Lista de Exercícios - Pilares da POO em Python

1. Encapsulamento (Fácil):

Crie uma classe `ContaBancaria` com atributos privados `saldo` e `titular`. Implemente métodos para depositar, sacar e consultar o saldo. Não permita saques com valor superior ao saldo.

2. Encapsulamento (Médio):

Adicione validações no método de saque da classe `ContaBancaria`. Caso o valor seja negativo ou superior ao saldo, lance uma exceção personalizada chamada `SaldoInsuficienteError`.

3. Herança (Fácil):

Crie uma classe base `Veiculo` com atributos `marca` e `modelo`. Crie duas classes filhas `Carro` e `Moto` que herdam de `Veiculo`. Adicione um método `informacoes()` em `Veiculo` que é reutilizado pelas classes filhas.

4. Herança (Médio):

Expanda o exercício anterior adicionando o atributo `tipo_combustivel` na classe `Carro` e `cilindradas` na classe `Moto`. Sobrescreva o método `informacoes()` para exibir detalhes específicos de cada classe filha.

5. Polimorfismo (Fácil):

Implemente uma classe base `FormaGeometrica` com um método `area()` que retorna 0. Crie classes `Quadrado` e `Circulo` que sobrescrevem o método `area()` para calcular as áreas corretamente com base nos atributos fornecidos.

6. Polimorfismo (Difícil):

Expanda o exercício anterior para incluir mais formas geométricas, como `Triangulo` e `Retangulo`. Adicione um método `descrever()` em cada classe que explica como a área é calculada para aquela forma específica.

7. Abstração (Fácil):

Crie uma classe abstrata `InstrumentoMusical` com o método `tocar()`. Implemente duas classes concretas, `Guitarra` e `Piano`, que herdam de `InstrumentoMusical` e fornecem implementações específicas para o método `tocar()`.

8. Abstração (Difícil):

Crie uma hierarquia de classes para representar um sistema de pagamento. A classe base `Pagamento` deve ter métodos abstratos como `processar_pagamento()` e `emitir_recibo()`. Implemente subclasses `CartaoCredito` e `Pix` com comportamentos específicos para cada tipo de pagamento.