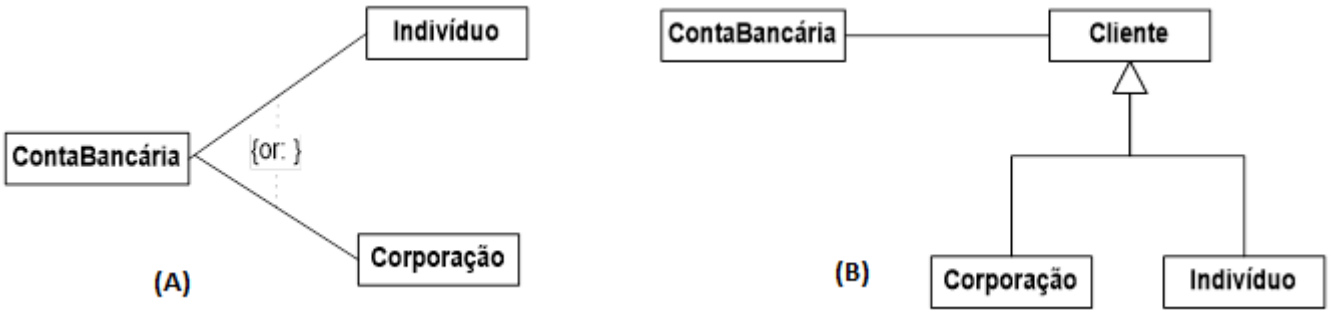
POO II - Lista de Exercícios

Padrões GRASP (General Responsibility Assignment Patterns) ou Padrões de Atribuição de Responsabilidade Geral

Questão 1 Quais são os padrões GRASP (ou padrões de responsabilidade geral)? Explique de maneira suscinta, cada um deles.

Questão 2 Considere os diagramas de classes de análise fornecidos na Figura 1, nos itens (a) e (b) abaixo, ambos de acordo com a notação da UML. Esses diagramas desejam representar o fato de que uma conta bancária pode estar associada a uma pessoa, que pode ser ou uma pessoa física (representada pela classe Indivíduo), ou uma pessoa jurídica (representada pala classe Corporação). Uma dessas duas soluções é melhor que a outra? Se sim, qual delas e em que sentido? Justifique sua resposta considerando alguns dos padrões GRASP.



Figuur 1: Diagramas com as duas soluções.

Questão 3 Qual a responsabilidade correta relacionada ao padrão GRASP Creator (Criador)?

Padrões GoF

Questão 4 Qual o objetivo dos padrões Comportamentais, segundo o catálogo GOF?

Questão 5 Qual o objetivo dos padrões Estruturais, segundo o catálogo GOF?

Questão 6 Qual o objetivo dos padrões de Criação, segundo o catálogo GOF?

Questão 7 Assinale a opção cujos padrões de projeto estão todos classificados como Comportamentais, segundo o catálogo GoF:

- A) Command, Bridge, Iterator, Mediator, Observer, State, Strategy
- B) Command, Bridge, Iterator, Mediator, Bridge, State, Strategy
- C) Command, Bridge, Iterator, Mediator, Composite, State, Strategy
- D) Command, Interpreter, Iterator, Mediator, Observer, State, Strategy

- E) Command, Interpreter, Iterator, Mediator, composite, State, Strategy

Questão 8 Assinale a opção cujos padrões de projeto estão todos classificados como criação, segundo o catálogo GoF:

- A) Command, Builder, Factory Method, Protype , Singleton
- B) Abstract Factory, Builder, Factory Method, Decorator, Singleton
- C) Abstract Factory, Builder, Factory Method, Protype, Singleton
- D) Abstract Factory, Builder, Composite, Protype, Singleton
- E) Abstract Factory, Bridge, Factory Method, Protype, Singleton

Questão 9 Assinale a opção cujos padrões de projeto estão todos classificados como Estruturais, segundo o catálogo GoF:

- A) Adapter, Bridge, Composite, Decorator, Façade, Flyweight, Singleton
- B) Adapter, Bridge, Protype, Decorator, Façade, Flyweight, Singleton
- C) Singleton, Bridge, Composite, Decorator, Façade, Flyweight, Proxy
- D) Singleton, Bridge, Protype, Decorator, Façade, Flyweight, Proxy
- E) Adapter, Bridge, Composite, Decorator, Façade, Flyweight, Proxy

Questão 10 O que é o MVC e como ele é estruturado?

