

**FIAP GRADUAÇÃO**

# DOMAIN DRIVEN DESIGN

Prof. Me. Thiago T. I. Yamamoto

#01 - ORIENTAÇÃO A OBJETOS

# TRAJETÓRIA

---



Orientação a Objetos



# #01 - AGENDA

---



- Apresentação da disciplina
- Orientação a Objetos
- Classes
  - Atributos e Comportamentos
- Relação de classes com objetos
- Modelo Visual



## THIAGO T. I. YAMAMOTO

COORDENADOR E PROFESSOR

+ de 20 anos de experiência em TI e 16 anos de atuação na FIAP como professor de Graduação e MBA. Atualmente, coordenador de 3 cursos do MBA, liderando a evolução curricular e a integração de novas tecnologias ao ensino. Experiência no desenvolvimento e na docência de disciplinas em Programação, Desenvolvimento de Sistemas e APIs, além da atuação como Scrum Master em cursos de Graduação.

- Coordenador dos MBAs na FIAP (Eng. Software, BPM e DevOps)
- Scrum Master dos cursos de Análise e Desenvolvimento de Sistemas e Engenharia de Software ON.
- Professor de Graduação e MBA FIAP.
- Mestre em Ciências pela Universidade Federal de São Paulo.
- Pós-graduado em Engenharia de Software.
- Bacharel em Ciência da Computação pela Unesp/Bauru.
- Certificado em Programação Java, ITIL e PSM I.

Apaixonado por ensino e tecnologia, pai de três pets e entusiasta de esportes como Musculação, Mergulho e Beach Tennis.



<https://www.linkedin.com/in/thiagoyamamoto/>



# DOMAIN DRIVEN DESIGN

- Preparar o profissional para o **Mercado de Trabalho**;
- Aprender os conceitos de **Programação Orientada a Objetos** com a **linguagem Java**;
- Desenvolver *back end* de aplicações **Java**;



- Programação orientada à objetos;
- Plataforma Java;
- Ambiente de execução: Compile/Runtime;
- Detalhes da linguagem:
  - Tipos primitivos, operadores, controle de fluxo, loops e etc.;
- Java Beans;
- Arrays/Collections Framework;
- Tratamento de Erros;
- JDBC;
- Design Patterns;

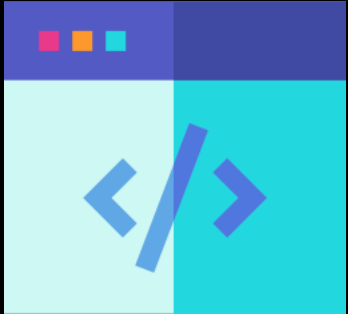




# METODOLOGIA

---

- Aulas “Hands On”;
- Projetos no **Github**;
  - <http://www.github.com/thiagoyama>
  - **Git** é um **versionador de arquivos**, será abordado em outra disciplina;

