Fundamentos da orientação a objetos

Analise e Modelagem de Sistemas

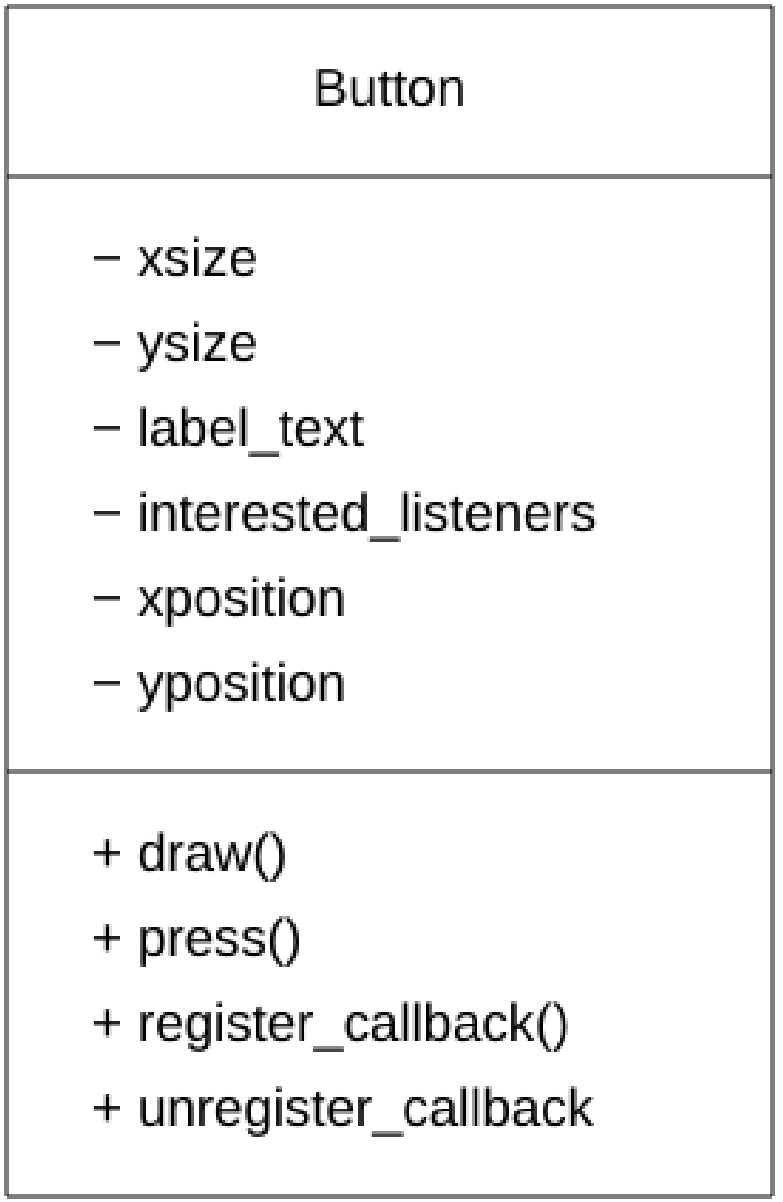
Eduardo Furlan Miranda

2025-10-12

Adaptado de: WERLICH, C. Análise e Modelagem de Sistemas. Londrina: EDE SA, 2020. ISBN 978-85-522-1683-4.

Orientação a objetos (OO)

- Paradigma baseado em objetos
- Um objeto é uma entidade que encapsula dados e funções
- Ex. de notação UML para uma classe:



OO

- 1966: linguagem de programação SIMULA
- Ex. em Python:

```
class Carro:
  def __init__(self, modelo):
    self.modelo = modelo
  def buzinar(self):
    print(f"{self.modelo}: Bi-bi!")

meu_carro = Carro("Fusca")
print(meu_carro.modelo)  # Acessa o atributo
meu_carro.buzinar()      # Chama o método
```

UML (1997)