Questão 11 Baixo acoplamento é um princípio-chave na Orientação objetos. O que deve ser feito para garantir o baixo acoplamento na relação entre classes e objetos de um sistema?

Questão 12 Em uma janela pode-se adicionar objetos como barras de rolagem, caixas de texto, labels, etc. Pode-se criar uma classe Auxiliar que será estendida pelos decoradores que irão inserir propriedades na janela. O padrão de projetos que possibilita essa implementação chama-se:

- A) Memento
- B) Template Method
- C) Decorator
- D) Prototype
- E) Builder

Questão 13 Dois dos principais patterns utilizados atualmente são descritos a seguir:

- I. Visa garantir que uma classe só tenha uma única instância e prover um ponto de acesso global a ela.
- II. Visa definir uma dependência um-para-muitos entre objetos para que quando um objeto mudar de estado os seus dependentes sejam notificados e atualizados automaticamente.

Os Design Patterns descritos em I e II são, respectivamente:

- A) Singleton e Observer.
- B) Composite e Adapter.

- C) Singleton e Command.
- D) Facade e Observer.
- E) Facade e Adapter.

Questão 14 O padrão de projetos cuja finalidade é oferecer uma interface de acesso a todos os objetos da aplicação é ? e pertence a categoria de padrões do catálogo GOF chamada ?. A resposta correta é:

- A) Observer – Criação
- B) Facade – Estrutural
- C) Observer – Estrutural
- D) Facade – Criação
- E) Observer – Comportamental

Questão 15 Qual o objetivo dos padrões Estruturais, segundo o catálogo GOF?

Abstract Factory

Questão 16 Crie um Hello, World que utilize o padrão Abstract Factory para escolher dentre duas formas de impressão:

- (a) na tela ou;
- (b) num arquivo chamado output.txt. Seu programa deve escolher dentre as duas fábricas aleatoriamente.

Questão 17 Considere os seguintes conceitos do mundo real: pizzaria, pizzaiolo, pizza, consumidor. Considere ainda que em uma determinada pizzaria, dois pizzaiolos se alternam. Um deles trabalha segundas, quartas e sextas e só sabe fazer pizza de calabresa (queijo + calabresa + tomate), o outro trabalha terças, quintas e sábados e só sabe fazer pizza de presunto (queijo + presunto + tomate). A pizzaria fecha aos domingos. Tente mapear os conceitos acima para o padrão Abstract Factory (hierarquia de fábricas, hierarquia de produtos, cliente) e implemente um programa que receba uma data como parâmetro (formato dd/mm/yyyy) e imprima os ingredientes da pizza que é feita no dia ou, se a pizzaria estiver fechada, informe isso na tela. Agora imagine que a pizzaria agora faz também calzones (novamente, de calabresa ou presunto). Complemente a solução com mais este componente.

Builder

Questão 18 - Na cadeia de restaurantes fast-food PatternBurgers há um padrão para montagem de lanches de crianças. O sanduíche (hambúrguer ou cheeseburger), a batata (pequena, média ou grande) e o brinquedo (carrinho ou bonequinha) são colocados dentro de uma caixa e o refrigerante (coca ou guaraná) é entregue fora da caixa. A classe abaixo é dada para representar o pedido de um consumidor:

```
import java.util.*;
public class Pedido {
private Set<String> dentroDaCaixa = new HashSet<String>();
private Set<String> foraDaCaixa = new HashSet<String>();
public void adicionarDentroDaCaixa(String item) {
dentroDaCaixa.add(item);
}
public void adicionarForaDaCaixa(String item) {
foraDaCaixa.add(item);
```

```
}
public String toString() {
StringBuffer buffer = new StringBuffer();
buffer.append("Seu pedido:\n");
buffer.append("Dentro da caixa:\n");
for (String item : dentroDaCaixa) buffer.append("\t" + item + "\n");
buffer.append("Fora da caixa:\n");
for (String item : foraDaCaixa) buffer.append("\t" + item + "\n");
buffer.append("\nTenha um bom dia!\n\n");
return buffer.toString();
}
}
```

