Desenvolvimento de Sistemas II

Prof. Me. Cleber Schroeder Fonseca

Curso Técnico em Informática para Internet
IFRS Rio Grande

2024

Orientação a Objetos

A Programação Orientada a Objetos (POO) é um paradigma de programação que organiza o design de software em torno de "objetos"em vez de funções e lógica.

Esses objetos podem ser vistos como instâncias de "classes", que são tipos de dados definidos pelo usuário.

A OO é uma abordagem de desenvolvimento de software que tenta modelar os componentes do mundo real e suas interações.

1. Antecedentes Históricos
2. Desenvolvimento e Popularização
3. Consolidação e Impacto Moderno
4. Conceitos e Filosofia
5. POO Hoje

Década de 1960: Simula

- **Simula**: Considerado o primeiro exemplo de uma linguagem de programação orientada a objetos, Simula foi desenvolvido por Ole-Johan Dahl e Kristen Nygaard em 1967 no Centro de Computação Norueguês. A linguagem foi inicialmente criada para simulações e introduziu conceitos como classes, objetos, herança e encapsulamento.

Década de 1970: Smalltalk

- **Smalltalk**: Desenvolvido por Alan Kay e sua equipe na Xerox PARC na década de 1970, Smalltalk refinou e popularizou os conceitos de POO. Smalltalk foi pioneira em muitos aspectos da POO moderna, incluindo o conceito de "mensagens"(equivalente a chamadas de métodos) e a implementação de um ambiente de desenvolvimento totalmente orientado a objetos.

Década de 1980: Influência no Desenvolvimento de Outras Linguagens

- **C++**: Desenvolvido por Bjarne Stroustrup no início dos anos 1980, C++ foi uma extensão do C que incorporava conceitos de POO. C++ trouxe a POO para a arena de linguagens de sistemas e aplicáveis a uma ampla gama de software de engenharia e comercial.
- **Objective-C**: Também desenvolvido na década de 1980, Objective-C combinou a simplicidade de Smalltalk com a eficiência do C, e mais tarde foi adotado pela Apple para o desenvolvimento de software para macOS e iOS.

Década de 1990 e 2000: Linguagens Modernas

- **Java**: Introduzida pela Sun Microsystems em 1995, Java foi projetada desde o início como uma linguagem puramente orientada a objetos. A popularidade de Java ajudou a consolidar a POO como a abordagem dominante no desenvolvimento de software.
- **Python**: Embora não fosse inicialmente uma linguagem orientada a objetos pura, Python incorporou muitos conceitos de POO, tornando-se uma linguagem extremamente versátil e popular tanto para educação quanto para desenvolvimento de software em larga escala.

A filosofia da POO está enraizada em imitar o mundo real através da modelagem de entidades como objetos que possuem estados e comportamentos. Isso permite que o software seja:

- **Modular**: Componentes de software podem ser desenvolvidos e modificados independentemente.
- **Reutilizável**: A herança e a composição permitem que código existente seja reutilizado em novos contextos.
- **Mantenível**: O encapsulamento e a abstração simplificam a compreensão e a modificação do código.

Atualmente, a POO é amplamente usada em uma variedade de linguagens de programação (como Java, C++, Python, Ruby, C) e é um componente central da maioria dos paradigmas de desenvolvimento de software, incluindo desenvolvimento web, desenvolvimento móvel, jogos, e sistemas corporativos.

Conceitos de POO

Os conceitos fundamentais da POO usando Python, são:

1. Classes e Objetos
2. Atributos
3. Métodos
4. Encapsulamento
5. Herança
6. Polimorfismo

Classes e Objetos

Classe: Uma classe é como um molde para criar objetos. Ela define um conjunto de atributos e métodos que os objetos criados a partir da classe terão.

Objeto: Um objeto é uma instância de uma classe. Quando criamos um objeto, estamos criando uma entidade com as características definidas pela classe.

Classes e Objetos


```python
class Cachorro:
    # Método construtor
    def __init__(self, nome, idade):
        self.nome = nome  # Atributo da instância
        self.idade = idade  # Atributo da instância

    # Método para exibir informações do cachorro
    def exibir_informacoes(self):
        print(f"Nome: {self.nome}, Idade: {self.idade} anos")

# Criando um objeto da classe Cachorro
meu_cachorro = Cachorro("Rex", 5)
meu_cachorro.exibir_informacoes()  # Saída: Nome: Rex, Idade: 5 anos
```

1. Orientação a Objetos
2. Conceitos de POO
   - Classes e Objetos
   - Atributos
   - Métodos
   - Encapsulamento
   - Herança
   - Polimorfismo
3. Benefícios da POO

Atributos

Atributos são variáveis que pertencem a uma classe. Existem dois tipos principais de atributos: de instância e de classe.