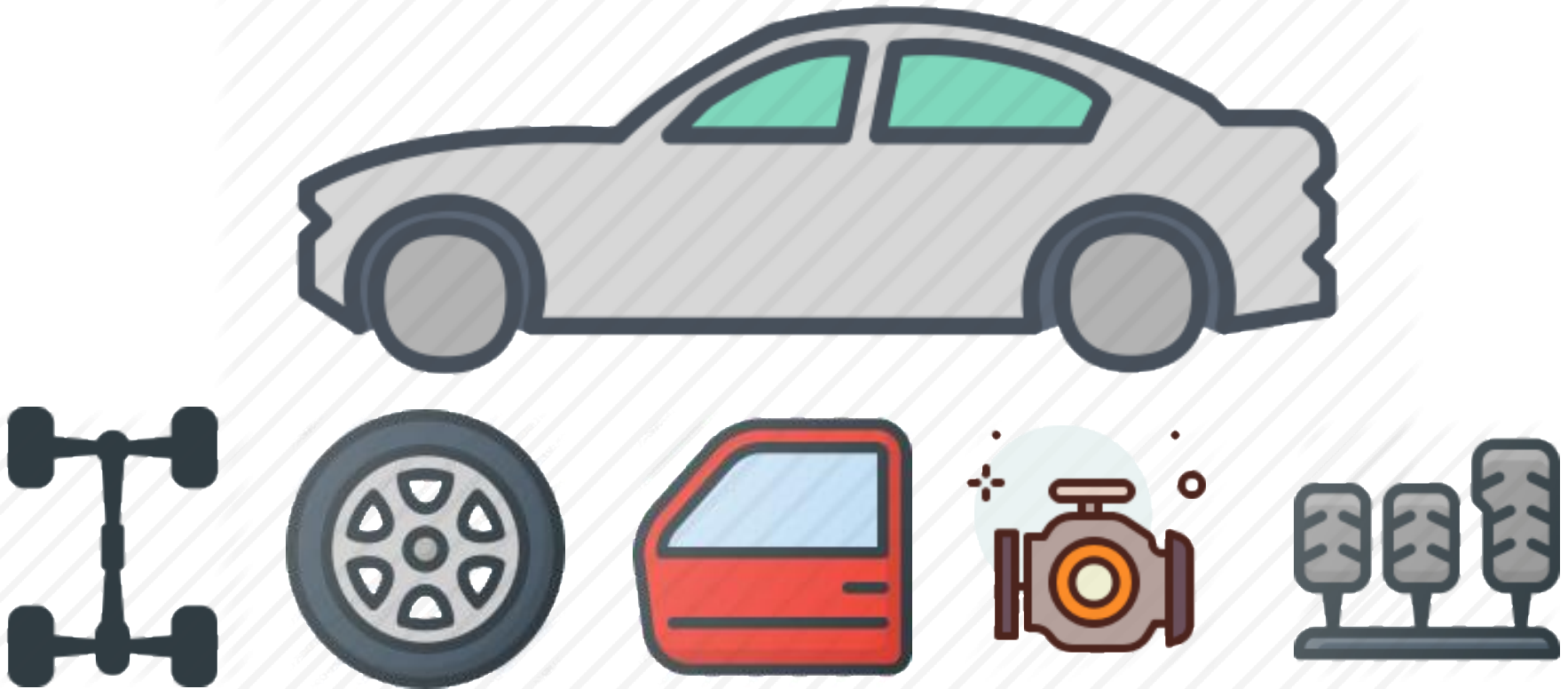


*“Quem ouve, esquece. Quem vê, lembra. Quem faz, aprende.”*  
Provérbio chinês.



# ORIENTAÇÃO A OBJETOS

- O que é Programação Orientada a Objetos?



- **Programação**
  - Uma **linguagem de programação** é um método padronizado para **expressar instruções** para um computador;
  - É um conjunto de **regras sintáticas** (gramatical) e **semânticas** (significado) usadas para definir um programa de computador;
- **Objeto**
  - Um objeto representa uma **entidade** que pode ser **física, conceitual** ou de **software**;
- **Programação Orientada a Objetos**
  - É um paradigma de **análise, projeto e programação** de sistemas de informação, baseado na composição e interação entre diversas unidades de software chamadas de **objetos**;

- **Cadeira** (material, cor, tem braço?, tem rodas?, etc.)
  - **Praia** (alumínio, pano, de deitar, etc.);
  - **Escritório** (ferro, estofado macio, preta, com braço, etc.);
  - **Rodas** (ferro, branca, com rodas, automática, manual, etc.);
  - **Banco**
    - Carro (couro, preto, regulável, etc.);
    - Praça (tijolo, verde, com encosto, etc.);
- **Bola** (material, formato, cor, etc.)
  - **Futebol** (couro, redonda, 40cm de diâmetro, branca, etc.);
  - **Tênis** (tecido, redonda, 5cm de diâmetro, amarela, etc.);
  - **Ping-Pong** (pvc, redonda, 1.5cm de diâmetro, branca, etc.);
  - **Futebol Americano** (couro, oval, 50cm de largura, marron, etc.);



- Bola (material, formato, cor, etc.)