Neste caso, o padrão Builder pode ser usado para separar as tarefas do atendente e do funcionário que monta o pedido. Somente este último sabe como montar os pedidos segundo os padrões da empresa, mas é o atendente quem lhe informa quais itens o consumidor pediu. Implemente a simulação do restaurante fast-food descrita acima utilizando o padrão Builder e escreva uma classe cliente que pede um lanche ao atendente, recebe-o do outro funcionário e imprime o pedido.

Builder

Questão 19 - Escreva classes para satisfazer os seguintes papéis do padrão Builder:

```
* Cliente: recebe como parametros o nome, endereco, telefone e e-mail de uma pessoa,
solicita ao diretor que construa informacoes de contato, recupera a informacao do builder
e imprime;
* Diretor: recebe como parametro o builder a ser utilizado e os dados de contato.
Manda o builder construir o contato;
* Builder: constroi o contato. Existem tres tipos de contato e um builder para cada tipo:
  ContatoInternet: armazena nome e e-mail;
  ContatoTelefone: armazena nome e telefone;
  ContatoCompleto: armazena nome, endereco, telefone e e-mail.
A classe que representa o papel client deve ter o metodo main() que ira criar um director
e um builder de cada tipo. Em seguida, deve pedir ao director que crie um contato
de cada tipo e imprimi-los (use o toString() da classe que representa a informacao
de contato).
```

Factory Method

Questão 20 - Construa um programa que receba como parâmetro um ou mais nomes, cada um podendo estar em um dos seguintes formatos:
§ ”nome sobrenome”;
§ ”sobrenome, nome”.
Escreva duas aplicações de construção de nomes, uma para cada formato. Cada uma deve ser responsável por armazenar os nomes criados e imprimi-los quando requisitado. Implemente o padrão Factory Method de forma que somente a criação do nome seja delegada às aplicações concretas (subclasses). Seu programa deve criar as duas aplicações e construir objetos da classe Nome, que deve ter propriedades nome e sobrenome para armazenar as informações em separado. Os nomes não precisam ser impressos em ordem.
Ex.:

```
java Nomes "McNealy, Scott" "James Gosling" "Naughton, Patrick"
James Gosling
Scott McNealy
Patrick Naughton
```

Questão 21 - Crie dois arquivos texto em um diretório qualquer:

```
publico.txt
```

Estas são informações públicas sobre qualquer coisa. Todo mundo pode ver este arquivo.
confidencial.txt
Estas são informações confidenciais, o que significa que você provavelmente sabe a palavra secreta!
Usando o padrão Factory Method, crie duas provedoras de informação: uma que retorna informações públicas e outra que retorna informações confidenciais. Utilize o provedor confidencial se o usuário informar a senha "designpatterns" como parâmetro para o programa, que deve recuperar a informação e exibi-la na tela.

Questão 22 - Escreva um programa que conte até 10 e envie os números para uma ferramenta de log. Esta ferramenta de log deve ser construída por uma fábrica. Utilize Factory Method para permitir a escolha entre dois tipos de log: em arquivo (log.txt) ou diretamente no console. A escolha deve ser por um parâmetro passado ao programa ("arquivo" ou "console").

