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

Paradigmas de Programação

- Forma de como abstrair a solução de um problema computacional

- Programação Estruturada
- Programação Funcional
- **Programação a Objetos**

Programação Estruturada:

- Principais características:
  - Possuem 3 estruturas básicas: **sequência, decisão e iteração**;
  - Programação orientada a procedimentos (subprogramas, sub-rotinas e funções), que se referem a blocos estruturados de códigos;
  - A comunicação entre os blocos se faz através de variáveis globais ou passagem de parâmetros;
  - Recomenda-se modularizar funções grandes em funções menores;
  - Linguagens: C, Pascal, Fortran.

Programação Funcional:

- Principais características:
  - Não existe declaração de variáveis, somente funções.
  - As operações consistem em composição de funções e o uso de recursividade em processos de repetição.
  - Linguagens: Lisp, Haskell

Programação Orientada a Objetos (POO)

- Principais características:
  - Dados e funções que manipulam os mesmos encapsulados em um mesmo elemento denominado **objeto**.
  - A comunicação entre objetos é feita pelo envio e recebimento de mensagens.
  - Projetar uma estrutura ou hierarquia de classes que descrevem objetos para realizar uma tarefa computacional.
  - Linguagens: Smalltalk (puramente OO), Java, C++, C#, Delphi, Ruby, Python.

Vantagens da POO

- Principais características:
  - Reaproveitamento de código
  - Facilidade de manutenção
  - Maior confiabilidade no código
  - Maior facilidade de gerenciamento
  - Maior Robustez

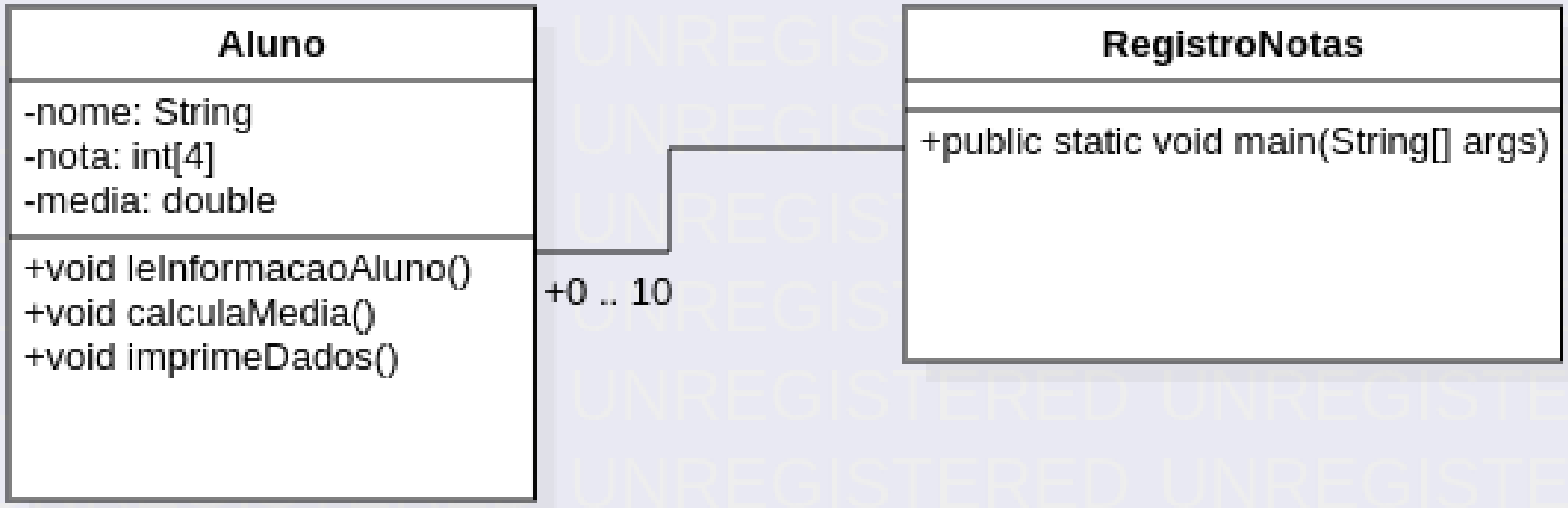
Exemplo:

Escreva um programa para registrar as 4 notas de cada um de 10 alunos em uma disciplina e calcular a média de cada um.

Programação Estruturada (C):

```
int notas[10][4];
int medias[10];
void leNotas(){...}
void calculaMedia(){...}
void imprimeMedia(){...}
void main(){...}
```

Programação Orientada a Objetos:



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

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

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

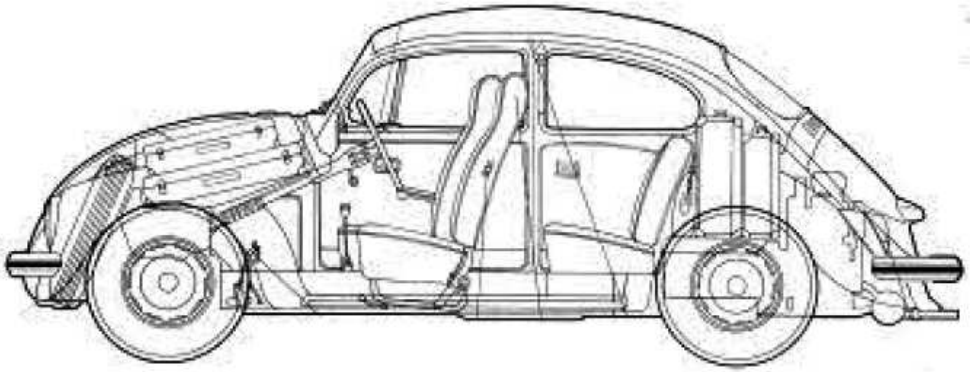
A implementação dentro de uma operação pode ser alterada sem que isso implique na alteração do código do objeto requisitante