Características ou Propriedades



Futebol

Material: couro;  
Formato: redondo;  
Tamanho: 40cm;  
Cor: Branca;



Tênis

Material: tecido;  
Formato: redondo;  
Tamanho: 5cm;  
Cor: Amarela;



Futebol Americano

Material: couro;  
Formato: oval;  
Tamanho: 50cm;  
Cor: Marron;



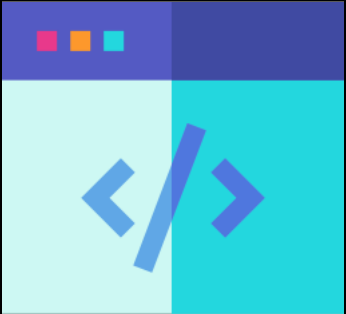
Basquete

Material: couro;  
Formato: redondo;  
Tamanho: 60cm;  
Cor: Laranja;

# PRÁTICA

---

- Busque ao menos 3 exemplos do cotidiano;
- Descreva as **propriedades** dos exemplos encontrados;



- Sistema de Caixa Eletrônico

- Objeto: Cliente

- Nome
- Endereço
- CPF
- RG



- Objeto: Conta Corrente

- Agência
- Número
- Saldo
- Cliente





- Sistema de E-Commerce

- Objeto: **Produto**
  - Nome
  - Descrição
  - Valor
- Objeto: **Estoque**
  - **Produto** ←
  - Quantidade
  - Prazo de Validade
- Objeto: **Cliente**
  - Cadastro
  - Senha do Cadastro
  - Nome
  - Endereço
  - CPF
  - RG





# CLASSES

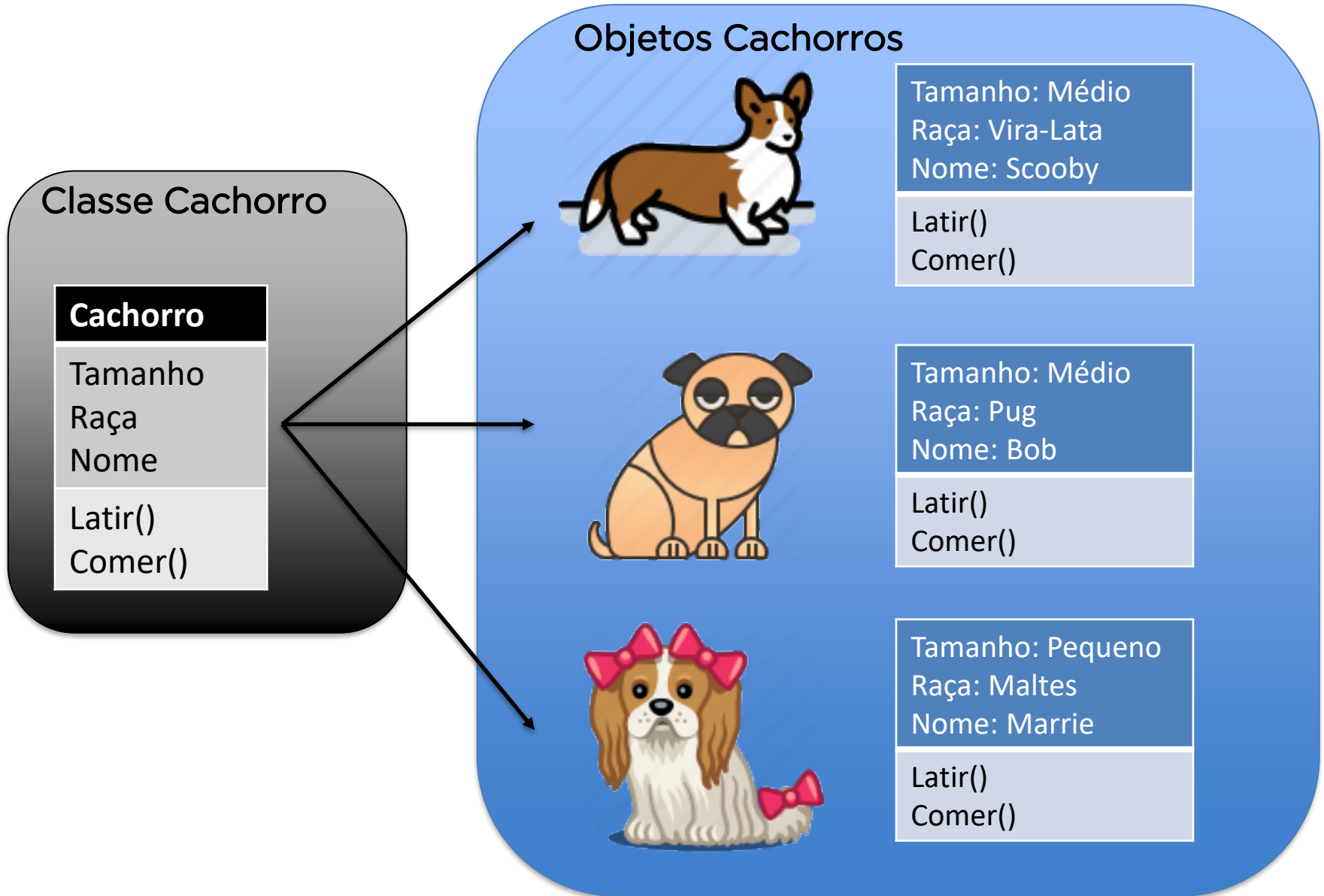
- As **abstrações** são representadas pelas **classes**;
- Uma classe deve **conter apenas os elementos necessários** para resolver um aspecto bem definido do sistema;
- A **classe é uma descrição nomeada** para um grupo de entidades (chamadas de **objetos** ou instâncias de classe) **que têm as mesmas características**;

