

PRÁTICA LABORATORIAL 02 (PL02)

CURSO	UNIDADE CURRICULAR
LICENCIATURA EM ENGENHARIA INFORMÁTICA	PROGRAMAÇÃO 4
DOCENTE	
HELDER RODRIGO PINTO	

PROGRAMAÇÃO ORIENTADA A OBJETOS EM PYTHON

1. Implemente as classes: Quadrado e Rectangulo, tal são definidas abaixo:
 - a. A classe Quadrado tem um atributo double: lado; e a classe Rectangulo tem dois atributos double: base e altura.
 - b. Sendo usada a notação “padrão”, os atributos devem ser definidos com propriedades.
 - c. Adicione os construtores (vazio, por parâmetros e cópia).
 - d. O construtor vazio da classe Quadrado deve criar um Quadrado com 1cm de lado.
 - e. O construtor vazio da classe Rectangulo deve criar um Rectangulo com 1cm de base e 1cm de altura.
 - f. Implemente os métodos de comportamentos das classes: Perimetro, Area e Diagonal.
 - g. Teste a aplicação criando vários objetos da classe Quadrado.
 - h. Teste a aplicação criando vários objetos da classe Retangulo.

Default	Restrições	Quadrado	Default	Restrições	Rectangulo
1	>0	- _lado : double + Perimetro() : double + Area() : double + Diagonal() : double	1	>0	- _base : double - _altura : double + Perimetro() : double + Area() : double
		$P = L + L + L + L$ $A = L^2$ $d = L \sqrt{2}$			$P = 2(base + altura)$ $A = base \times altura$

Figura 1 –Classes Quadrado e Retângulo

2. Crie a classe Ponto.
 - a. A classe Ponto tem dois atributos double: x e y.
 - i. Sendo usada a notação “Pascal”, os atributos devem ser definidos como auto-propriedades.
 - b. Adicione os construtores (vazio, por parâmetros e cópia).
 - i. O construtor vazio deve criar o ponto Origem (x=0, y=0).

- c. Implemente o método de comportamento da classe: DistE2P() que calcula a distância entre dois pontos.

Este método é recebe um Ponto p e devolve um double que representa a distância entre o objeto corrente (que chama o método) e o objeto enviado por parâmetros.

- d. Teste a aplicação criando vários objetos da classe Ponto.

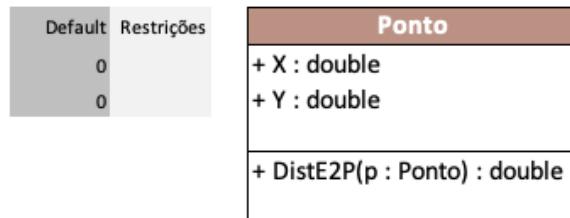


Figura 3 – Classe Ponto

3. Crie a classe Recta.

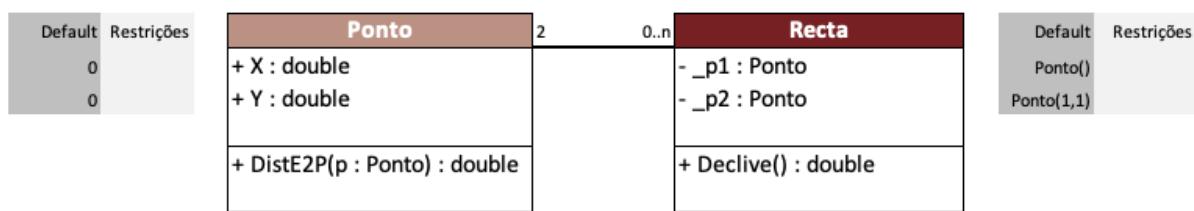
- a. A classe Recta tem dois atributos do tipo Ponto: _p1 e _p2.

i. Sendo usada a notação “Padrão”, os atributos devem ser implementados como propriedades.

- b. Adicione os construtores (vazio, por parâmetros e cópia) e propriedades.

ii. O construtor vazio deve criar dois pontos (_p1 e _p2) como sendo ponto Origem (x=0, y=0), da forma que entender adequado.

