

Padrões de Projeto: Chain of Responsibility & Factory Method

Bruno Vinícius

Padrões de Projeto

Padrões de Projeto

O que são?

- Soluções Reutilizáveis
- Categorias
- Comunicação Eficiente

O que são os Padrões de Projeto

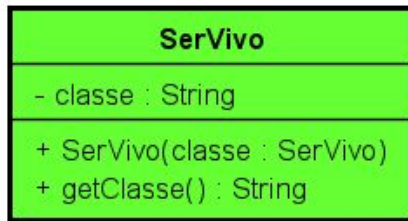
- Soluções Reutilizáveis: São soluções comprovadas para problemas comuns em design de software, aumentando a eficiência do desenvolvimento.
- Categorias: Existem três tipos - padrões de criação, estruturais e comportamentais, cada um lidando com diferentes aspectos do design de software.
- Comunicação Eficiente: Facilitam a comunicação entre desenvolvedores, pois fornecem um vocabulário comum para situações complexas de design.

Conclusão

Padrões de projeto são soluções reutilizáveis para problemas comuns no design de software. Eles são categorizados em criação, estruturais e comportamentais, lidando com diferentes aspectos do design. Além disso, facilitam a comunicação entre desenvolvedores, fornecendo um vocabulário comum para situações complexas de design. Em resumo, são uma ferramenta essencial para qualquer desenvolvedor de software.

Chain of Responsibility

- O padrão de projeto Chain of Responsibility é uma forma de organizar o código que envolve a comunicação entre objetos, evitando o acoplamento entre eles.
- Ele consiste em uma cadeia de objetos que podem receber e processar uma solicitação, passando-a para o próximo objeto da cadeia até que ela seja atendida ou rejeitada.
- Cada objeto da cadeia tem uma lógica específica para lidar com a solicitação, e pode ser substituído por outro objeto sem afetar o funcionamento do sistema.



```

public abstract class AbstractDietaHandler {
    protected AbstractDietaHandler nextHandler;

    public void setNextHandler(AbstractDietaHandler handler) {
        this.nextHandler = handler;
    }

    public void processar(SerVivo serVivo) {
        if (nextHandler != null) {
            nextHandler.processar(serVivo);
        }
    }
}

```

```

public class SerVivo {

    private String classe;

    public SerVivo(String classe) {
        this.classe = classe;
    }

    public String getClasse() {
        return classe;
    }
}

```

```

public class CarnivoroHandler
    extends AbstractDietaHandler {
    public void processar(SerVivo serVivo) {
        if (serVivo.getClasse().equals("carnivoro")) {
            System.out.println("Segundo Ser Vivo:\n"
                + "O ser vivo é um carnívoro.");
        } else {
            super.processar(serVivo);
        }
    }
}

```

```

public class HerbivoroHandler
    extends AbstractDietaHandler {
    public void processar(SerVivo serVivo) {
        if (serVivo.getClasse().equals("herbivoro")) {
            System.out.println("Primeiro Ser Vivo:\n"
                + "O ser vivo é um herbívoro.");
        } else {
            super.processar(serVivo);
        }
    }
}

```

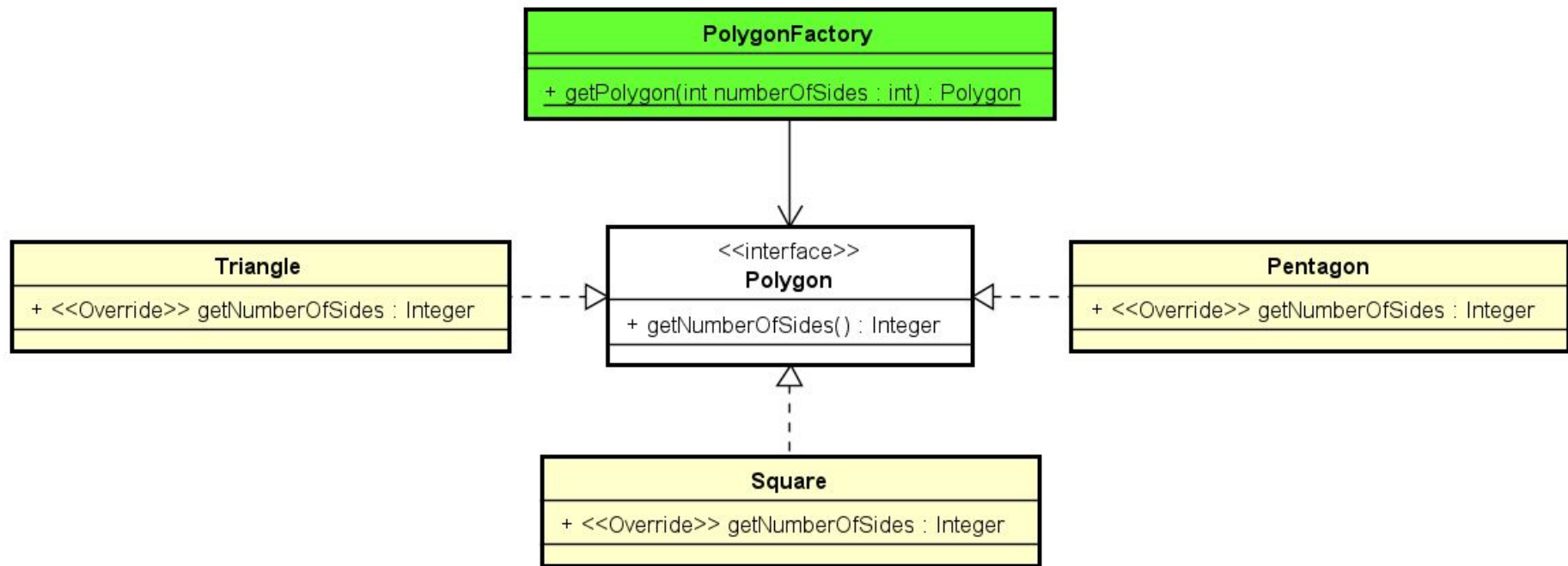
```

public class OnivoroHandler
    extends AbstractDietaHandler {
    public void processar(SerVivo serVivo) {
        if (serVivo.getClasse().equals("onivoro")) {
            System.out.println("Terceiro Ser Vivo:\n"
                + "O ser vivo é um onívoro.");
        } else {
            super.processar(serVivo);
        }
    }
}