Cachorro
Tamanho
Raça
Nome
Latir()

*Uma classe cachorro, vários objetos*

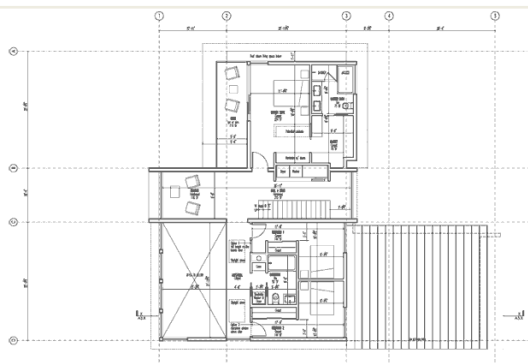


- Estas características são os **atributos** (propriedades, campos de dados) e as **operações** (comportamentos, métodos, funções) que podem ser executadas nestes objetos;
- Em outros termos, uma classe descreve os **serviços providos por seus objetos** e **quais informações** eles podem armazenar;
- Na programação orientada a objetos a classe é a **unidade básica de programação**;
- Todos os **programas são escritos como um conjunto de classes**, e todos os códigos que você escrever devem fazer parte de uma classe;



# O QUE É UMA CLASSE?

- Uma classe é a descrição de um conjunto de objetos que compartilham os mesmos *atributos, operações, relações, e semânticas*;
  - *Um objeto é uma instância de uma classe*;
- Uma classe é uma abstração, uma vez que:
  - Enfatiza características relevantes;
  - Suprime outras características;