- c. Teste a aplicação criando vários objetos da classe Recta.



$$\text{dist} = \sqrt{(\text{x2}-\text{x1})^2 + (\text{y2}-\text{y1})^2}$$

$$m = (\text{P1.Y}-\text{P2.Y})/(\text{P1.X}-\text{P2.X})$$

Figura 4 – Diagrama de Classes Ponto e Reta



4. Crie as classes EndIPv4 e DHCP.

- Defina os atributos, propriedades e métodos de comportamento, considerando os respetivos valores default e restrições.
- Documente as classes e métodos e crie documentação automática com o Sphinx.
- Teste a aplicação criando vários objetos de cada uma das classes.
- Implemente testes unitários adequados para validar as restrições e regras de negócio.

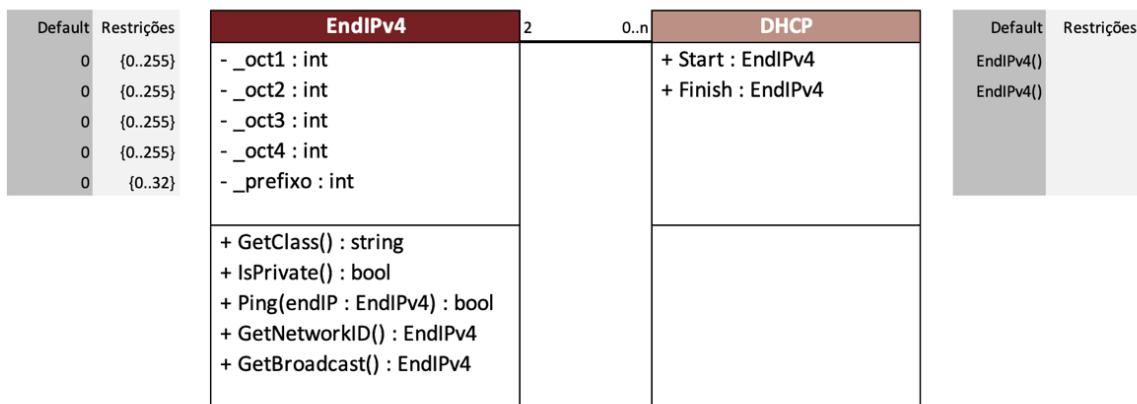


Figura 5 – Diagrama de Classes EndIPv4 e DHCP

5. Desenhe uma aplicação que simule a criação de objetos Produto.

- Defina as classes Produto, ProdutoAlimentacao e ProdutoHigiene, tal como no diagrama:

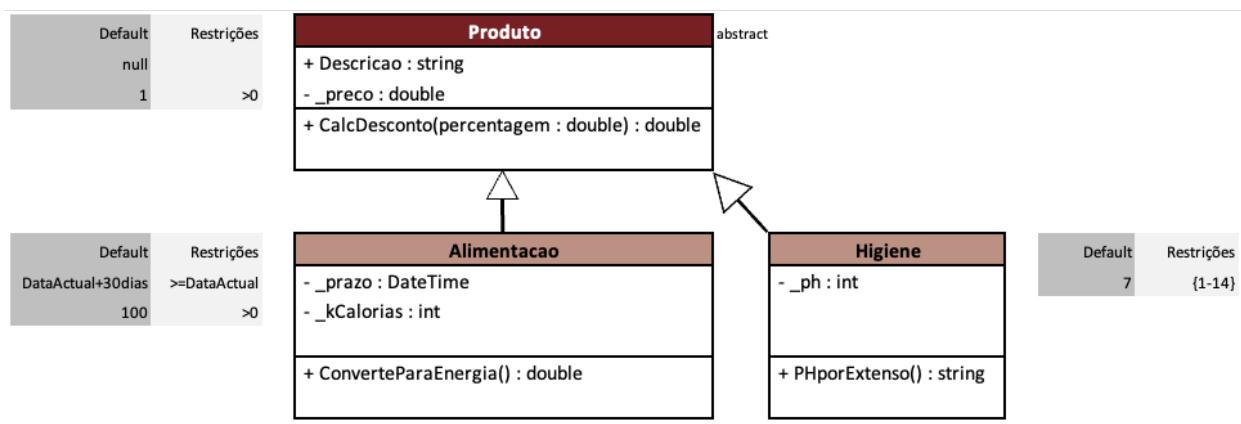


Figura 6 – Diagrama de Classes Produto

- Teste a aplicação criando listas de Produtos diferentes (Alimentação e Higiene)
- Documente as classes e métodos e crie documentação automática com o Sphinx.
- Implemente testes unitários adequados para validar as restrições e regras de negócio.

