Programação Orientada a Objeto (Python e Java)

Rafael Alves da Costa

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
FATEC Carapicuíba

Aula 3 - Estrutura de dados

08/2025

Sumário

História da Programação Orientada a Objetos (POO)

História da Programação Orientada a Objetos

- **Alan Kay disse:** "I invented the term Object-Oriented, and I can tell you I did not have C++ in mind." (Eu inventei o termo orientado a objeto, e posso dizer que não tinha C++ em mente.)
- **Simula** passou a ser usada com frequência da década de 70, sendo inclusive a linguagem utilizada para desenvolver as primeiras versões de Smalltalk.
- Programar nesta linguagem deveria ser tão fácil quanto bater um papo furado (small talk) em inglês.
- Referência: Daniel Cordeiro

Alan Kay holding the prototype of the Dynabook

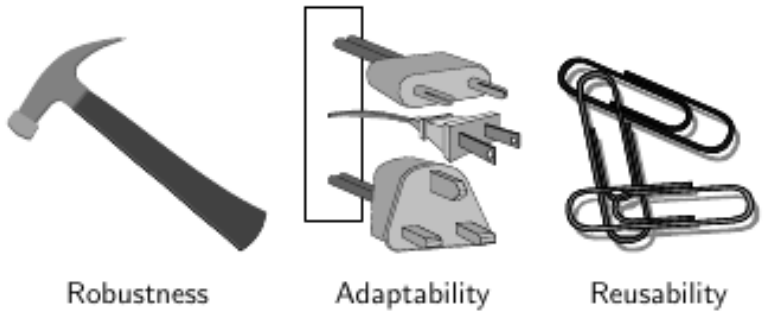
Kay (1940-)

Fonte: en.wikipedia.org

Introdução à POO

POO

- Objetos como atores principais
- Cada objeto é uma instância de uma classe
- Classes definem variáveis de instância e métodos
- Visão concisa e consistente dos objetos
- **Objetivos de um projeto Orientado a Objetos**
  - Robustez
  - Adaptabilidade
  - Reutilização



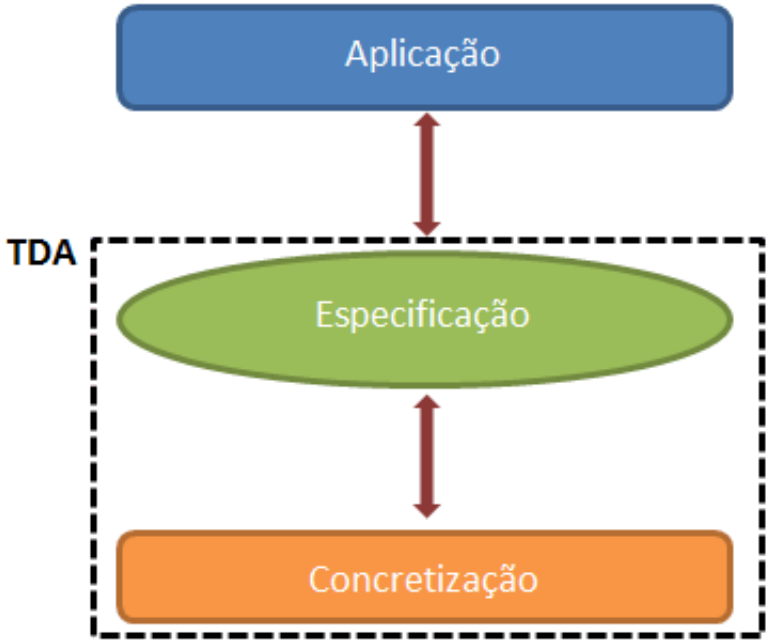
Objetivos de um projeto OO

Fonte: Data structures and algorithms in Python[1].

[1] Goodrich, M. T., Tamassia, R., Goldwasser, M. H. (2013). Data structures and algorithms in Python.

Princípios de Projetos OO

- Modularidade
  - Divisão em unidades funcionais separadas
  - Claridade na organização e interação dos componentes
  - Exemplo: Módulos em Python[1]
- Abstração
  - Destilação de sistemas complexos em partes fundamentais
  - Tipos Abstratos de Dados (TDAs)



  - Especificação do que cada operação faz, mas não como[1]

Princípios de Projetos OO

- Encapsulamento
  - Esconder detalhes internos de implementação
  - Liberdade para alterar implementações sem afetar outros componentes
  - Convenções em Python para membros não públicos[1]
- Vamos ver no notebook!

Definição de Classe

- Uma classe serve como o meio principal para abstração na POO. Em Python, cada pedaço de dados é representado como uma instância de alguma classe. Uma classe fornece um conjunto de comportamentos na forma de funções membro (também conhecidas como **métodos**), com implementações comuns a todas as instâncias dessa classe. Uma classe também serve como um modelo para suas instâncias, determinando efetivamente a maneira como as informações de estado para cada instância são representadas na forma de **atributos** (também conhecidos como campos, variáveis de instância ou membros de dados)[1].

Definição de Classe

- **Exemplo: Classe CreditCard**
  - Como primeiro exemplo, analisaremos uma implementação de uma classe CreditCard baseada no projeto introduzido no livro[1].
  - Em Python, **o identificador self** desempenha um papel fundamental. No contexto da classe CreditCard, pode-se presumir que há muitas instâncias diferentes de CreditCard, e cada uma deve manter seu próprio saldo, seu próprio limite de crédito, e assim por diante. Portanto, cada instância armazena suas próprias variáveis de instância para refletir seu estado atual[1].
- Vamos ver no notebook!

Definição de Classe

- **Identificador** `self`
  - Identifica a instância em que um método é invocado
  - Cada instância armazena suas próprias variáveis de instância[1]
- **O Construtor**
  - Método __init__ inicializa a instância
  - Estabelece o estado do objeto com variáveis de instância[1]
- **Encapsulamento**
  - Uso de sublinhado para indicar membros não públicos
  - Usuários não devem acessar diretamente esses membros[1]
- **Métodos Adicionais como exemplo**
  - `charge`: adiciona preço ao saldo, verifica limite (No notebook!)
  - `make_payment`: reduz saldo com pagamento (No notebook!)

Considerações

Considerações

- **Recordando!!!**
  - Breve história da POO.
  - Introdução a POO.
  - Exemplos com Python e Java!
  - Exercícios.