2024/02 - ENGENHARIA DE SOFTWARE II

PADRÕES DE PROJETO

Camile Weber e Cassiano Sobierai

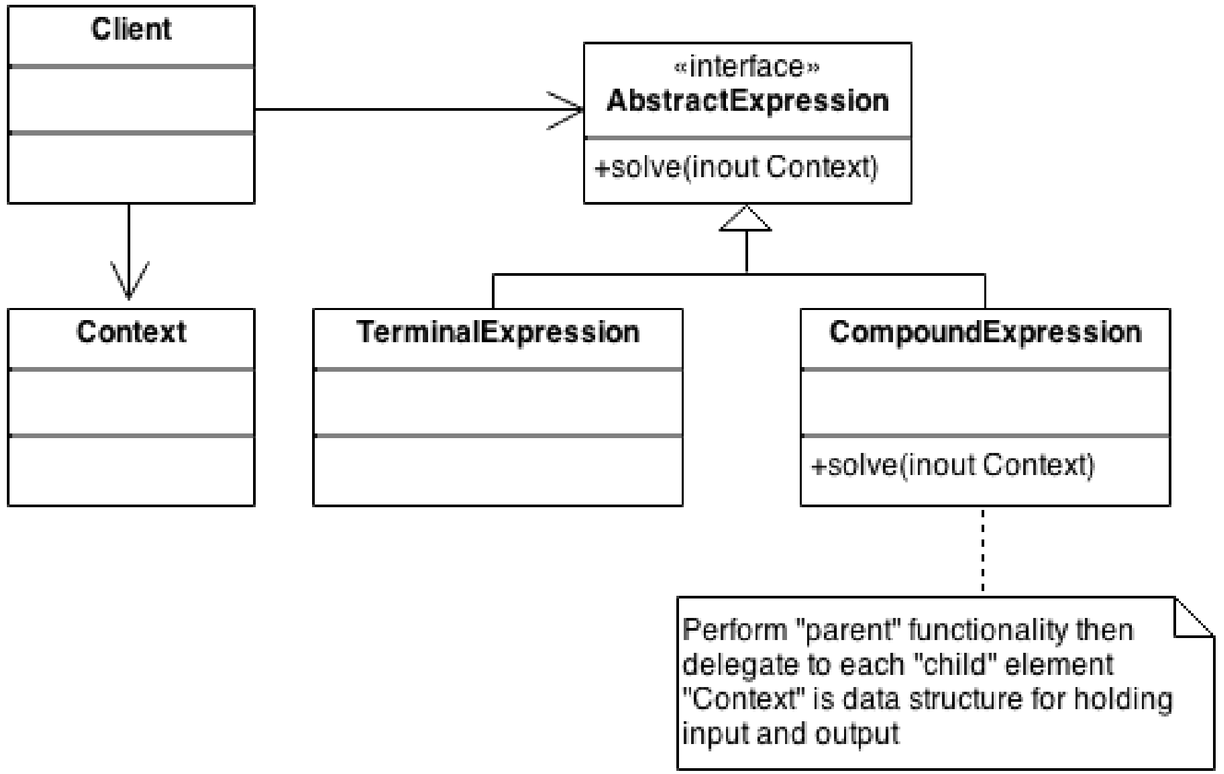
04 DE NOVEMBRO DE 2024

INTERPRETER (GOF)

PADRÃO INTERPRETER

- Fornece uma maneira de avaliar ou interpretar expressões em uma linguagem;
- Geralmente implementa uma gramática;
- Usado quando precisamos analisar e executar operações baseadas em regras definidas por essa linguagem.

O PADRÃO ENVOLVE A CRIAÇÃO DE UMA ÁRVORE DE EXPRESSÃO, ONDE CADA NÓ DA ÁRVORE REPRESENTA UMA REGRA NA GRAMÁTICA.



COMPONENTES DO PADRÃO

CLIENT

O cliente que constrói a árvore de expressões a partir de uma linguagem ou gramática e chama o método interpret() para avaliar.

CONTEXT

Contém informações globais que são necessárias para a interpretação, como variáveis e valores.

NONTERMINAL

Expressões que são compostas por outras expressões, como operações matemáticas ou lógicas que combinam outras expressões.

