链滴

GOF 设计模式小白教程之解释器模式

作者：valarchie

原文链接：https://ld246.com/article/1567330724643

来源网站：链滴

解释器模式（Interpreter）

定义：

给分析对象定义一个语言，并定义该语言的文法表示，再设计一个解析器来解释语言中的句子。也就说，用编译语言的方式来分析应用中的实例。这种模式实现了文法表达式处理的接口，该接口解释一特定的上下文。

通俗解释：

我们可以用一个最最简单的计算器程序来举例，该计算器只有加法和减法并且不能加括号。我们通过入一串简单的表达式例如：1+3+4-5然后根据计算得出结果为3。这其中的每个数值和符号都属于一表达式。每种表达式都对应一种功能，可以是保存数值的功能，也可以是计算数值的功能。解释器模其实我们在实际业务开发中比较少碰到，例如渲染页面的标签语言或者Spring EL　表达式也属于也于解释器模式。总得来说，解释器相当于一个傻瓜版的编译器。

代码：

表达式接口，拥有解释方法

```
public interface Expresion {
    // 解释翻译
    int interprete();

}
```

加法表达式

```
public class AddExpression implements Expresion {

    private Expresion left;
    private Expresion right;

    public AddExpression(Expresion left, Expresion right) {
        this.left = left;
        this.right = right;
    }

    @Override
    public int interprete() {
        return left.interprete() + right.interprete();
    }

}
```

减法表达式

```
public class MinusExpression implements Expresion {

    private Expresion left;
    private Expresion right;
```

```
    public MinusExpression(Expresion left, Expresion right) {
        this.left = left;
        this.right = right;
    }

    @Override
    public int interprete() {
        return left.interprete() - right.interprete();
    }


}
```

数值表达式

```
public class ValueExpression implements Expresion{

    private int value;

    public ValueExpression(int value) {
        this.value = value;
    }

    @Override
    public int interprete() {
        return value;
    }

}
```

测试解释器模式，通过将1+3+4-5翻译成一串表达式的组合，我们省略了这个Parse语句转化成表达组合的过程，也省略了Context环境变量类（存储变量名和值的键值对），因为我们直接用数值进行算没有用变量。

```
public class TestInterpreter {

    public static void main(String[] args) {

        // 我们省略了将表达式Parse的过程和Context环境变量类
        // 1+3+4-5
        int result = new MinusExpression(new AddExpression(new AddExpression(new ValueExp
ession(1),
                new ValueExpression(3)), new ValueExpression(4)),new ValueExpression(5)).interpret
();

        System.out.println("计算结果为：" + result);

    }

}
```

运行结果：

```
计算结果为：3
```

解析：

1. 扩展性好。由于在解释器模式中使用类来表示语言的文法规则，因此可以通过继承等机制来改变或展文法。

2. 容易实现。在语法树中的每个表达式节点类都是相似的，所以实现其文法较为容易。

3. 可利用场景比较少。对于复杂的文法比较难维护，且会引起类膨胀。