AfroDev

Java Básico - Aula 4 - Parte 1 e 2

Classes Abstratas

- Classe marcada com a palavra chave “abstract”
- Pode conter métodos concretos ou abstratos
- Podem ser estendidas (herança)
- NÃO podem ser instanciadas
- NÃO podem ter atributos static ou final
- São, como o próprio nome diz, uma abstração de um conceito. A ideia ainda em formação, digamos…
- Cria caminho para patterns poderosos com o Template
- Compartilha código entre classes filhas (evite duplicação de código)

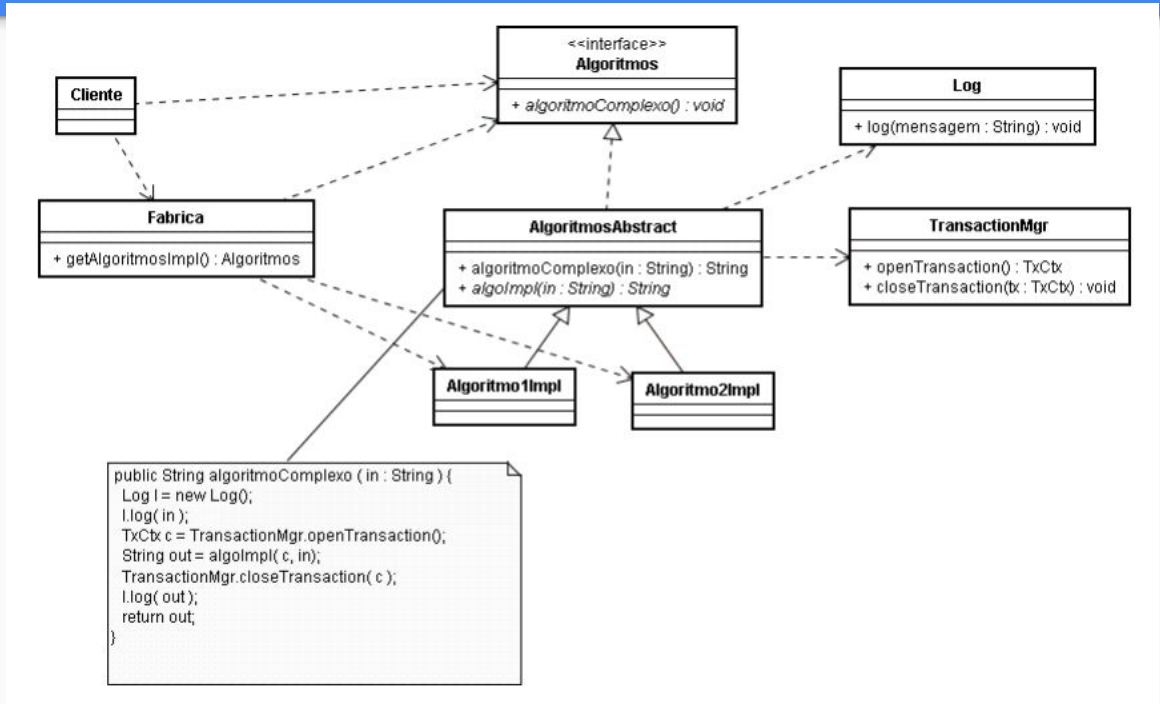
Métodos Abstratos

- É um método que é somente declarado (SEM código)

Pattern Template - Conceito e Exemplos

- http://www.vincehuston.org/dp/template_method.html
- https://java-design-patterns.com/patterns/template-method/

Pattern Template - Exemplo



Classes Abstratas

```
3  public abstract class TaxadorAbstrato {
4
5      protected abstract double cobrarIcms(double valorProduto);
6
7      public void teCobrei(double valorProduto) {
8          enviarCobranca();
9          cobrarIcms(valorProduto);
10         recebendoCobranca();
11     }
12
13     protected void enviarCobranca() {
14         System.out.println("Enviando cobranca comum a todos...");
15     }
16
17     protected void recebendoCobranca() {
18         System.out.println("Recebendo cobranca comum a todos...");
19     }
20
21 }
22
```

