



PYTHON Object Oriented Programming 	 INSTITUTO FEDERAL Paraíba Campus João Pessoa Programação e Estrutura de Dados Professor: Alex Sandro da Cunha Rêgo Última atualização: 04/10/2018	Prática 4
POLIMORFISMO: HERANÇA X CLASSE ABSTRATA		

ORIENTAÇÕES

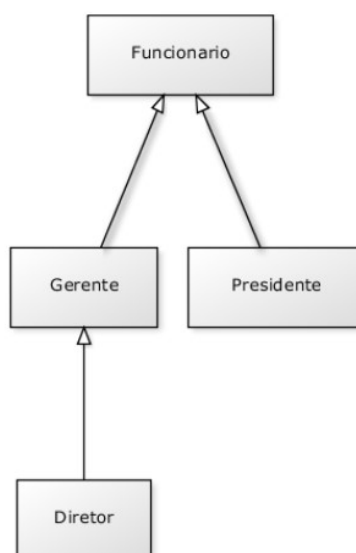
Pré-requisitos:

- Lógica de programação, conhecimento na criação de classes e objetos em Python: construtor, métodos e propriedades públicas e privadas, de classe e de instância.
- Entendimento do conceito de herança, classe abstrata e polimorfismo

Instruções

- Leia o enunciado com atenção e faça o que se pede

Considere a hierarquia de classes abaixo:



Codifique uma **classe abstrata** denominada **Funcionário** que atenda aos seguintes requisitos:

- **Propriedades:** nome, salário, grau de instrução (ensino médio, ensino superior, especialista, mestre ou doutor). Utilize o recurso **Enumerations** em Python para representar as opções de grau de instrução;
- Construtor com os parâmetros "nome" e "salário base";
- Encapsulamento das propriedades;
- Métodos get() e set() para as propriedades. Identifique quais podem ter métodos get ou set implementados, diante da lógica do problema;
- Método **__str__** que imprima a seguinte mensagem: "Objeto do tipo <nome da classe>: <nome do funcionário>, salário <salário>"
- Método **contracheque()**, que exibe quanto o funcionário ganha considerando o salário e a bonificação;
- Método abstrato **addBonificação()** que se comporta da seguinte forma:

- (a) Um funcionário da classe **Diretor** Recebe 30% de bonificação em cima do salário. Além disso, acrescenta-se o adicional de qualificação: 15% especialista, 25% mestre e 50% doutor;
- (b) Um funcionário da classe **Presidente** Recebe o triplo do salário. Apenas se for doutor, terá um adicional de qualificação de 5 salários;
- (c) Um funcionário da classe **Gerente** só recebe, como bonificação, o adicional de qualificação (ver regra em Diretor)

Outros métodos podem ser criados para atender ao que pede este exercício, caso ache conveniente.

A criação dos métodos deve levar em conta as situações que podem “ferir” as regras do negócio. Por exemplo, qualquer que seja o tipo do funcionário, seu salário não pode ser negativo. Qualquer violação das regras do negócio deve lançar uma exceção **FuncionarioException** e, obviamente, tratada no local adequado.

Crie um programa **main.py** que instancie e armazene em um list objetos da classe Gerente, Presidente e Diretor. Exiba na tela a relação dos funcionários da seguinte forma (simulação considerando salário de R\$ 2.000,00 para todos):

Objeto	Funcionário	Grau de Instrução	Salário Base
=====	=====	=====	=====
Diretor	João Silva	Mestrado	R\$ 3.600,00
Gerente	Ana Karina	Especialização	R\$ 2.300,00
...			

Aplicando a técnica de **polimorfismo**, percorra todos os objetos da coleção e exiba quanto ganha cada funcionário. O polimorfismo deve ser realizado invocando o método abstrato correspondente. Realize chamadas ao método `__str__()` sempre que for conveniente.

Referências:

Enum in Python. Disponível em: <<https://www.geeksforgeeks.org/enum-in-python/>>. Acesso em: 22 mar. 2022