Padrões de Projeto

Trabalho prático 04

Christian Marlon Souza Couto
Luana Almeida Martins

Padrão de Projeto Escolhido

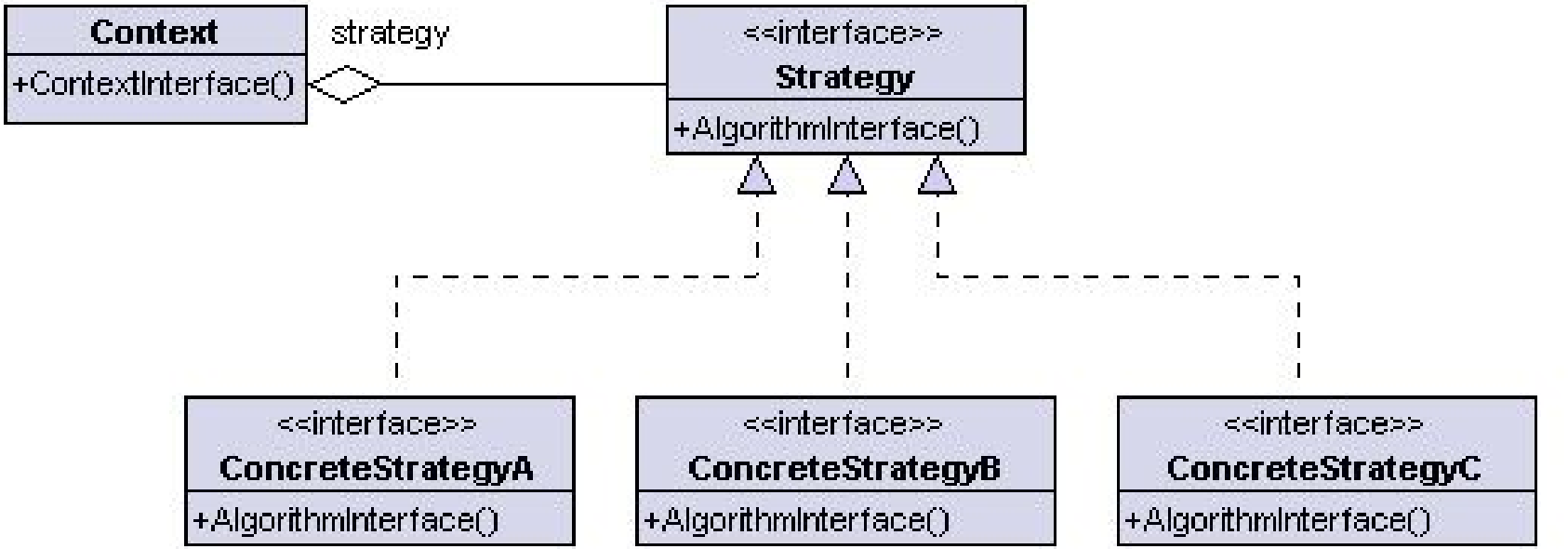
- *Strategy*
- *Abstract Factory*

Padrão *Strategy* (GAMMA, 1995)

- O padrão Strategy define uma família de algoritmos, encapsula cada um deles e os torna intercambiáveis.

- O Strategy deixa o algoritmo variar independente dos clientes que os utilizam.

Padrão *Strategy*



Padrão *Strategy*

```
public double calcularPreco(int distancia) {
    double preco = 0;
    if (TipoFrete.NORMAL.equals(tipo)) {
        preco = distancia * 1.25 + 10;
    } else if (TipoFrete.SEDEX.equals(tipo)) {
        preco = distancia * 1.45 + 12;
    }
    return preco;
}
```

Padrão *Strategy*

```
public interface Frete {

    public double calcularPreco(int distancia);
}
```

Padrão *Strategy*

```
public class Normal implements Frete {

    public double calcularPreco(int distancia) {
        return distancia * 1.25 + 10;
    }
}
public class Sedex implements Frete {

    public double calcularPreco(int distancia) {
        return distancia * 1.45 + 12;
    }
}
```

Padrão *Abstract Factory*

```
public static void main(String[] args) {
        int distancia = 100;
        int opcaoFrete = 1;
        TipoFrete tipoFrete = TipoFrete.values()[opcaoFrete - 1];

        Frete frete = null;
        if(tipoFrete.equals(TipoFrete.NORMAL)) {
            frete = new Normal();
        }
        if(tipoFrete.equals(TipoFrete.SEDEX)) {
            frete = new Normal();
        }
    }
}
```

Padrão *Abstract Factory* (GAMMA, 1995)

- Permite a criação de famílias de objetos relacionados ou dependentes.

- Por meio de uma única interface e sem que a classe concreta seja especificada.

Padrão *Abstract Factory*

```
public enum TipoFrete {
    NORMAL {
        @Override
        public Frete obterFrete() {
            return new Normal();
        }},
    SEDEX {
        @Override
        public Frete obterFrete() {
            return new Sedex();
        }};
    public abstract Frete obterFrete();
}
```

Padrão *Abstract Factory*

```
public static void main(String[] args) {
    try (Scanner entrada = new Scanner(System.in)) {
        System.out.print("Informe a distÃ¢ncia: ");
        int distancia = entrada.nextInt();
        System.out.print("Qual o tipo de frete (1) Normal, (2) Sedex: ");
        int opcaoFrete = entrada.nextInt();
        TipoFrete tipoFrete = TipoFrete.values()[opcaoFrete - 1];

        Frete frete = tipoFrete.obterFrete();
        double preco = frete.calcularPreco(distancia);
        System.out.printf("O valor total é de R$%.2f", preco);
    }
}
```

Implementação