**Enunciado do Exercício Prático: Aplicativo de Edição de Fotos com Padrão Strategy**

Você está encarregado de desenvolver um componente de um aplicativo de edição de fotos que permite aos usuários aplicar diferentes filtros às suas imagens. Para facilitar a expansão futura do aplicativo (como a adição de novos filtros) e para permitir que os usuários mudem os filtros dinamicamente, você decidiu usar o padrão de design **Strategy**.

**Objetivos**

1. Implementar o padrão **Strategy** para permitir a troca flexível de algoritmos de filtros em uma aplicação de edição de fotos.

2. Criar uma classe de contexto (**ImageEditor**) que utilizará diferentes estratégias de filtros.

3. Desenvolver classes concretas de estratégia para dois tipos de filtros: Preto e Branco e Sépia.

**Requisitos Funcionais**

**Filtro**

- Crie uma interface de estratégia chamada **FilterStrategy** com um método **apply** que aceita um argumento **image**.

- Desenvolva duas classes concretas que implementam **FilterStrategy**:

- **BlackAndWhiteFilter**: Aplica um filtro preto e branco à imagem.

- **SepiaFilter**: Aplica um filtro sépia à imagem.

Cada método **apply** deve imprimir uma mensagem indicando que o filtro foi aplicado, incluindo o nome do arquivo da imagem.

**Editor de Imagens**

- Crie uma classe **ImageEditor** que contém uma referência a um **FilterStrategy**.

- Inclua um método **set\_filter** que permite mudar a estratégia de filtro do editor.

- Adicione um método **apply\_filter** que usa a estratégia atual para aplicar um filtro à imagem fornecida.

**Instruções Detalhadas**

**1. Definição da Interface de Estratégia:** defina uma interface FilterStrategy com um método abstrato apply(self, image).

**2. Implementação das Estratégias Concretas:** implemente **BlackAndWhiteFilter** e **SepiaFilter**, duas classes que herdam de **FilterStrategy**.

Em cada classe, implemente o método apply para imprimir uma mensagem formatada que indique que o respectivo filtro foi aplicado ao nome do arquivo passado como argumento.

**3. Desenvolvimento da Classe Contexto:**

- Implemente a classe **ImageEditor** que pode usar objetos **FilterStrategy**.

- Inclua um método **set\_filter(self, filter\_strategy)** para alterar a estratégia de filtro do editor.

- Adicione um método **apply\_filter(self, image)** que chama o método **apply** do objeto de estratégia de filtro atual.

**4. Teste Seu Código:**

- Crie uma instância de **ImageEditor** e inicialize-a com **BlackAndWhiteFilter**.

- Chame **apply\_filter** com um exemplo de nome de arquivo de imagem para verificar a saída.

- Altere o filtro para SepiaFilter usando set\_filter e aplique-o ao mesmo nome de arquivo para observar a mudança de comportamento.