Dokumentation einer sicherheitsgerichteten 2004-Architektur

Softwarequalität

an der Dualen Hochschule Baden-Württemberg Stuttgart

07.04.2020

Bearbeitungszeitraum	28.03.2020 - 15.05.2020
Matrikelnummer, Kurs	8540946; 6430174;, STG-TINF17-ITA
Dozent	Jamal Krini

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

1 Gruppenarbeit

Die Aufgabe bestand in der Implementierung einer 2003-Architektur. Da unsere Gruppe aus vier Mitgliedern besteht, wurde eine 2004-Architektur implementiert. Die Implementierte Anwendung Für die Calculators haben wurde ein Interface entwickelt, welches jeder Calculator implementiert. Dieses Interface bietet eine Methode, um die Ergebnisse der Berechnungen der Calculators abzufragen. Die Main-Klasse bildet den Einstiegspunkt des Programms und bündelt die Funktionalität zum Aufrufen der Calculators und der Voting-Funktionalität.

Für die Operatoren, die die Calculators unterstützen, wurde ein Enum-Datentyp erstellt

1.1 Aufrufen der Calculators

Zum Aufrufen der Calculators und Aufrufen des Votings wurde die Methode calculate entwickelt.

Diese hat eine Liste an CalculatorInput-Objekten als Eingabeparameter und erzeugt auf Grundlagen dessen vier Calculator-Objekte basierend Calculator1, Calculator2, Calculator3 und Calculator4. Diese vier Calculator-Objekte werden dann in eine List gefügt, um anschließend durch diese zu iterieren und die Methode getResult des Interfaces aufrufen. Die zurückgegebenen Ergebnisse der getResult-Methode werden in eine neue Liste gefügt. Mit den Ergebnissen wird die Vote-Methode aufgerufen. Vor dem Aufruf der Voting-Funktionalität werden die Eingabedaten ausgegeben, damit das Voting-Ergebnis nachvollzogen werden kann. Für jeden Fall (0 Fehler, 1 Fehler, ...) wurde eine Methode erstellt, welche die calculate-Methode mit den korrespondierenden Eingabedaten aufruft.

1.2 Voting

1.3 Implementierung der einzelnen Calculators

Die Implementierung der Calculators ist in den folgenden Abschnitten beschrieben. Bei der Implementierung wurde darauf geachtet, dass nicht alle Calculators gleich implementiert

sind, da dies dem Sinn der 2004-Architektur nicht entsprechen würde. Alle Calculators besitzen ihre eigenen Unit-Tests.

2 Calculator 1 (Bearbeiter: 8540946)

```java
public class Calculator1 implements ICalculator {
    private double _value1;
    private double _value2;
    private Operator _operator;

    public Calculator1(double value1, double value2, Operator Operator){
        this._value1 = value1;
        this._value2 = value2;
        this._operator = Operator;
    }

    @Override
    public double getResult() {
        double result = 0;
        switch (this._operator){
            case PLUS:
                result = add();
                break;
            case MINUS:
                result = subtract();
                break;
            case DIV:
                result = divide();
                break;
            case MULT:
                result = multiply();
                break;
        }
        return result;
    }

    private double add(){
        return this._value1 + this._value2;
    }

    private double subtract(){
        return this._value1 - this._value2;
    }

    private double divide(){
        return  this._value1 / this._value2;
    }

    private double multiply(){
        return this._value1 * this._value2;
    }
}
```

Listing 2.1: Calculator 1

Die Klasse *Calculator1* implementiert das Interface *ICalculator* und besitzt drei Attribute vom Typ *double*: *_value1*, *_value2* und *_operand*.
Dem Konstruktor der Klasse werden die Eingabeparameter value1, value2 und operand übergeben und dieser setzt die Klassenattribute gleich der Eingabeparameter.
Neben den drei Attributen besitzt die Klasse die vier private Methoden *add()*, *subtract()*, *multiply()* und *divide()*. Diese sind Hilfsfunktionen, um aus den *values* das Ergebnis zu berechnen. Welche dieser Methoden aufgerufen wird, hängt von dem Operator ab.
Dieser wird geprüft, wenn der Nutzer die Methode *getResult()* aufruft. Die Methode *getResult()* implementiert die entsprechende Methode des Interfaces. Über die switch-Anweisung wird die zu dem jeweiligen Operator passende Methode zur Berechnung aufgerufen und am Ende wird das Resultat zurückgegeben. Für die jeweilige Berechnung werden die vier Methoden *add()*, *subtract()*, *multiply()* und *divide()* aufgerufen, die auf die Klassenattribute zugreifen und entsprechend das Ergebnis berechnen und zurückgeben.
Das zurückgegebene Ergebnis wird in der lokalen Variable *result* der Methode *getResult()* gespeichert und die Methode gibt den Wert dieser Variable zurück.

