1. Explique o que é o padrão de projeto Proxy e em quais situações ele é mais adequado de ser utilizado.

O padrão Proxy é uma poderosa ferramenta de design de software que permite adicionar funcionalidades aos objetos de maneira flexível e controlada. Ele é especialmente útil para controle de acesso, otimização de recursos, interfaces remotas e operações adicionais, melhorando a modularidade e a manutenção do código.

2) Quais são os principais tipos de Proxy e como cada um deles funciona?

Os diferentes tipos de Proxy (Virtual, Remoto, de Proteção, Cache e Inteligente) permitem adicionar funcionalidades específicas e controle ao acesso de objetos, aumentando a flexibilidade, a segurança e o desempenho das aplicações. Cada tipo de Proxy é adequado para situações distintas, dependendo das necessidades específicas do sistema em desenvolvimento.

3) Quais são as vantagens de utilizar o padrão Proxy em um sistema? Dê exemplos práticos para ilustrar suas respostas.

O uso do padrão Proxy traz várias vantagens, como controle de acesso, otimização de desempenho, simplificação de comunicação remota, facilitação de logging e monitoramento, e simplificação do código cliente. Cada tipo de Proxy (Virtual, Remoto, de Proteção, Cache, e Inteligente) oferece benefícios específicos que ajudam a construir sistemas mais eficientes, seguros e manuteníveis.

Exemplo do melhor desempenho: class HighResolutionImage:

def \_\_init\_\_(self, filename):

self.filename = filename

self.load\_image()

def load\_image(self):

print(f"Loading high-resolution image {self.filename}")

def display(self):

print(f"Displaying {self.filename}")

class ImageProxy:

def \_\_init\_\_(self, filename):

self.filename = filename

self.real\_image = None

def display(self):

if not self.real\_image:

self.real\_image = HighResolutionImage(self.filename)

self.real\_image.display()

# Uso do Proxy Virtual

image = ImageProxy("large\_image.png")

image.display()

4) Descreva as desvantagens do padrão Proxy e como elas podem impactar o desenvolvimento e a manutenção de um sistema.

Apesar de suas vantagens, o padrão Proxy pode introduzir complexidade adicional, impactar o desempenho, dificultar o debugging e os testes, aumentar o uso de memória e tornar a manutenção e a evolução do sistema mais desafiadoras. Esses fatores devem ser cuidadosamente considerados ao decidir implementar proxies em um sistema, garantindo que os benefícios superem as desvantagens potenciais. A análise cuidadosa do contexto e das necessidades específicas do projeto é essencial para fazer um uso eficaz do padrão Proxy.

5) Utilizando a linguagem Java, elabore um exemplo onde o padrão Proxy seria útil para melhorar a performance de um sistema. Explique detalhadamente o código desenvolvido.

Primeiro, definimos uma interface comum que tanto o objeto real quanto o proxy irão implementar. Isso garante que o cliente pode usar o proxy da mesma forma que usaria o objeto real.

public interface Image {

void display();

}

Em seguida, implementamos a classe real que carrega a imagem de alta resolução.

public class RealImage implements Image {

private String filename;

public RealImage(String filename) {

this.filename = filename;

loadImageFromDisk();

}

private void loadImageFromDisk() {

System.out.println("Loading " + filename);

}

@Override

public void display() {

System.out.println("Displaying " + filename);

}

}

A classe RealImage carrega a imagem do disco quando é instanciada. Isso é caro em termos de tempo de carregamento.

Agora, implementamos a classe Proxy que irá atrasar a criação da RealImage até que seja realmente necessário.

public class ProxyImage implements Image {

private RealImage realImage;

private String filename;

public ProxyImage(String filename) {

this.filename = filename;

}

@Override

public void display() {

if (realImage == null) {

realImage = new RealImage(filename);

}

realImage.display();

}

}

A classe ProxyImage possui a mesma interface (Image). No método display(), ela cria uma instância de RealImage apenas quando este método é chamado pela primeira vez, implementando a técnica de carga preguiçosa.

Finalmente, utilizamos o proxy em um cliente.

public class ProxyPatternDemo {

public static void main(String[] args) {

Image image1 = new ProxyImage("high\_resolution\_image1.jpg");

Image image2 = new ProxyImage("high\_resolution\_image2.jpg");

// As imagens serão carregadas apenas quando display() é chamado

System.out.println("Proxy created. No images loaded yet.");

image1.display(); // Carrega e exibe a primeira imagem

image1.display(); // Apenas exibe a primeira imagem

image2.display(); // Carrega e exibe a segunda imagem

image2.display(); // Apenas exibe a segunda imagem

}

}

1. **Interface Image:** Define um método display() que será implementado tanto pela RealImage quanto pelo ProxyImage.
2. **Classe RealImage:** Representa a imagem real. No construtor, ela carrega a imagem do disco (simulado pelo método loadImageFromDisk()). O método display() exibe a imagem.
3. **Classe ProxyImage:** Contém uma referência a um objeto RealImage. No método display(), ela verifica se o RealImage já foi criado. Se não, cria o RealImage e, em seguida, chama o método display() do RealImage.
4. **Classe ProxyPatternDemo:** Demonstra o uso do padrão Proxy. Cria instâncias de ProxyImage para duas imagens. As imagens reais são carregadas apenas quando o método display() é chamado pela primeira vez.

6) Defina o padrão de projeto Facade e descreva seu principal objetivo em um sistema de software.

O padrão Facade é uma poderosa ferramenta de design de software que simplifica a interação com subsistemas complexos, proporcionando uma interface de alto nível que é fácil de usar e entender. Ele melhora a manutenibilidade e a flexibilidade do código, reduzindo o acoplamento entre o cliente e o subsistema, tornando os sistemas mais modulares e compreensíveis.

7) Quais são as vantagens de utilizar o padrão Facade? Explique como ele pode facilitar a manutenção e a evolução de um sistema.

O padrão Facade oferece uma interface simplificada e unificada para sistemas complexos, reduzindo a complexidade do código cliente e facilitando a manutenção e evolução do sistema. Ele melhora a legibilidade, promove o desacoplamento e centraliza a lógica de interação, tornando o desenvolvimento e a manutenção de software mais eficientes e menos propensos a erros.

8) Discuta as desvantagens do padrão Facade e em quais situações ele pode não ser a melhor escolha.

Embora o padrão Facade ofereça uma interface simplificada e unificada para sistemas complexos, é importante considerar suas desvantagens e os contextos específicos do projeto. Em situações que exigem acesso detalhado, alta performance, modularidade extrema ou flexibilidade para mudanças frequentes, o uso do padrão Facade pode não ser a melhor escolha. Nessas circunstâncias, uma abordagem mais direta, com interações explícitas e desacopladas entre os componentes, pode ser mais adequada para atender às necessidades do sistema e garantir sua evolução e manutenção eficazes.

9) Dê um exemplo teórico de um sistema que poderia se beneficiar da implementação do padrão Facade. Descreva como a Facade simplifica a interação com o sistema.

O padrão Facade simplifica a interação com um sistema de e-commerce complexo, fornecendo uma interface unificada e de alto nível que esconde as complexidades dos serviços subjacentes. Ele promove o desacoplamento, melhora a manutenibilidade e facilita a evolução do sistema, tornando o desenvolvimento e a manutenção mais eficientes.