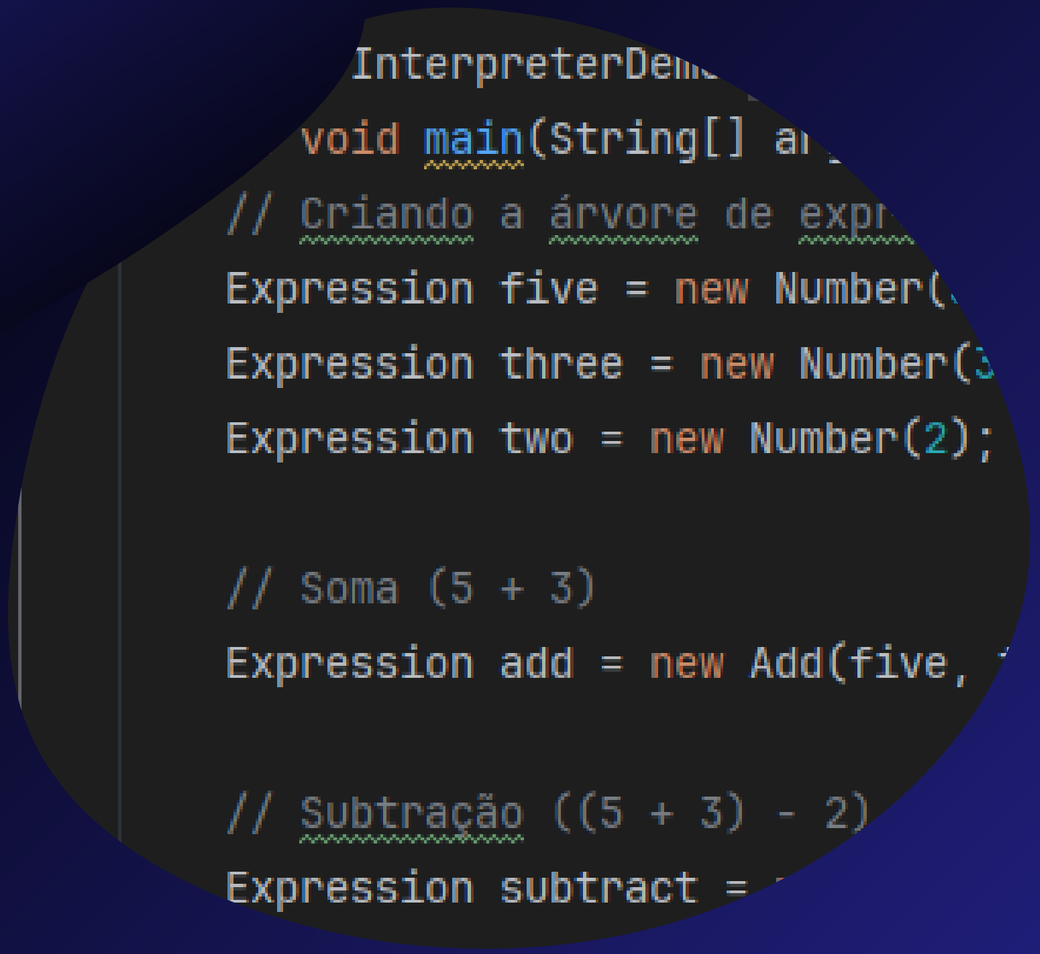
COMPONENTES DO PADRÃO

TERMINAL

Representa as expressões que não podem ser divididas em subexpressões, como números ou variáveis.

EXPRESSION

Interface comum que declara o método interpret(), o qual todas as expressões (simples ou compostas) devem implementar.