Classe



Instanciação



Objeto

## Classe Curso

### Propriedades

Nome

Local

Dias oferecidos

Carga horária

Hora de Início

Hora de Término



### Comportamentos

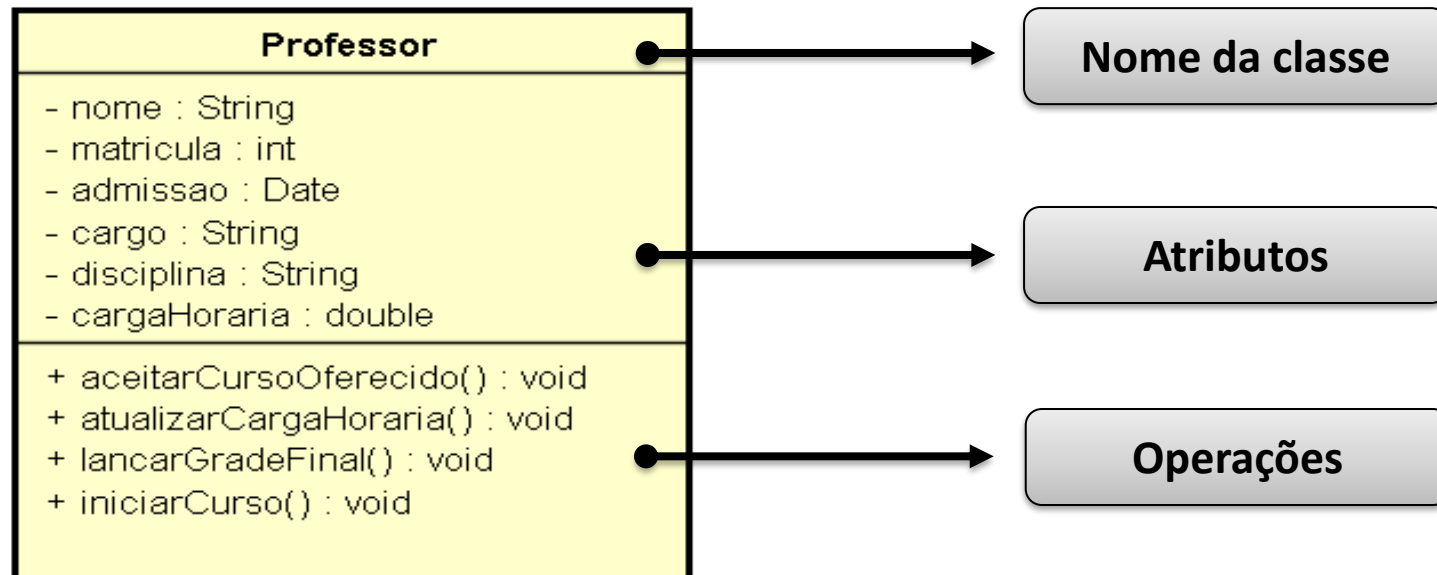
Adicionar um aluno

Excluir um aluno

Obter lista de alunos

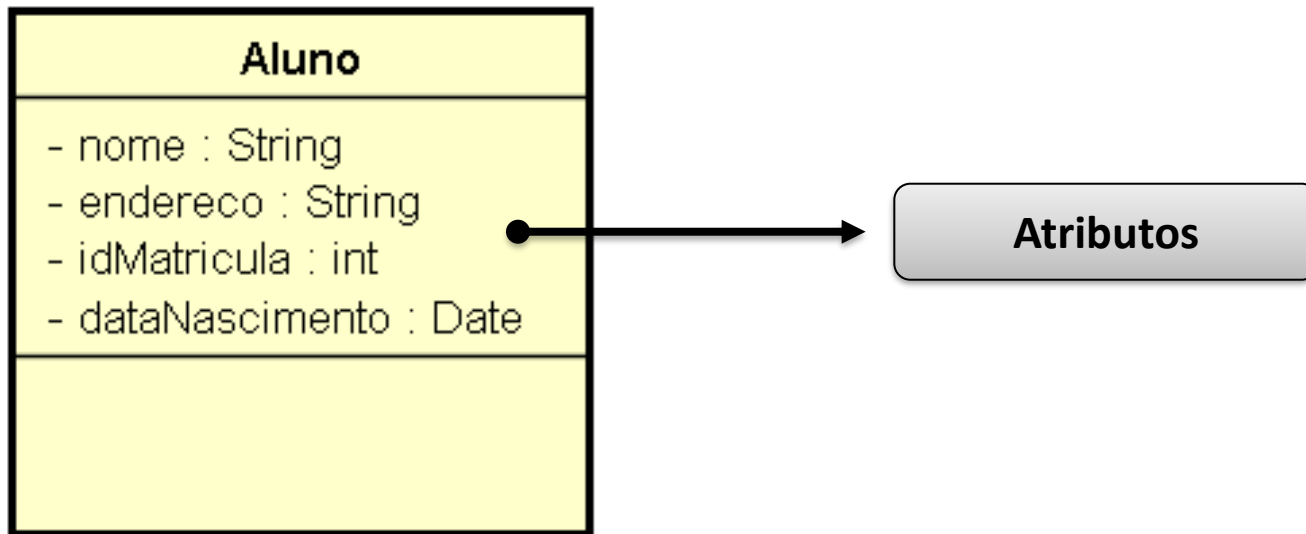
Verificar se está cheio

- É possível representar **graficamente** uma classe através de um **diagrama de classes** (UML), este diagrama é