3 Calculator 2 (Bearbeiter: 6430174)

```java
public class Calculator2 implements ICalculator{

    private double result;

    public Calculator2 (double first_value, double sec_value, Operator Operator)
    {
        if (Operator == Operator.PLUS){
            result = first_value + sec_value;
        }
        else if (Operator == Operator.MINUS){
            result = first_value - sec_value;
        }
        else if (Operator == Operator.MULT){
            result = first_value * sec_value;
        }
        else if (Operator == Operator.DIV) {
            result = first_value / sec_value;
        }
    }

    public double getResult() {
        return result;
    }
}
```

Listing 3.1: Calculator 2

Der *Calculator2* implementiert das Interface *ICalculator*. Die Klasse *Calculator2* hat ein privates Attribut result vom Typ double. Der Konstruktor der Klasse übergibt die Eingabeparameter an das Calculator-Objekt. Die Eingabeparameter sind: Wert 1, Wert 2 und der Operator. Im Konstruktor wird direkt mittels der Eingabeparameter das Ergebnis berechnet. Dafür wird mit If-Abfragen der Operand ausgemacht und entsprechend das Ergebnis berechnet. Das Ergebnis wird dem Klassenattribut result zugewiesen. Die Klasse verfügt außerdem über die öffentliche Methode *getResult()*, die die gleichnamige Methode des Interfaces implementiert. Die Methode gibt den Wert des Attributs result zurück.

4 Calculator 3 (Bearbeiter:)

5 Calculator 4 (Bearbeiter:)

6 Anhang

```
Input Calculator1: Input 1: 1.0, Input 2: 1.0, Operator: PLUS
Input Calculator2: Input 1: 1.0, Input 2: 1.0, Operator: PLUS
Input Calculator3: Input 1: 1.0, Input 2: 1.0, Operator: PLUS
Input Calculator4: Input 1: 1.0, Input 2: 1.0, Operator: PLUS
Voting-Ergebnis: Kein Fehler. Korrekte Loesung: 2.0

Input Calculator1: Input 1: 1.0, Input 2: 1.0, Operator: PLUS
Input Calculator2: Input 1: 1.0, Input 2: 1.0, Operator: PLUS
Input Calculator3: Input 1: 1.0, Input 2: 1.0, Operator: PLUS
Input Calculator4: Input 1: 1.0, Input 2: 1.0, Operator: MULT
Voting-Ergebnis: Ein Fehler. Korrekte Loesung: 2.0

Input Calculator1: Input 1: 1.0, Input 2: 1.0, Operator: PLUS
Input Calculator2: Input 1: 1.0, Input 2: 1.0, Operator: PLUS
Input Calculator3: Input 1: 1.0, Input 2: 1.0, Operator: MULT
Input Calculator4: Input 1: 1.0, Input 2: 3.0, Operator: MULT
Voting-Ergebnis: Zwei Fehler. Korrekte Loesung: 2.0

Input Calculator1: Input 1: 1.0, Input 2: 1.0, Operator: PLUS
Input Calculator2: Input 1: 1.0, Input 2: 1.0, Operator: PLUS
Input Calculator3: Input 1: 1.0, Input 2: 1.0, Operator: MULT
Input Calculator4: Input 1: 1.0, Input 2: 1.0, Operator: MULT
Voting-Ergebnis: Zwei Fehler aufgetreten. Keine eindeutige Lösung vorhanden. Mögliche Lösungen: 2.0, 1.0

Input Calculator1: Input 1: 1.0, Input 2: 1.0, Operator: PLUS
Input Calculator2: Input 1: 1.0, Input 2: 1.0, Operator: DIV
Input Calculator3: Input 1: 4.0, Input 2: 2.0, Operator: MULT
Input Calculator4: Input 1: 3.0, Input 2: 3.0, Operator: MINUS
Voting-Ergebnis: Keine übereinstimmenden Ergebnisse.


Process finished with exit code 0
```

Abbildung 6.1: Ausgabe des Tools