Teste época especial 2022

1. O GRASP (General Responsability Abstraction Software Pattern) descreve um conjunto de princípios fundamentais para um correto desenho de um determinado objeto. O princípio Low Coupling é um deles e diz que afirma que se pode reduzir o impacto de uma mudança e encorajar a reutilização de classes previamente criadas. Este princípio faz com que as classes sejam mais fáceis de compreender, quando analisadas isoladas, no entanto, em certos caso a existência de High Coupling não é necessariamente um problema.

O princípio High Cohesion pertence também aos princípios do GRASP, onde, ao se criar uma classe esta tem de ser boa ou seja, não devem ter métodos que implementem funcionalidades a mais, para além daquelas que o objetivo de criação da classe. Este princípio complementa o princípio do Low Coupling, e as suas vantagens são iguais às do Low Coupling.

2.

```
public interface Drive {
    public byte[] read(File fn);
    public boolean write(File fn, byte[] data);
}

public class NetDrive implements Drive {
    //atributes and other methods
    public byte[] read(File fn) {/* ... */}
    public boolean write(File fn, byte[] data) {/* ... */}
}

public class PenDrive implements Drive {
    public byte[] read(File fn) {/* ... */}
    public boolean write(File fn, byte[] data) {/* ... */}
}

public class TransferUtils {
    public static void copyFile(Drive usb, File orig, Drive ssd, ...) {
        byte[] data = usb.read(orig);
        //...
        ssd.write(dest, data);
    }
}
```

③ Exame Especial - 2021-2022

a)

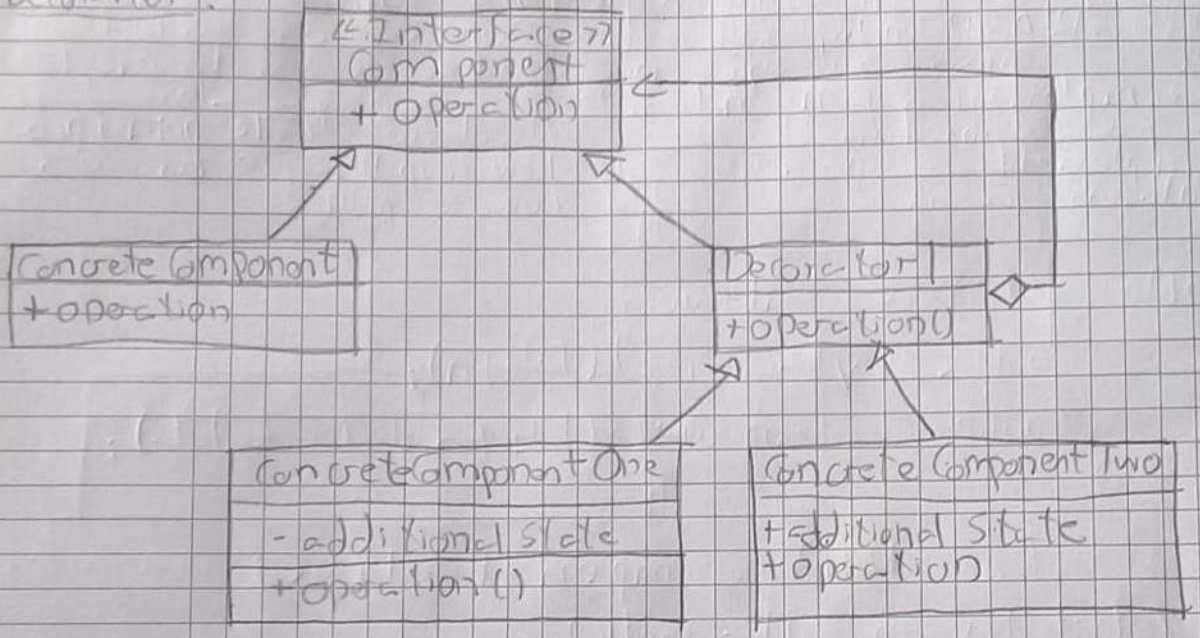
O padrão é o Strategy.

b)