EXEMPLO DE CODIFICAÇÃO

Interpretando Expressões Matemáticas

ABSTRACT EXPRESSION

```
public interface Expression {  16 usages  3 implementations
    //deve ser implementado por todas as classes de expressões
        int interpret();  5 usages  3 implementations
}
```

TERMINAL EXPRESSION

```java
public class Number implements Expression {  3 usages
    private int number;  2 usages

    //representa uma expressão terminal
    public Number(int number) {  3 usages
        this.number = number;
    }

    @Override  5 usages
    public int interpret() {
        return this.number;
        //apenas retorna o numero
    }
}
```

NON-TERMINAL EXPRESSION

Para soma e subtração

```java
public class Add implements Expression {  1 usage
    private Expression leftExpression;  2 usages
    private Expression rightExpression;  2 usages

    //representa uma expressão não terminal
    //executa a soma de duas contas
    public Add(Expression leftExpression, Expression rightExpression) {
        this.leftExpression = leftExpression;
        this.rightExpression = rightExpression;
    }

    @Override  5 usages
    public int interpret() {
        return leftExpression.interpret() + rightExpression.interpret();
        //retorna a soma
    }
}
```

```
public class InterpreterDemo {  no usages
    public void main(String[] args) {
        // Criando a árvore de expressão: (5 + 3) - 2
        Expression five = new Number(5);
        Expression three = new Number(3);
        Expression two = new Number(2);

        // Soma (5 + 3)
        Expression add = new Add(five, three);

        // Subtração ((5 + 3) - 2)
        Expression subtract = new Subtract(add, two);

        // Interpretando a expressão
        System.out.println("Resultado: " + subtract.interpret());
    }
}
```

CLIENT

INTERPRET() = PEÇA-CHAVE

AO ADICIONAR NOVAS **EXPRESSÕES**, BASTA IMPLEMENTAR O MÉTODO *INTERPRET()* DE ACORDO COM A LÓGICA DESEJADA, SEM MODIFICAR O RESTANTE DO CÓDIGO.
ELE QUE EXECUTA A **LÓGICA** DA EXPRESSÃO.

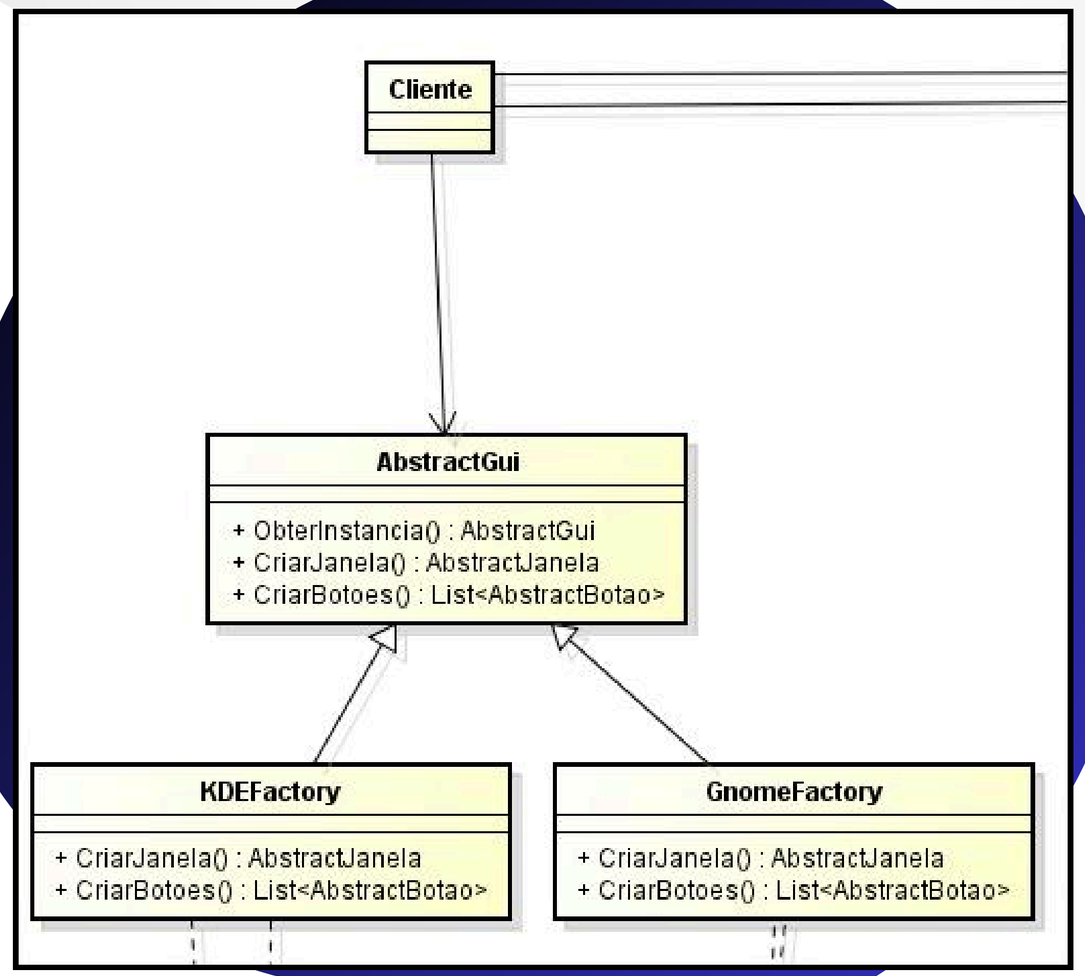
ABSTRACT FACTORY (GOF)