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



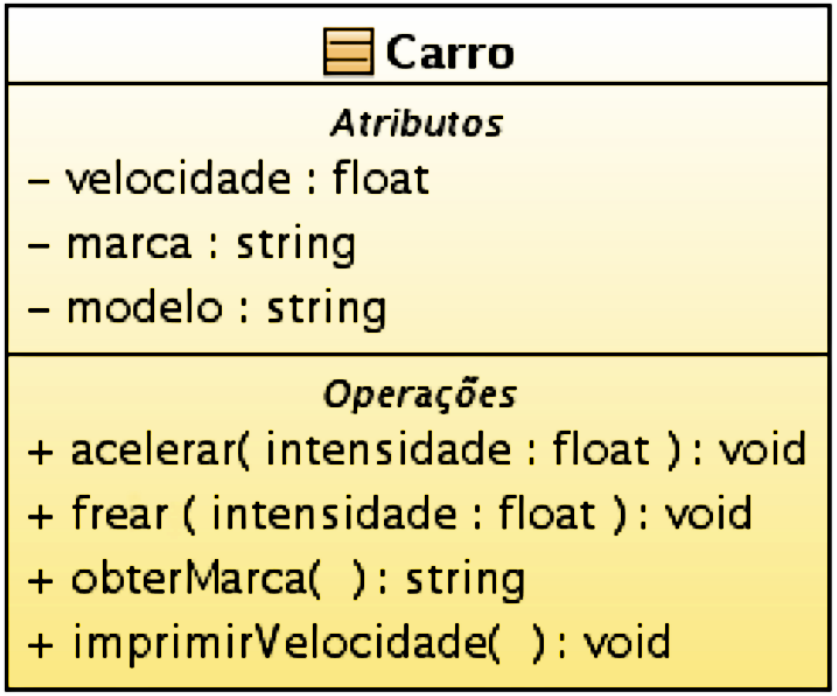
Classe

Objetos

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**:

contexto 1: 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.

contexto 2: Jogo de Fórmula 1

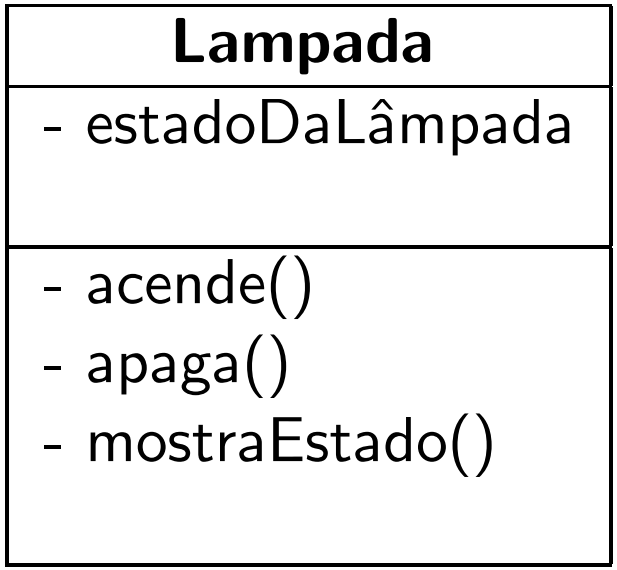
Um usuário deseja controlar seu carro no jogo.
Características essenciais:

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

Exemplos - modelando classes

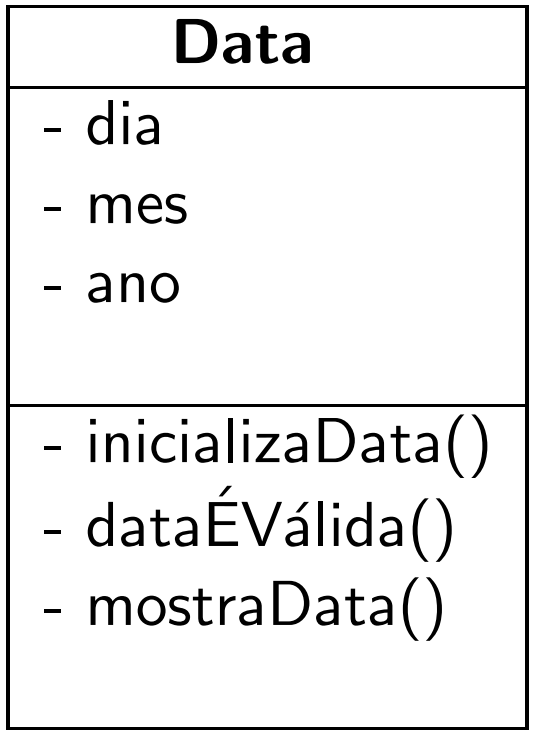
Exemplo (1/3)

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



Exemplo (2/3)

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



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 [LE-01 - Aula 03]

- Responda ao que se pede no moodle da disciplina.