Singleton

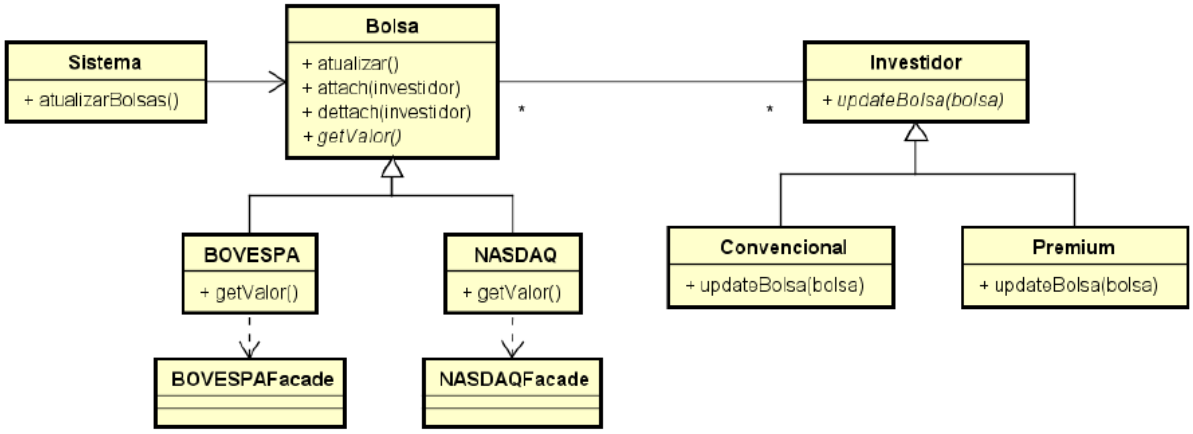
Questão 23 - Escreva, compile e execute o programa abaixo. Em seguida, troque sua implementação para que a classe Incremental seja Singleton. Execute novamente e veja os resultados.

```
class Incremental {
    private static int count = 0;
    private int numero;
    public Incremental() {
        numero = ++count;
    }

    public String toString() {
        return "Incremental " + numero;
    }
}

public class TesteIncremental {
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) {
            Incremental inc = new Incremental();
            System.out.println(inc);
        }
    }
}

class Incremental {
    private static int count = 0;
    private static Incremental singleton;
    private Incremental() {

    }
    public static Incremental getInstancia(){
        ++count;
        if(singleton==null)singleton = new Incremental();
        return singleton;
    }
    public String toString() {
        return "Incremental " + count;
    }
}
```

```
public class TesteIncremental {
    public static void main(String[] args) {
        for (int i = 0; i < 10; i++) {
            Incremental inc = Incremental.getInstancia();
            System.out.println(inc);
        }
    }
}
```

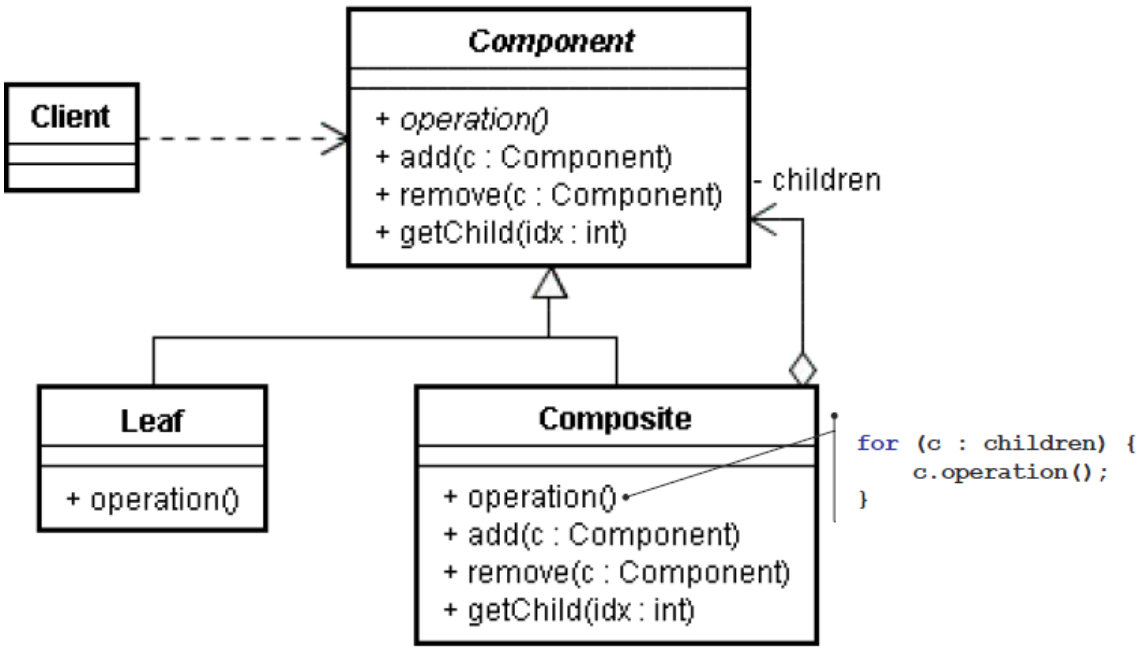
Identifique os padrões abaixo pelas soluções propostas. Cite o padrão de maneira completa: nome, problema, solução...

