POO (Programação Orientada a Objetos)

Programador Back-End

Prof.: João Pereira

Classes

No conceito de Orientação a Objetos, uma classe é uma estrutura para a definição de objetos. Através da definição da classe, são descritas as propriedades (ou atributos) que o objeto terá.

Além da definição dos atributos, a especificação de classes também descreve qual o comportamento dos objetos da classe, ou seja, quais funcionalidades podem ser aplicadas aos objetos da classe. Essas funcionalidades são especificadas através de métodos. Um **método** nada mais é que o equivalente a um procedimento (ou função), porém restrito às suas variáveis locais e aos atributos definidos para a classe.

Na implementação de uma classe, deve-se implementar o método construtor, que serve para inicializar os atributos/propriedades do objeto, para que estes estejam em um estado válido e usável.

Classes

No Python, um método construtor é definido da seguinte forma:

```
def __init__(self, ...):
      #inserção de atributos...
```

Onde:

def __init__(self): é o construtor da classe. O método "__init__" em Python é chamado automaticamente quando um objeto de uma classe é criado. Seu objetivo principal é inicializar os atributos do objeto. Recebe **self** como parâmetro, que se refere ao próprio objeto sendo criado.

__init__: o duplo sublinhado antes da palavra **init** indica que é um método privado. Embora tecnicamente seja possível acessar os elementos de fora da classe no python. O duplo sublinhado depois da palavra **init** é usado para evitar conflitos com nomes de classes derivadas.

Getters e Setters
(Métodos de Acesso e de Modificação)

No Python, pode-se criar getters e setters para o acesso e modificação de atributos de uma classe. Embora o Python permita o acesso direto aos atributos, é uma boa prática usar métodos para acessar e definir valores de atributos, pois fornece um controle maior acerca da forma com a qual os atributos são manipulados.

Getter: método de acesso. Acessa os valores de um determinado atributo.

Setter: método modificador. Define valores para os atributos.

Exemplo Prático

Para um sistema de uma Calculadora de 2 números, é implementada a classe Calculadora, tendo como atributos **num1** e **num2** e métodos para a manipulação destes dados.

```
class Calculadora: #define uma classe chamada Calculadora
    def __init__(self, __num1, __num2): #método __init__  é o construtor da classe Calculadora.
        self.__num1 = 0 #atributo privado num1
        self.__num2 = 0 #atributo privado num2

    def set_num1(self, num1):
        self.__num1 = num1

    def set_num2(self, num2):
        self.__num2 = num2

    def soma(self):
        return self.__num1 + self.__num2
```

Exemplo Prático

```
num1 = float(input("Digite o 1º número: "))
num2 = float(input("Digite o 2º número: "))
op = int(input("Digite uma opção: 1-Soma 2-Subtração 3-Multiplicação 4-Divisão. \nOp. Desejada: "))

calculadora = Calculadora() #criando uma instância da classe Calculadora

calculadora.set_num1(num1)
calculadora.set_num2(num2)

match op:
    case 1:
        print(f'A soma de {num1} com {num2} = {calculadora.soma()}')
```

Exercício Prático

Implemente os métodos **subtracao(), multiplicação()** e **divisao()** para o código anterior.

Diagrama de Classes

Visibilidade

A visibilidade de atributos e métodos é indicada por símbolos que aparecem antes do nome do atributo ou método. Os símbolos comuns incluem “+” para público, “-” para privado “#” para protegido e “~” para pacote.

1. **Público (+)**: Indica que o atributo ou método é acessível fora da classe. Qualquer outra classe pode acessar um membro público.

2. **Privado (-):** Indica que o atributo ou método é acessível apenas dentro da própria classe. Não pode ser acessado diretamente por outras classes. Usado quando é necessário restringir o acesso, ocultando de classes externas.

3. **Protegido(#):** Indica que o atributo ou método é acessível dentro da própria classe e em subclasses (herança). Não pode ser acessado por classes que não são subclasses. Usado quando é necessário permitir acesso para classes filhas, mas não para outras classes externas

Diagrama de Classes

Visibilidade

4. Pacote(~): Indica que o atributo ou método é acessível apenas dentro do pacote (ou módulo) ao qual a classe pertence. Não pode ser acessado por classes fora do pacote. É uma forma intermediária de visibilidade entre privado e protegido, permitindo acesso a classes relacionadas no mesmo pacote. A escolha da visibilidade adequada depende do design da classe e de como será o controle do acesso aos membros da classe.

Use visibilidade pública (+) para membros que precisam ser acessíveis a outras partes do sistema ou a classes externas.

Use visibilidade privada (-) para ocultar detalhes de implementação e fornecer encapsulamento. A maioria dos atributos deve ser privada.

Use visibilidade protegida (#) quando desejar que os membros sejam acessíveis às subclasses, permitindo extensão da classe.

Use visibilidade de pacote (~) quando membros devem ser acessíveis apenas dentro do pacote. Isso é útil para controlar o acesso em um escopo mais restrito do que público, mas menos restrito do que protegido.