uma representação da **estrutura e relações das classes** que servem de modelo para objetos;
- Uma classe é representada através de um **retângulo com três compartimentos**;

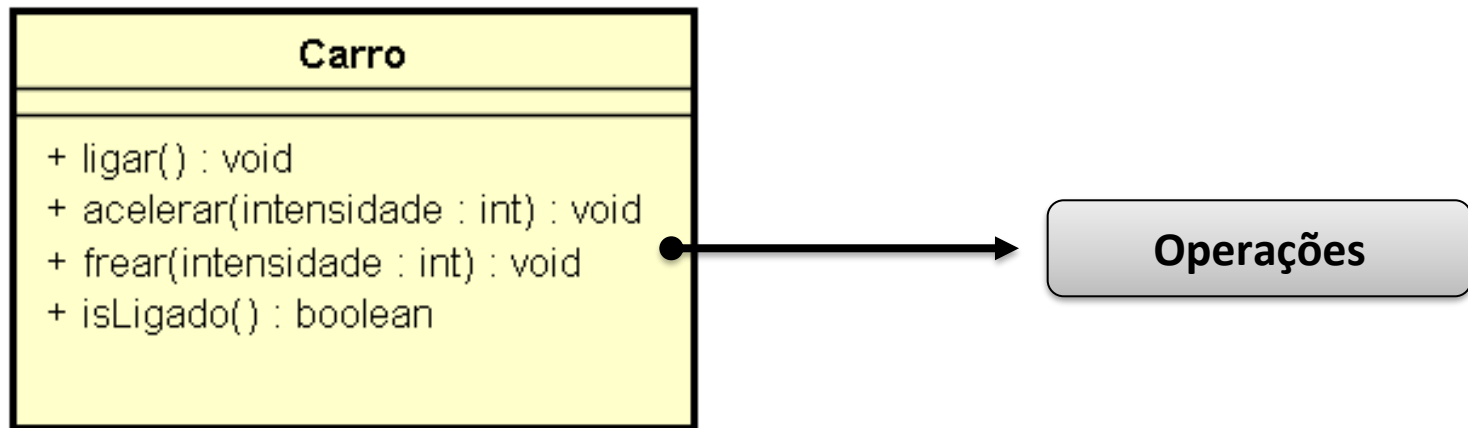




- Um **atributo** é o **nome** que se dá à **propriedade** de uma classe;
- O **atributo** **descreve** o **tipo** de **valores** que a propriedade possui;
  - Um classe pode ter **qualquer número de atributos** ou nenhum atributo;