PADRÃO ABSTRACT FACTORY

- O padrão fornece uma interface para criação de famílias de objetos relacionados ou dependentes, sem especificar classes concretas.
- Seu objetivo é facilitar a criação e troca de famílias de objetos, permitindo flexibilidade e reduzindo o acoplamento entre a aplicação e as classes concretas.

A APLICAÇÃO NÃO CHAME DIRETAMENTE AS CLASSES CONCRETAS E QUE A ADIÇÃO OU REMOÇÃO DE UM TIPO DE INTERFACE DÊ O MENOR TRABALHO POSSÍVEL

IDEIA



COMPONENTES DO PADRÃO

FACTORY INTERFACE

Define uma interface para criar produtos de diferentes tipos sem especificar suas implementações.

CONCRETE FACTORIES

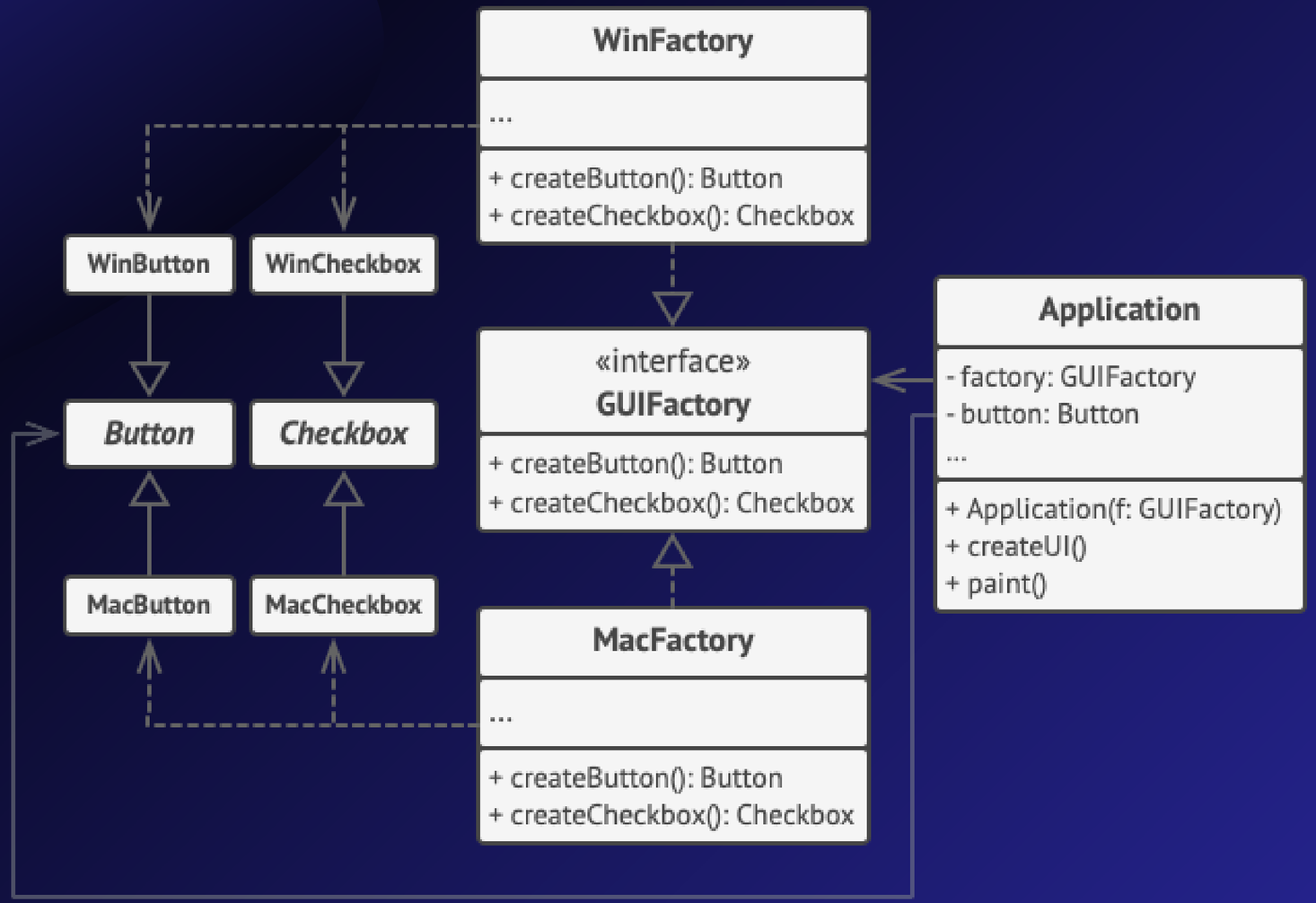
Implementam a interface da fábrica abstrata e criam os produtos específicos para cada tipo.