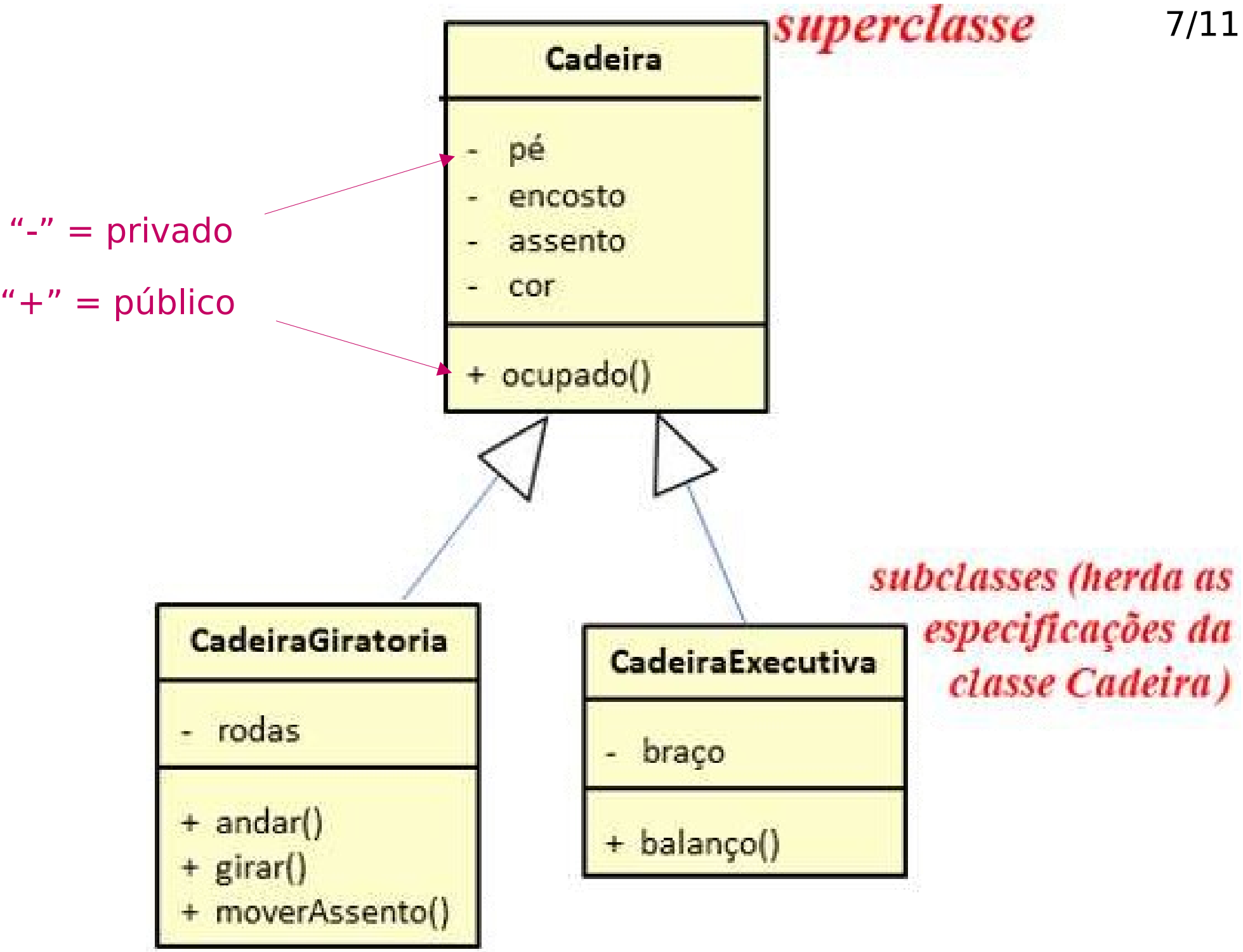
- Linguagem de modelagem de uso geral
  - Visual (usa coleção de modelos gráficos)
  - Orientada a objetos
- Padrão OMG, ISO/IEC 19501
- Usado em várias áreas além da computação
- Modela sistemas, independente de linguagens de programação
- Um modelo UML pode ser usado para gerar código fonte

OO

- Abstração como ideia central
- Classe: representação da abstração (“projeto”)
  - Representa um conjunto de objetos
    - Características (atributos) e comportamentos (métodos)
- Características: atributos (variáveis) e métodos (funções)
- Objeto: instância da classe, “prédio construído”
  - São únicos
  - Comportamentos: valores dos atributos e ações

OO

- Herança
  - Novas classes reaproveitando o que já existe
- Superclasse e subclasses
  - A subclasse herda da superclasse, e pode acrescentar
- Encapsulamento
  - Esconde os detalhes da implementação
    - Ex.: não ter acesso direto às variáveis ou funções dentro do objeto


superclasse
Cadeira
- pé
- encosto
- assento
- cor
+ ocupado()
"-" = privado
"+" = público
CadeiraGiratoria
- rodas
+ andar()
+ girar()
+ moverAssento()
CadeiraExecutiva
- braço
+ balanço()
subclasses (herda as especificações da classe Cadeira)

OO - Polimorfismo estático

- O mesmo método é implementado várias vezes na mesma classe, com parâmetros diferentes
  - A escolha do método a ser chamado vai variar de acordo com o parâmetro passado

- Ex. Java:

```
class Calculadora {
    int somar(int a, int b) { return a + b; }
    int somar(int a, int b, int c) { return a + b + c; }
}
```

Obs.: Python não tem

OO - Polimorfismo dinâmico

- O mesmo método é implementado várias vezes nas subclasses derivadas, com os mesmos parâmetros
  - A escolha do método depende do objeto que o chama
    - E, consequentemente, da classe que o implementa

```
class Cachorro:
    def falar(self):
        print("Au au!")

class Gato:
    def falar(self):
        print("Miau!")

animais = [Cachorro(), Gato()]
for animal in animais:
    animal.falar()
```

- POO - Programação Orientada a Objeto
- A/POO - Análise e Projeto Orientado a Objeto
  - Aplica os conceitos de OO na análise e na elaboração do projeto, que são fases que antecedem a programação
- “Quando construídos corretamente, sistemas orientados a objetos são flexíveis a mudanças, possuem estruturas bem conhecidas e proveem a oportunidade de criar e implementar componentes com alto grau de reutilização”

OO - vantagens

- Reutilização de código
- Utilização de um único padrão conceitual para a análise, o projeto e a implementação
- O tempo de desenvolvimento do software é mais rápido
- A construção de sistemas mais complexos é simplificado pelo fato de cada objeto ser simples, fácil de testar e integrar ao demais objetos
- Pode ter os custos de desenvolvimento reduzidos