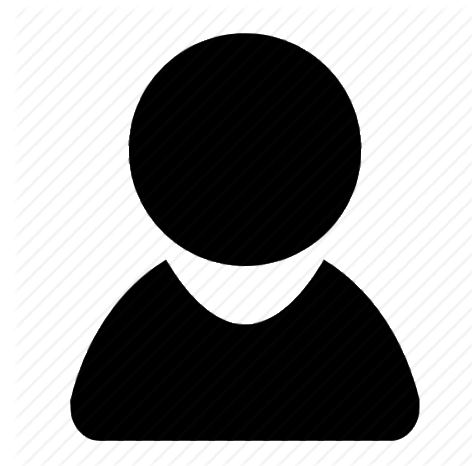


- **Operação é um serviço** que pode ser solicitado a partir de um objeto para **executar de comportamento**. Uma operação tem uma assinatura, que pode restringir os parâmetros reais que são possíveis;
- Um **classe** pode ter qualquer número de operações ou nenhuma operação





- A classe **Pessoa** possui os seguintes atributos e operações:

Pessoa
<ul style="list-style-type: none"><li>- nome : String</li><li>- sexo : String</li><li>- idade : int</li><li>- casa : Casa</li><li>- carro : Carro</li></ul>
<ul style="list-style-type: none"><li>+ exibirDadosPessoais() : void</li><li>+ exibirPatrimonio() : void</li></ul>



- A classe **Pessoa** pode gerar vários objetos:


## Objeto #1

nome = "Pedro"  
idade = 52  
sexo = "Masculino"  
casa =   
carro = 

exibirDadosPessoais()  
exibirPatrimonio()



## Objeto #2

nome = "Telma"  
idade = 25  
sexo = "Feminino"  
casa =   
carro = 

exibirDadosPessoais()  
exibirPatrimonio()



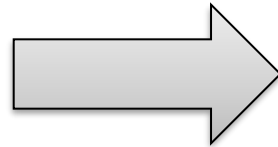
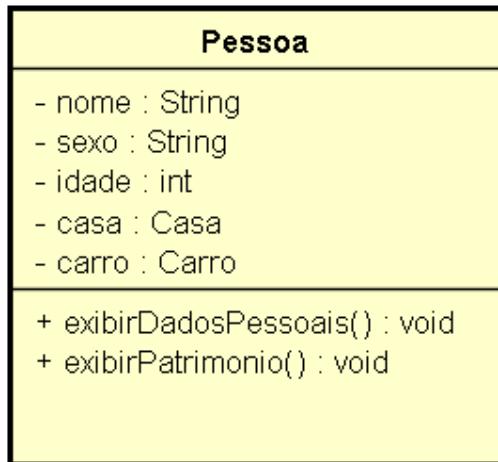
## Objeto #3