ABSTRACT PRODUCTS

Interfaces ou classes base para diferentes tipos de produtos que podem ser criados.

CONCRETE PRODUCTS

Implementações específicas dos produtos abstratos que são criados pelas fábricas concretas.



EXEMPLO DE CODIFICAÇÃO

Pseudocódigo

APP.PY

```python
# Cliente que usa a Fábrica Abstrata
class Application:
    def __init__(self, factory: GUIFactory):
        self.button = factory.create_button()
        self.checkbox = factory.create_checkbox()

    def render_interface(self):
        print(self.button.render())
        print(self.checkbox.check())

# Função principal para executar o programa
def main():
    os_type = input("Escolha o sistema operacional (Windows/Mac): ").strip().lower()

    if os_type == "windows":
        factory = WindowsFactory()
    elif os_type == "mac":
        factory = MacFactory()
    else:
        print("Sistema operacional não suportado.")
        return

    app = Application(factory)
    app.render_interface()
```

FACTORIES.PY

```python
# Interface da Fábrica Abstrata
class GUIFactory(ABC):
    @abstractmethod
    def create_button(self) -> Button:
        pass

    @abstractmethod
    def create_checkbox(self) -> Checkbox:
        pass
```

```python
# Fábrica Concreta para Windows
class WindowsFactory(GUIFactory):
    def create_button(self) -> Button:
        return WindowsButton()

    def create_checkbox(self) -> Checkbox:
        return WindowsCheckbox()
```

O MESMO CÓDIGO
PARA O MAC

PRODUCTS.PY

```python
# Produtos Abstratos
class Button(ABC):
    @abstractmethod
    def render(self) -> str:
        pass

class Checkbox(ABC):
    @abstractmethod
    def check(self) -> str:
        pass
```

```python
# Produtos Concretos para Windows
class WindowsButton(Button):
    def render(self) -> str:
        return "Renderizando botão estilo Windows."

class WindowsCheckbox(Checkbox):
    def check(self) -> str:
        return "Checkbox estilo Windows checado."
```

O MESMO CÓDIGO
PARA O MAC

REFERÊNCIAS

https://medium.com/@rajeshvelmani/understanding-language-interpretation-with-the-interpreter-design-pattern-in-java-b2a3969eaf9. Acesso em 01 nov. 2024

https://www.geeksforgeeks.org/interpreter-design-pattern/. Acesso em 01 nov. 2024

https://www.devmedia.com.br/padrao-abstract-factory/23030/. Acesso em 02 nov. 2024

https://refactoring.guru/pt-br/design-patterns/abstract-factory/. Acesso em 02 nov. 2024

MUITO OBRIGADO!