```


Factory Method

- O Factory Method encapsula a criação de objetos. Isso significa que o código que usa a classe não precisa saber sobre as classes concretas, apenas sobre a interface ou classe abstrata.
- O Factory Method delega a responsabilidade de instanciar a classe para subclasses. Isso é feito através de um método, que é geralmente definido em uma interface ou implementado de forma padrão em uma classe abstrata.
- O Factory Method permite maior flexibilidade e reutilização do código. As subclasses podem substituir o método fábrica para alterar a classe de objetos que serão criados.



```
public interface Polygon {  
    Integer getNumberOfSides();  
}  
  
public class PolygonFactory {  
    public static Polygon getPolygon(int numberOfSides) {  
        if (numberOfSides == 3) {  
            return new Triangle();  
        } else if (numberOfSides == 4) {  
            return new Square();  
        } else if (numberOfSides == 5) {  
            return new Pentagon();  
        } else {  
            throw new IllegalArgumentException("Número ruim de lados");  
        }  
    }  
}
```

```
public class Triangle implements Polygon {  
    @Override  
    public Integer getNumberOfSides() {  
        return 3;  
    }  
}
```

```
public class Square implements Polygon {  
    @Override  
    public Integer getNumberOfSides() {  
        return 4;  
    }  
}
```

```
public class Pentagon implements Polygon {  
    @Override  
    public Integer getNumberOfSides() {  
        return 5;  
    }  
}
```

CHAIN OF RESPONSIBILITY:

Primeiro Ser Vivo:

O ser vivo é um herbívoro.

Segundo Ser Vivo:

O ser vivo é um carnívoro.

Terceiro Ser Vivo:

O ser vivo é um onívoro.

FIM DO CHAIN OF RESPONSIBILITY

FACTORY METHOD

p1: 3

p2: 4

p3: 5

FIM DO FACTORY METHOD

```
public static void main(String[] args) {  
    // CHAIN OF RESPONSIBILITY  
    System.out.println("CHAIN OF RESPONSIBILITY:\n\n");  
    AbstractDietaHandler herbivoro = new HerbivoroHandler();  
    AbstractDietaHandler carnivoro = new CarnivoroHandler();  
    AbstractDietaHandler onivoro = new OnivoroHandler();  
  
    herbivoro.setNextHandler(carnivoro);  
    carnivoro.setNextHandler(onivoro);  
  
    SerVivo serVivo1 = new SerVivo("herbivoro");  
    SerVivo serVivo2 = new SerVivo("carnivoro");  
    SerVivo serVivo3 = new SerVivo("onivoro");  
  
    herbivoro.processar(serVivo1);  
    herbivoro.processar(serVivo2);  
    herbivoro.processar(serVivo3);  
    System.out.println("FIM DO CHAIN OF RESPONSIBILITY\n\n");  
    // FIM DO CHAIN OF RESPONSIBILITY  
  
    // FACTORY METHOD  
    System.out.println("FACTORY METHOD\n\n");  
    Polygon p1 = PolygonFactory.getPolygon(3);  
    Polygon p2 = PolygonFactory.getPolygon(4);  
    Polygon p3 = PolygonFactory.getPolygon(5);  
  
    System.out.println("p1: " + p1.getNumberOfSides());  
    System.out.println("p2: " + p2.getNumberOfSides());  
    System.out.println("p3: " + p3.getNumberOfSides());  
    System.out.println("\nFIM DO FACTORY METHOD\n\n");  
    // FIM DO FACTORY METHOD  
}
```

GitHub do Projeto:

<https://github.com/vrunobinicius/seminario-poo-2023>

Principais Fontes:

https://youtu.be/sq4kZnS9cd4?si=hR6aVmrCg_jlDRgF

<https://youtu.be/-e9bFrcxG9E?si=LkcwaYNwvxuYnCwF>