nome = "Julio"  
idade = 20  
sexo = "Masculino"  
casa =   
carro =

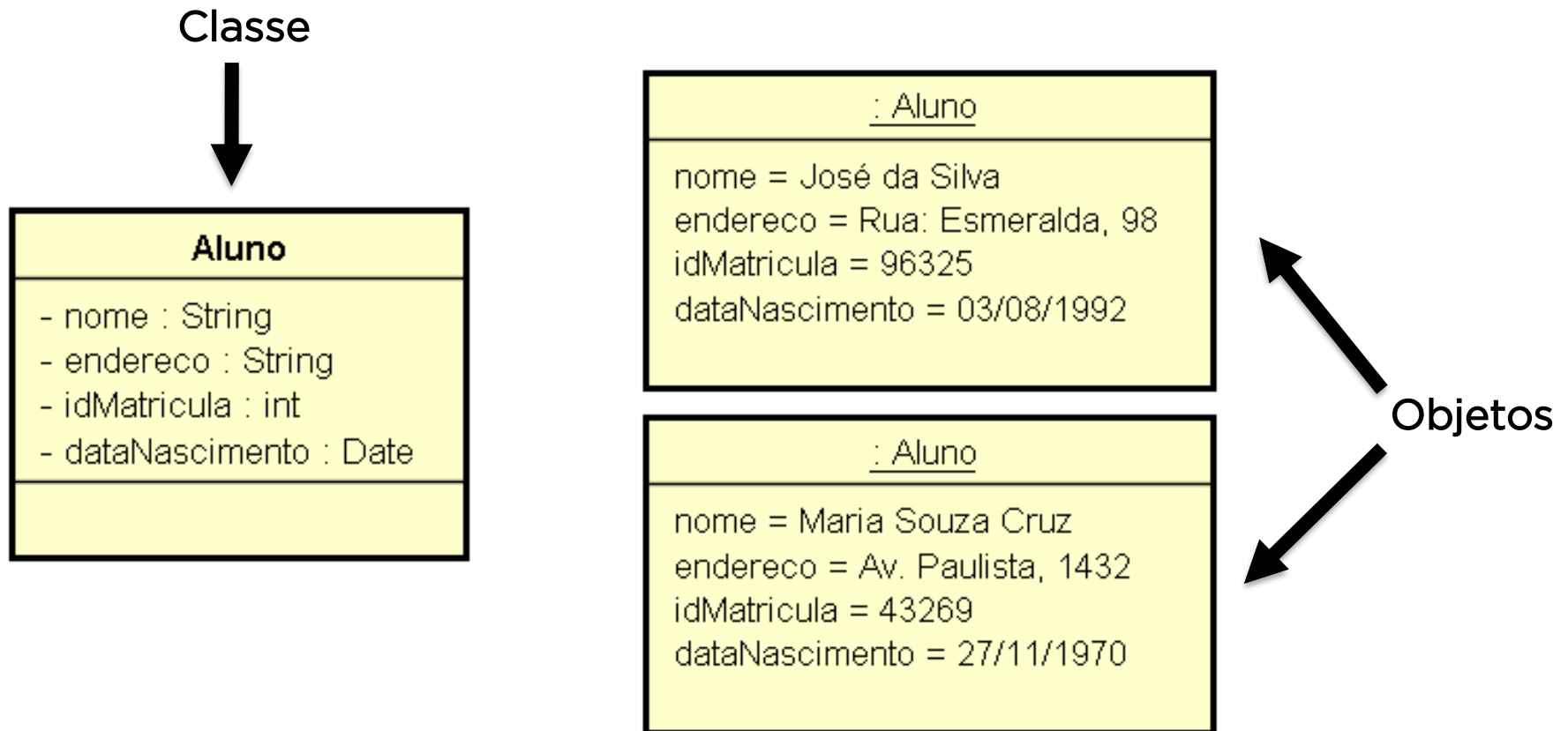
exibirDadosPessoais()  
exibirPatrimonio()



- Uma classe é uma definição abstrata de um objeto;
  - Ela define a **estrutura** e **comportamento** de cada objeto da classe;
  - Ela serve como um **modelo** para a **criação de objetos**;
- Classes **não** são coleções de objetos;



- Atributos em classes e objetos:



- Um **conjunto de princípios** (abstração, encapsulamento, herança e polimorfismo) **guiando a construção do software**, em conjunto com linguagens, bancos de dados e outras ferramentas que suportam esses princípios. (*Object Technology - A Manager's Guide, Taylor, 1997.*)
- **Vantagens** da orientação a objetos:
  - Facilidades arquiteturais e reuso de código;
  - Reflete em modelos de mundo real;
  - Incentiva a estabilidade;
  - É adaptável à mudanças;



# PRÁTICA 1

---



- Represente graficamente as classes de forma que elas que abstraíam:
  - Um candidato - no contexto de uma agência de empregos;
  - Um médico - no contexto de um hospital;
  - Um piloto - no contexto de uma corrida de Fórmula 1;
- Apresente de forma gráfica uma **instância (objeto)** de cada classe definida anteriormente;



# Copyright © 2020 - 2025 Prof. Thiago T. I. Yamamoto

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).

*“Se você traçar metas absurdamente altas e falhar, seu fracasso será muito melhor que o sucesso de todos” – James Cameron*