Questão 24 - Qual é o padrão proposto pela solução da Figura 2 abaixo?



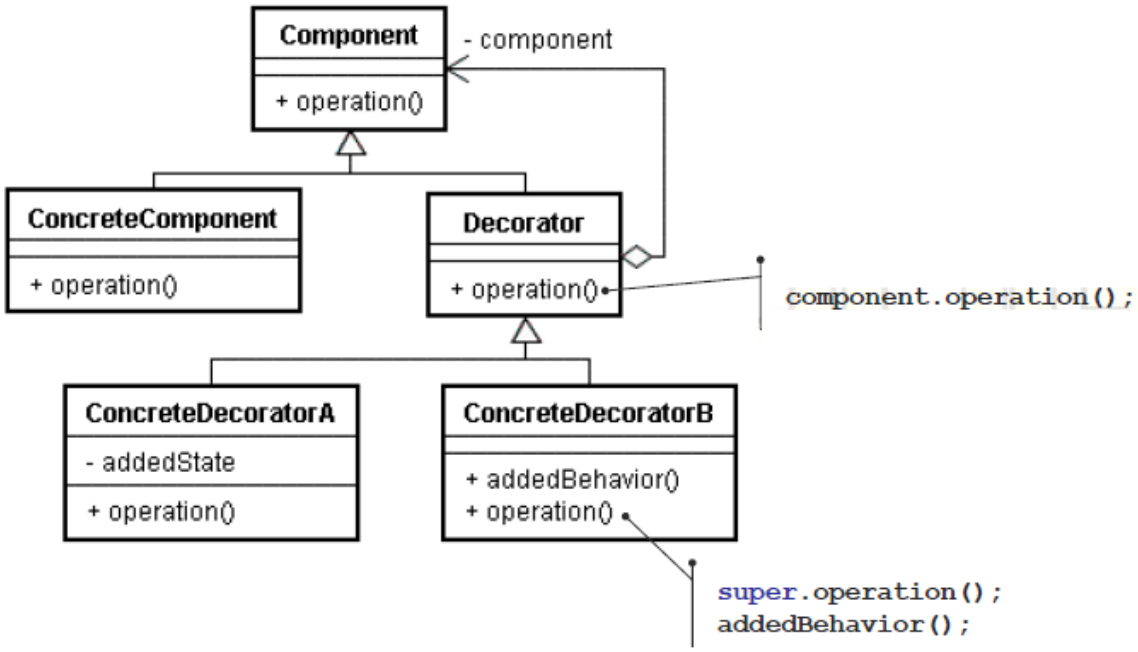
Figuur 2: Solução proposta por um padrão de projeto.

Questão 25 - Qual é o padrão proposto pela solução da Figura 3 abaixo?