Classes Abstratas

```java

public class TaxadorParana extends TaxadorAbstrato {

    @Override
    protected double cobrarIcms(double valorProduto) {
        System.out.println("Cobrando 10% no PR");
        return valorProduto * 1.1;
    }

}

```

```java

public class TaxadorSaoPaulo extends TaxadorAbstrato {

    @Override
    protected double cobrarIcms(double valorProduto) {
        System.out.println("Cobrando 30% em SP");
        return valorProduto * 1.3;
    }

}

```

Classes Abstratas

```java

public class TaxadorParana extends TaxadorAbstrato {

    @Override
    protected double cobrarIcms(double valorProduto) {
        System.out.println("Cobrando 10% no PR");
        return valorProduto * 1.1;
    }

}

```

```java

public class TaxadorSaoPaulo extends TaxadorAbstrato {

    @Override
    protected double cobrarIcms(double valorProduto) {
        System.out.println("Cobrando 30% em SP");
        return valorProduto * 1.3;
    }

}

```

Classes Abstratas

```
3 public class TaxadorDemo {
4
5     public static void main(String[] args) {
6
7         TaxadorAbstrato parana = new TaxadorParana();
8         parana.teCobrei(10D);
9
10        TaxadorAbstrato saoPaulo = new TaxadorSaoPaulo();
11        saoPaulo.teCobrei(10D);
12    }
13
14 }
15
```

Problems @ Javadoc Declaration Console Coverage

```
<terminated> TaxadorDemo [Java Application] C:\Program Files\Java\jre1.8.0_171
Enviando cobranca comum a todos...
Cobrando 10% no PR
Recebendo cobranca comum a todos...
Enviando cobranca comum a todos...
Cobrando 30% em SP
Recebendo cobranca comum a todos...
```

Classes Abstratas - Exercício

- Seu chefe quer que você crie uma calculadora que faça soma e subtração
- Ele também quer que ANTES de qualquer operação você escreva no console a mensagem “Início da operação…”
- Ele também quer que DEPOIS de qualquer operação você escreve na console a mensagem “Fim da operação…”
- Ah… ele também te disse que não sabe quais outras operações vão aparecer, mas por ora será somente soma e subtração

Interfaces

Interfaces

Interfaces

- “Apenas” um contrato
- Refere-se ao “o que” e não “como”
- Pode ter um ou mais “comos” (implementações)
- Cereja do bolo dos conceitos OO
- Também não podem ser instanciadas
- Deixa o código extremamente flexível
- É por aqui que podemos ter uma base única de código de um mesmo produto, mas vender para clientes diferentes

Interfaces

- Os métodos por default são públicos (fique esperto com isso)
- Aliás… vamos discutir um pouco sobre isso...
- Interfaces como APIs
- Também nascem as specs e os vendors
- Concorrência é saudável
- Exemplos de API Java: JDBC, JPA, JMS, JCACHE, etc...

Interfaces

- Exemplo em Pattern: http://www.vincehuston.org/dp/strategy.html
- Mais um: http://www.vincehuston.org/dp/iterator.html
- Mais um: https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html

Interfaces vs Classes Abstratas

- Ambas não podem ser instanciadas
- Ambas podem conter somente a declaração de métodos
- Nas abstratas, métodos podem ser public, private ou protected
- Nas interfaces, métodos são sempre public
- Extensão pode ser feita somente uma vez (sendo abstrata ou não)
- Classe pode implementar mais de uma interface

Interfaces - Exercício

- Cria uma interface para um serviço de progressões. Dado um número inteiro qualquer, o programa deverá exibir a quantidade de elementos (igual ao número) daquela progressão.
- Independente da progressão, o console deverá exibir o resultado no mesmo formato
- Progressão aritmética com constante 1, ou seja: 1,2,3,4,5,etc (vai incrementando 1)
- Progressão geométrica com razão 2, ou seja, 1,2,4,8,16 etc (elemento x2)

Revisão

- Aprendemos um ferramenta poderosa para compartilhar código entre classes de maneira elegante e eficiente. Utilizamos classes abstratas para esse motivo e para implementarmos algo que ainda não está muito bem definido. Junto desse conceito estão os métodos abstratos
- As interfaces são a cereja do bolo do conceito OO. Elas separam o contrato da implementação. Deixam o código menos acoplado e bem mais flexível