



PYTHON Object Oriented Programming 	 INSTITUTO FEDERAL Paraíba Campus João Pessoa Programação e Estrutura de Dados Professor: Alex Sandro da Cunha Rêgo Última atualização: 28/02/2023	Lista 6
<p align="center">CRIANDO CLASSES E OBJETOS</p> <p align="center">Construindo Estruturas de Classes a partir de Sentenças</p> <p align="center"><i>Material original: Edemberg Rocha/Thiago Moura (com adaptações)</i></p>		

1. Implemente uma classe chamada **Data**, que poderá ser usada para representar uma data. A classe deve atender aos seguintes requisitos:

- Definir os atributos de instância privados **dia**, **mês** e **ano**, do tipo inteiro;
- Um Construtor parametrizado que recebe como argumento, na sequência, o dia, mês e ano que representam a data. Se o ano não for indicado como argumento, utilize o ano de 2023 como ano padrão;
- Métodos acessores e modificadores para os atributos privados;
- Método `__str__(self)` que retorna a data no formato dd/mm/aaaa;
- Um método `setData()` com a seguinte assinatura:

```
def setData(self, dia:int, mes:int, ano:int)
```

Validações devem ser realizadas quando necessárias, seja no construtor ou na tentativa de modificação dos atributos

Escreva um programa em um arquivo separado para criar objetos da classe **Data** e testar seus métodos.

2. Implemente uma classe chamada **Aluno** conforme os seguintes requisitos:

- Atributos privados de instância **matricula**, do tipo int; **nome**, do tipo String; notas do tipo *list*. A matricula deve ter obrigatoriamente 8 dígitos, conforme o seguinte padrão (valores são apenas exemplos):

Ano **Semestre** **Sequencial**

20231050

- Construtor parametrizado que recebe como argumento a matrícula do aluno e seu nome;

- c) Métodos acessores `getNome(self)` e `getMatricula(self)` ou usando a sintaxe nativa de Python com `@property` e `@<propriedade>.setter`;
- d) Método modificador para a propriedade privada `nome`;
- e) criar o método `getMatriculaFormatada(self)` para devolver a matrícula no seguinte formato:

“2023.1.050”
- f) criar o método `media(self)` que retorna a média aritmética das notas;
- g) Método `adicionaNota(self, outraNota:int)`, para adicionar uma nota à lista de notas do aluno.

Escreva um programa em um arquivo separado para criar objetos da classe **Aluno** e testar seus métodos.

3. Escreva uma classe para representar um **país** de qualquer continente. Um país tem como atributos privados o seu **nome**, o **nome da capital**, sua **dimensão** em Km² e uma **lista de países** com os quais ele faz fronteira. A classe deve atender aos seguintes requisitos:
 - a) Fornecer um construtor que receba o nome, capital e a dimensão do país em Km²;
 - b) Disponibilize métodos `get` e `set` para os atributos que considerar convenientes;
 - c) Implementar um método que adiciona o nome de um país que faz fronteira com o país que representa o objeto. O método não deve permitir duplicidade de países e não deve diferenciar maiúscula de minúscula.
 - d) Adicionar o método `__str__` para retornar as informações do país da seguinte forma:

“Brasil, capital Distrito Federal, 8516000 km²”
 - e) Implemente um método que ao receber um objeto que representa outro país, informe se ambos possuem fronteira (ou não). A assinatura do método é definida por:

```
def fazFronteiraCom(self, outroPais: Pais)->bool:
```

Exemplo de uso:

```
brasil = Pais("Brasil", "Distrito Federal", 8516000)
...
argentina = Pais("Argentina", "Buenos Aires", 2780000)

if brasil.fazFronteiraCom(argentina):
    print("Sim, Brasil e Argentina são países vizinhos")
```

Escreva um programa em um arquivo separado para criar objetos da classe **País** e testar seus métodos.

4. Implemente uma classe **ContaCorrente**, com os atributos de instância **numero, nome do titular e saldo**. Ainda na classe, implemente os métodos **depositar()** e **sacar()**, parametrizando neles, o valor a ser depositado ou sacado, respectivamente. O método **depositar()** não possui retorno, devendo apenas incrementar o valor do saldo. O método **sacar()** deverá **retornar** um valor **booleano** (**true** – se sacou com sucesso, pois há saldo suficiente, decrementando-o; ou **false** caso contrário). A classe deve declarar um construtor para receber os valores iniciais aos atributos correspondentes. Se não for informado o saldo, o valor padrão é zero. Defina os métodos **get** e **set** que achar convenientes para o domínio do problema. Por fim, faça o que se pede:

- a) Escreva um programa para criar dez instâncias de **ContaCorrente**, armazenando-os em uma *list*. Os valores para inicialização dos objetos poderão ser lidos a partir do teclado ou povoados automaticamente.
- b) Após povoamento do *list* de contas corrente, disponibilize um menu de operações para o usuário. **Em um loop**, o programa ficara solicitando ao usuário, qual a operação ele deseja realizar:

```
(d) Depositar
(s) Sacar
(d) Saldo
(r) Sair
=====
Opção: _
```

A opção **depositar** deve solicitar a leitura pelo teclado do número da conta e o valor a ser depositado, antes de realizar a operação correspondente. realizar a operação. A opção **sacar** deve ler o numero da conta e o valor a ser sacado, indicando na tela se o saque ocorreu com sucesso ou não. A opção **Saldo** deve solicitar a leitura do numero da conta e exibir o seu respectivo saldo. Por fim, a opção **sair** encerra a execução do programa.