São atributos que pertencem a cada instância individual de uma classe.

```python
class Pessoa:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

# Criando objetos da classe Pessoa
pessoa1 = Pessoa("Alice", 30)
pessoa2 = Pessoa("Bob", 25)

print(pessoa1.nome)  # Saída: Alice
print(pessoa2.idade)  # Saída: 25
```

Atributos

Atributos de Classe

São atributos que pertencem à própria classe e são compartilhados entre todas as instâncias da classe.

```python
class Carro:
    # Atributo de classe
    rodas = 4

    def __init__(self, marca, modelo):
        self.marca = marca
        self.modelo = modelo

# Criando objetos da classe Carro
carro1 = Carro("Toyota", "Corolla")
carro2 = Carro("Honda", "Civic")

print(carro1.rodas)  # Saída: 4
print(carro2.rodas)  # Saída: 4
```

Métodos

Métodos são funções definidas dentro de uma classe. Eles descrevem os comportamentos que os objetos da classe podem realizar.

Métodos


```python
class Retangulo:
    def __init__(self, largura, altura):
        self.largura = largura
        self.altura = altura

    def area(self):
        return self.largura * self.altura

    def perimetro(self):
        return 2 * (self.largura + self.altura)

# Criando um objeto da classe Retangulo
retangulo = Retangulo(5, 10)

print(retangulo.area())  # Saída: 50
print(retangulo.perimetro())  # Saída: 30
```

1. Orientação a Objetos
2. Conceitos de POO
   - Classes e Objetos
   - Atributos
   - Métodos
   - Encapsulamento
   - Herança
   - Polimorfismo
3. Benefícios da POO

Encapsulamento

Encapsulamento é o conceito de esconder os detalhes internos de uma classe e permitir o acesso apenas através de métodos específicos. Isso é feito usando atributos e métodos privados (prefixados com '__').

Encapsulamento

```python
class ContaBancaria:
    def __init__(self, titular, saldo):
        self.titular = titular
        self.__saldo = saldo  # Atributo privado

    def depositar(self, quantia):
        self.__saldo += quantia

    def sacar(self, quantia):
        if quantia <= self.__saldo:
            self.__saldo -= quantia
        else:
            print("Saldo insuficiente")

    def exibir_saldo(self):
        print(f"Saldo: R${self.__saldo}")

# Criando um objeto da classe ContaBancaria
conta = ContaBancaria("João", 1000)
conta.depositar(500)
conta.sacar(200)
```

1. Orientação a Objetos
2. Conceitos de POO
   - Classes e Objetos
   - Atributos
   - Métodos
   - Encapsulamento
   - Herança
   - Polimorfismo
3. Benefícios da POO

Herança

Herança é o conceito de criar uma nova classe a partir de uma classe existente. A nova classe (**subclasse**) herda os atributos e métodos da classe existente (**superclasse**).

Herança

```python
class Animal:
    def __init__(self, nome):
        self.nome = nome

    def fazer_som(self):
        pass

class Cachorro(Animal):
    def fazer_som(self):
        print("Au Au")

class Gato(Animal):
    def fazer_som(self):
        print("Miau")

# Criando objetos das subclasses
cachorro = Cachorro("Rex")
gato = Gato("Felix")

cachorro.fazer_som()  # Saída: Au Au
gato.fazer_som()  # Saída: Miau
```

1 Orientação a Objetos

2 Conceitos de POO
Classes e Objetos
Atributos
Métodos
Encapsulamento
Herança
Polimorfismo

3 Benefícios da POO

Polimorfismo

Polimorfismo é a capacidade de diferentes classes de serem tratadas como se fossem da mesma classe através de uma interface comum. Em Python, isso é geralmente implementado usando métodos que têm o mesmo nome em classes diferentes.

Polimorfismo


```python
class Passaro:
    def fazer_som(self):
        print("Piu Piu")

class Vaca:
    def fazer_som(self):
        print("Muu")

def fazer_animal_fazer_som(animal):
    animal.fazer_som()

# Criando objetos
passaro = Passaro()
vaca = Vaca()

fazer_animal_fazer_som(passaro)  # Saída: Piu Piu
fazer_animal_fazer_som(vaca)  # Saída: Muu
```

1. Orientação a Objetos
2. Conceitos de POO
   - Classes e Objetos
   - Atributos
   - Métodos
   - Encapsulamento
   - Herança
   - Polimorfismo
3. Benefícios da POO

Benefícios da POO

- **Modularidade**: Componentes do código são separados em classes, facilitando a manutenção e o entendimento.
- **Reutilização de Código**: A herança e os métodos reutilizáveis permitem o uso de código existente em novas aplicações.
- **Facilidade de Manutenção**: O encapsulamento ajuda a proteger os dados e a modularidade do código facilita a manutenção.
- **Flexibilidade e Extensibilidade**: Novas funcionalidades podem ser adicionadas com menos esforço e risco de causar problemas em outras partes do código.

MUITO OBRIGADO!

Cleber Schroeder Fonseca

http://ifrs.edu.br/riogrande

profcleberfonseca@gmail.com
cleber.fonseca@riogrande.ifrs.edu.br