Exercícios de Classes e Métodos Abstratos

Juliana Pirolla

Exercício 1

Escreva um programa em Python que utilize classes abstratas para representar diferentes tipos de polígonos. O programa deve seguir as seguintes especificações:

- Defina uma classe abstrata chamada Polygon.
- A classe Polygon deve ter um método abstrato chamado noofsides(), que não possui implementação.
- Crie quatro classes concretas: Triangle, Pentagon,Hexagon e Quadrilateral, que herdam da classe Polygon.
- Cada classe concreta deve sobrescrever o método noofsides() para exibir uma mensagem informando o número de lados do polígono correspondente.
- No código principal, crie instâncias das classes Triangle, Pentagon, Hexagon e Quadrilateral e chame o método noofsides() para cada uma delas.

Exercício 2

Crie uma classe abstrata chamada "Colaborador" com um método abstrato chamado "calcular_salario_liquido". A classe representa um colaborador em uma empresa simplificada, com diferentes tipos: escriturários, vendedores e gerentes. Cada tipo de colaborador possui um salário bruto base e comissões específicas. Os escriturários não possuem comissões, os vendedores recebem 10% de comissão sobre as vendas realizadas, e os gerentes recebem 5% de comissão sobre o lucro líquido do departamento. Além disso, do salário bruto são aplicados descontos de 10% para segurança social e 15% para imposto de renda na fonte. Os dados dos colaboradores são fornecidos em um arquivo com campos separados por vírgulas, incluindo informações como nome, cargo, salário base, vendas realizadas e lucro líquido. A implementação da classe abstrata e do método abstrato permitirá que outras classes concretas herdem da classe "Colaborador" e forneçam sua própria implementação para o cálculo do salário líquido.

Exercício 3

2. O código abaixo foi escrito por um programador que não conhece o conceito de classes base abstratas. Reescreva esse código com a ajuda do módulo `abc`.

```
import math
class Solid:
    def __init__(self, mass):
        self._mass = mass
    def mass(self):
        return self._mass
    def volume(self):
        pass
    def density(self):
        return self._mass / self.volume()

class RectangularParallelepiped(Solid):
    def __init__(self, mass, lx, ly, lz):
        Solid.__init__(self, mass)
        self._x = lx
        self._y = ly
        self._z = lz
    def volume(self):
        return self._x * self._y * self._z

class Sphere(Solid):
    def __init__(self, mass, r):
        Solid.__init__(self, mass)
        self._r = r
    def volume(self):
        return 4 / 3 * math.pi * self._r ** 3
```