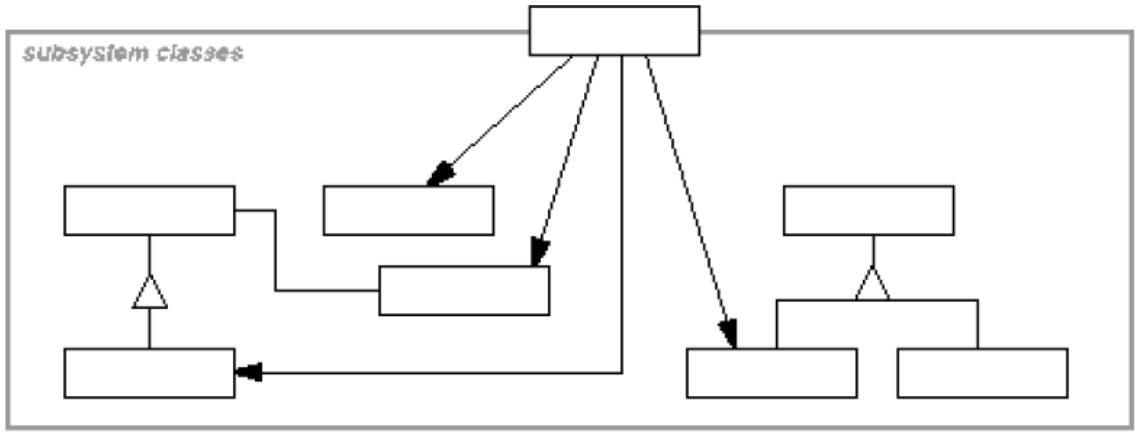
Figuur 3: Solução proposta por um padrão de projeto.

Questão 26 - Qual é o padrão proposto pela solução da Figura 4 abaixo?



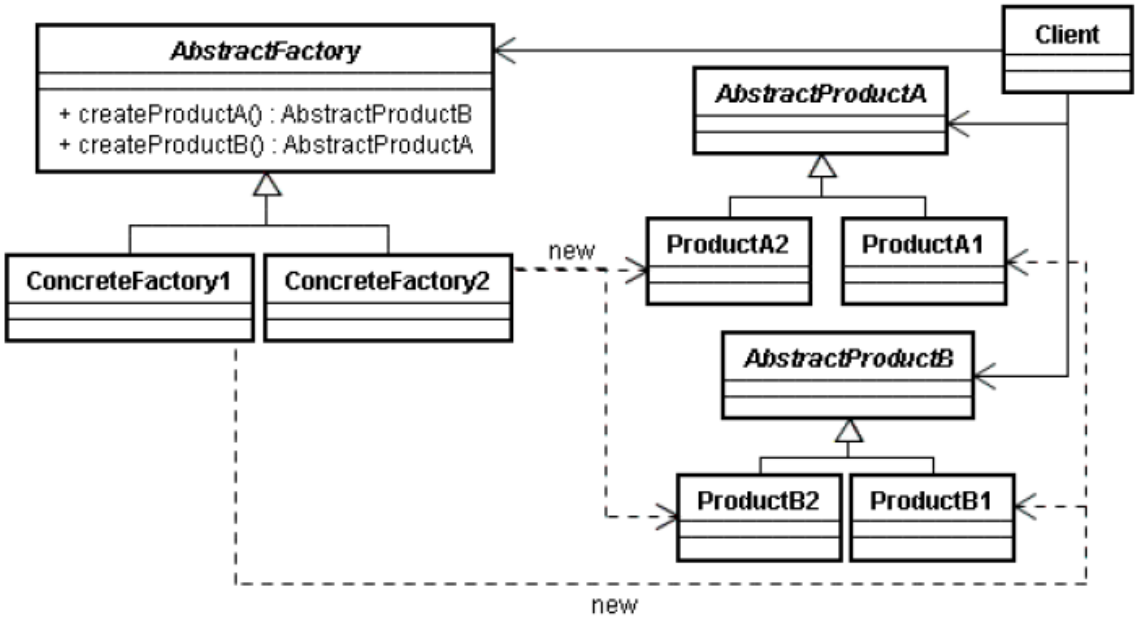
Figuur 4: Solução proposta por um padrão de projeto.

Questão 27 - Qual é o padrão proposto pela solução da Figura 5 abaixo?



Figuur 5: Solução proposta por um padrão de projeto.

Questão 28 Qual é o padrão proposto pela solução da Figura 6 abaixo?



Figuur 6: Solução proposta por um padrão de projeto.