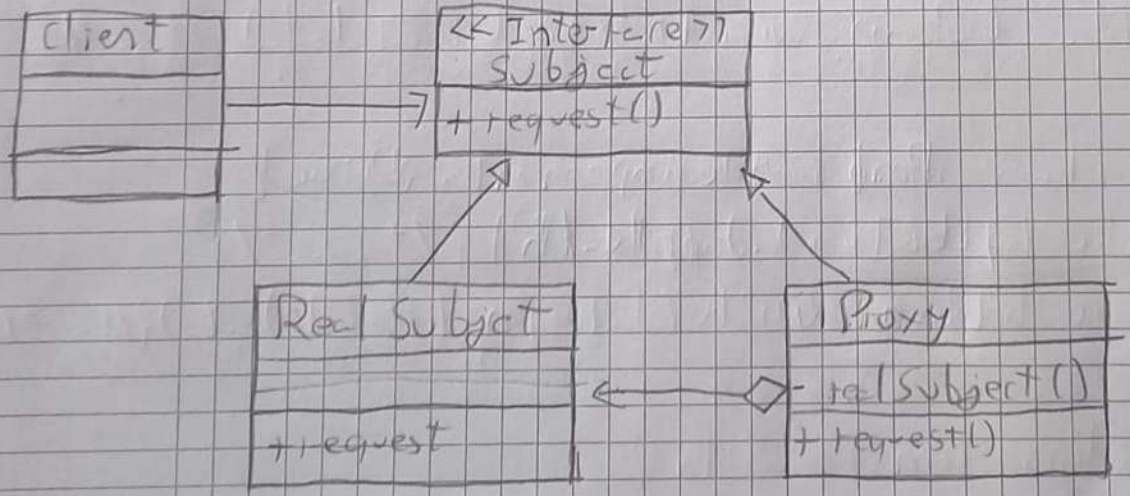
```
Context c = new Context(new ImplementationOne());
c.algorithm();
c.setStrategy(new ImplementationTwo());
c.algorithm();
```

4

Decorator:

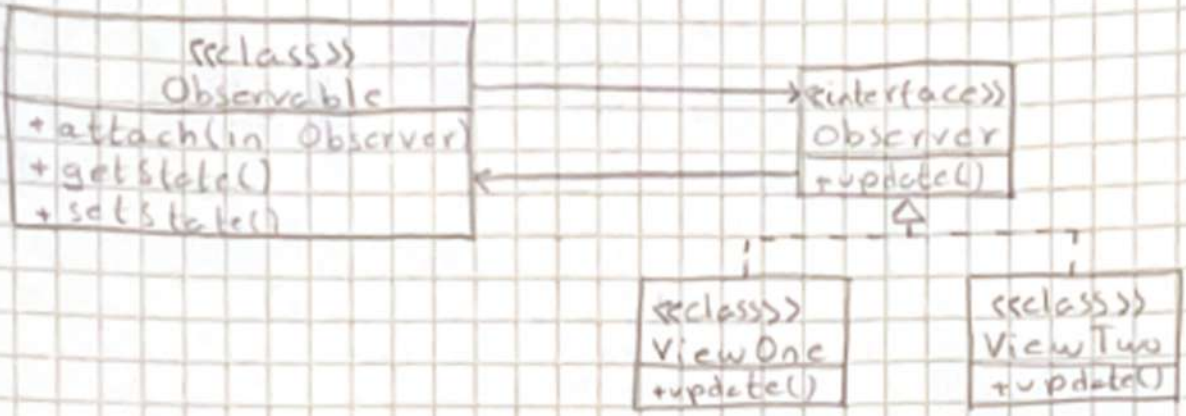


Proxy:



Teóric

(5)



Este padrão permite definir uma relação de dependência 1:N onde quando um objeto (Observable) muda de estado, os Observers são notificados para se atualizarem.

Um exemplo de utilização seria por exemplo um certo Livro (Observable) e Compradores (Observers). Em vez dos consumidores estarem constantemente a verificar a disponibilidade do Livro, podem apenas ser notificados pelo Livro quando o mesmo estiver disponível.

(6)

A arquitetura apresentada é a Layered Architecture
Esta arquitetura consiste, normalmente, em 4 camadas distintas:

- Presentation, Business, Persistence e Database.

Cada camada tem a sua funcionalidade própria.
Em geral, temos sempre de percorrer as camadas de forma sequencial à exceção de quando alguma camada se encontre aberta.

Desvantagens: Agility, Ease of Deployment, Performance e Scalability

Vantagens: Testability e Development