pela indústria do software quanto por instituições acadêmicas.

Abstração

O que é?

Processo mental humano **de focar atenção aos aspectos mais relevantes** de alguma coisa, ao mesmo tempo **ignorar os aspectos menos importantes**

Para que serve?

Para gerenciar a complexidade de um objeto, tornando viável sua implementação.

Mas atenção!!

A **abstração é dependente do contexto** sobre o qual o objeto é analisado

- **O que é importante em um contexto pode não ser importante em outro**

Abstração

Exemplo - objeto Carro:

Revenda de carros

Necessita de um sistema para controlar os carros que possui.
Características essenciais:

- Atributos: código, marca, modelo, ano, preço
- Funções: obterCódigo, obterModelo, definirPreço, etc.

Jogo de Fórmula 1

Um usuário deseja controlar seu carro no jogo.
Características essenciais:

- Atributos: código, cor, equipe, velocidade máxima
- Funções: frear, acelerar, trocarPneus, etc.

Exemplos - modelando classes

Exemplo (1/3)

- Classe **Lampada**, seus atributos e operações.

Lampada
- estadoDaLâmpada
- acende() - apaga() - mostraEstado()

Exemplo (2/3)

- Classe **Data**, seus atributos e operações.

Data
- dia - mes - ano
- inicializaData() - dataÉVálida() - mostraData()

Exemplo (3/3)

- Classe **RegistroAcademico**, seus atributos e operações.

RegistroAcademico
- nomeDoAluno - numeroDeMatricula - dataDeNascimento - éBolsilta - anoDeIngresso
- inicializaRegistro(nome, matrícula, data, bolsa, ano) - calculaMensalidade() - mostraRegistro()

Exercícios de fixação (APS 01)

- Responda ao que se pede no moodle da disciplina.