6. Implemente uma solução para um Banco, que gere várias contas poupança de clientes.

a. Implemente a classe ContaPoupanca, com as respetivas restrições:

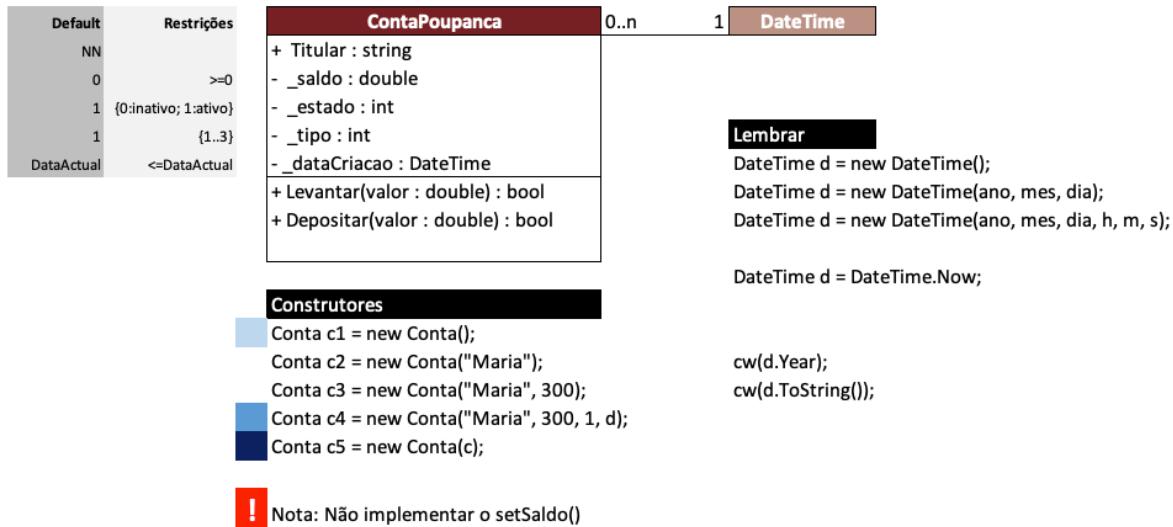


Figura 7 – Diagrama de Classes para o Design Pattern Strategy

b. Crie um método para fazer o depósito de juros na conta poupança. Para tal, considere que para cada tipo de conta, o cálculo segue os seguintes critérios:

- conta poupança clássica (1): valor de juro a depositar = saldo atual * 0.01;
- conta poupança infantojuvenil (2): valor de juro a depositar = saldo atual * 0.03;
- conta poupança renda fixa (3): valor de juro a depositar = saldo atual * 0.04;

c. Teste a aplicação de forma que criar uma conta poupança e verifique se o valor do juro depositado é calculado consoante o tipo de conta poupança no momento.

d. Implemente testes unitários adequados para validar as restrições e regras de negócio.

7. Implemente uma solução para uma agenda de contactos, baseando-se no padrão MVC.

a. Implemente a classe Contacto (model), com as respetivas restrições:

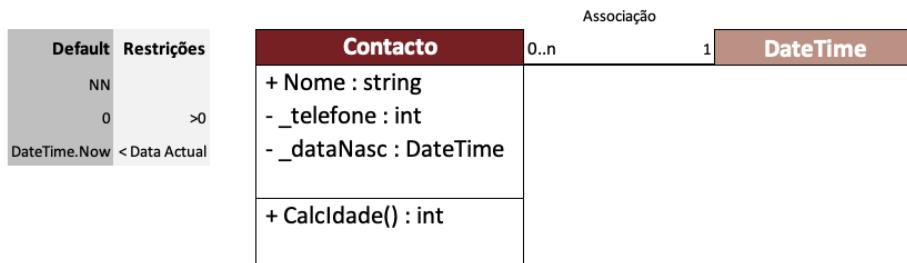


Figura 8 – Diagrama de Classes para o Design Pattern MVC

- b. Implemente as classes ContactoView e ContactoController, considerando uma aplicação em Console App.
- c. Teste a aplicação de forma que criar uma lista de contactos fazendo a mediação da gestão e visualização de dados pelo Controller.
- d. Implemente testes unitários adequados para validar as restrições e regras de negócio.