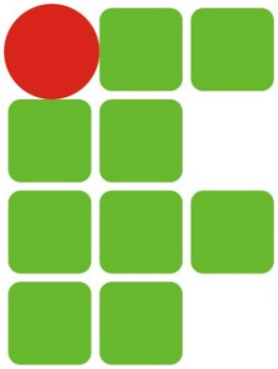


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Programação Orientada a Objetos

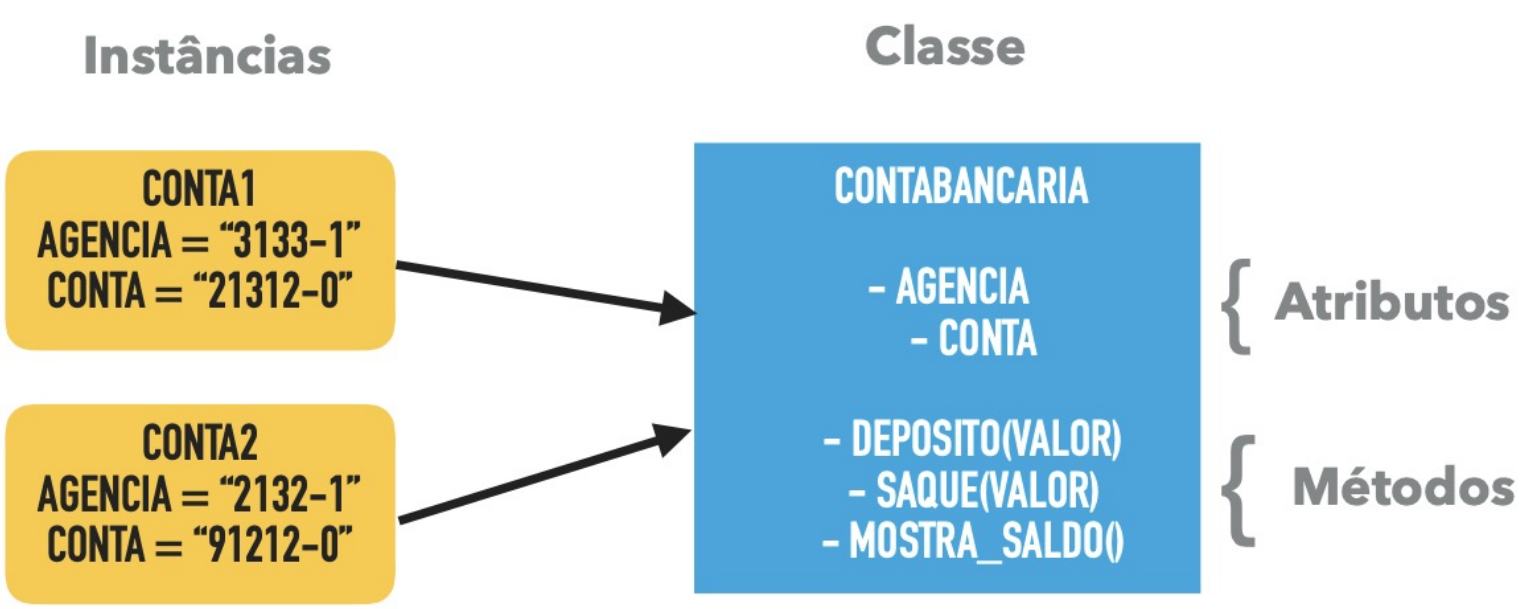
APRESENTAÇÃO DA DISCIPLINA

PROFESSOR:
Demetrios Coutinho
Demetrios.coutinho@ifrn.edu.br

AGENDA

- Conceito de Encapsulamento

CLASSES VS INSTÂNCIAS



Quem pode acessar os dados

- A maioria das linguagens orientadas a objetos possuem o conceito de **controle de acesso**;
  - Nessas linguagens atributos e métodos podem **ser privados, protegidos ou públicos;**
- No **python não existe isso!**
  - Python não acredita em leis que te forçam a algo que pode ser prejudicial no futuro;
  - O python fornece diretrizes (não obrigatórias) e boas práticas;
  - Assim sendo, tecnicamente todos os atributos e métodos são públicos.

Quem pode acessar os dados

- Por convenção podemos ainda colocar um **underscore** na frente de **um atributo ou método;**
  - Programadores python vão interpretar isso como um sinal de que aquele atributo/método é privado;
- Outra possibilidade é colocar **um underscore duplo** na frente do identificador do atributo ou método;
  - Ao colocar underscore duplo, o python realiza name mangling;
  - Veja um exemplo:

Exemplo

```
1  class SecretString:
2      '''Uma maneira nada segura de armazenar uma string
3      que contém um segredo.'''
4      def __init__(self, plain_string, pass_phrase):
5          self.__plain_string = plain_string
6          self.__pass_phrase = pass_phrase
7      def decrypt(self, pass_phrase):
8          '''Só mostra o segredo se o senha estiver certa.'''
9          if pass_phrase == self.__pass_phrase:
10             return self.__plain_string
11         else:
12             return ''
```

In [6]:

```
1  secret_string = SecretString("ACME: Top Secret", "antwerp")
```

In [9]:

```
1  print(secret_string.decrypt("antwerp"))
```

```
ACME: Top Secret
```

Exemplo

```
1  print(secret_string.__plain_text)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call
 last)
<ipython-input-10-376091f5ceea> in <module>()
----> 1 print(secret_string.__plain_text)

AttributeError: 'SecretString' object has no attribute '__plain_text'
```

- Entretanto, podemos facialmente acessar a senha e o segredo:

In [13]:

```
1   print(secret_string._SecretString__pass_phrase)
2  print(secret_string._SecretString__plain_string)
```

```
antwerp
ACME: Top Secret
```

Name Magling

- O **name magling** do python coloca o nome da classe como prefixo quando usamos o underscore duplo;
  - Em geral, programadores python não irão mexer em variáveis como underscore duplo ou mesmo simples;
  - Salvo se eles tiverem uma boa razão para fazer isso;

Resumo

- O python usa _ (underscore) para explicitar se um atributo (método) é publico, protegido ou privado:
- **nome - public**: pode ser acessada de qualquer lugar
- **_name - protected**: Como um membro público, mas não deve ser acessada (convenção)
- **__name - private**: Não pode ser acessada de fora da classe

Getter and Setter

Então como acessar ou editar um atributo privado?

```
def set_nome(self, nome):
    self.__nome = nome

def get_nome(self):
    return self.__nome
```

DÚVIDAS?