JUnit实验报告

班级: 软件工程02

学号:

姓名:

指导教师:

实验环境 实验步骤

一、软件介绍

Test Runner for Java

Language Support for Java(TM) by Red Hat

- 二、软件安装
- 三、实验过程
 - 1.新建项目
 - 2.引入JUnit相关包
 - 3.编写测试代码
 - @Test
 - @Before
 - @lgnore
 - 5.开始测试
 - 6.测试结果

实验环境

• 软件环境:

o 操作系统: Windows 11 23H2

IDE: VSCodeJDK: jdk-22

○ 其他软件: Test Runner for Java (提供JUnit) 、Language Support for Java(TM) by Red

Hat (Java项目构建)

实验步骤

一、软件介绍

Test Runner for Java

一个轻量级扩展,用于在 Visual Studio Code 中运行和调试 Java 测试用例。该扩展支持以下测试框架:

- JUnit 4 (v4.8.0+)
- JUnit 5 (v5.1.0+)
- TestNG (v6.9.13.3+)

Java 测试运行器与 Red Hat 的 Java 语言支持和 Java 调试器协同工作,提供以下功能:

- 运行/调试测试用例
- 自定义测试配置
- 查看测试报告
- 在测试资源管理器中查看测试

Language Support for Java(TM) by Red Hat

Language Support for Java™ by Red Hat 是为 Visual Studio Code 开发的 Java 语言支持扩展,提供了语法高亮、代码补全、调试支持等功能,帮助 Java 开发人员更轻松地在 VS Code 中编写、调试和管理 Java 代码。

- 调试支持:集成了调试器,可以在 Visual Studio Code 中调试 Java 程序,包括设置断点、观察变量值等功能。
- 项目管理和构建支持: 可以在 Visual Studio Code 中管理 Maven 和 Gradle 项目,执行构建和依赖管理等操作。
- 集成测试框架:支持与常见的 Java 测试框架集成,如 JUnit 和 TestNG,可以轻松地运行和管理测试用例。

二、软件安装

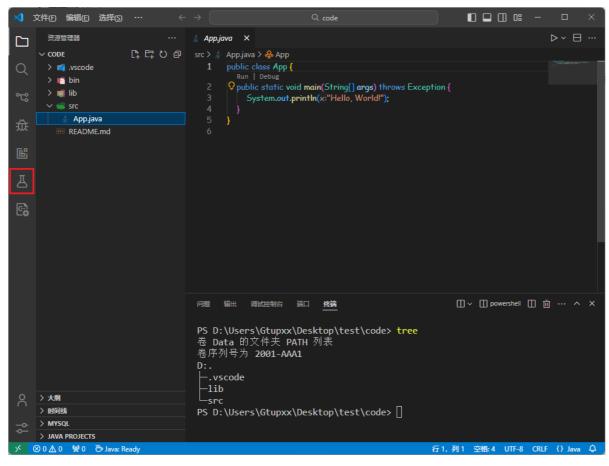
1. JDK下载地址: Java Downloads | Oracle 中国

2. 其余插件可从VSCode直接下载。

三、实验过程

1.新建项目

通过Language Support for Java(TM) by Red Hat,我们可以快速构建出一个基本的java项目模板:



如果Test Runner for Java被正确安装了,可以看到左侧出现了测试的徽章(上图红框)。

我们先简单编写一个Calculator类用于充当接下来被测对象:

```
1
2
     * @Author: 柚岩龙蜥
 3
    * @Date: 2024-05-26 22:37:56
4
     * @LastEditors: 柚岩龙蜥
 5
    * @LastEditTime: 2024-05-26 22:41:32
 6
     * @FilePath: \code\src\Calculator.java
7
     * @Description:
8
9
     */
10
    public class Calculator
11
12
    {
        private static int result; // 静态变量,用于存储运行结果
13
14
15
        public void add(int n)
16
        {
17
            result = result + n;
18
        }
        public void substract(int n)
19
20
        {
21
            result = result - 1; //Bug: 正确的应该是 result = result - n
```

```
22
23
        public void multiply(int n)
24
        {
25
            // 此方法尚未写好
26
        }
27
        public void divide(int n)
28
29
            result = result / n;
30
        }
31
        public void square(int n)
32
        {
33
           result = n * n;
34
        }
        public void squareRoot(int n)
35
36
        {
37
            for (; ;); //Bug: 死循环
38
        }
        public void clear() // 将结果清零
39
40
        {
            result = 0;
41
42
        }
43
        public int getResult()
44
        {
45
           return result;
46
        }
47
    }
48
```

2.引入JUnit相关包

我们点击进入Test界面,选择"Enable Java Tests",并在随后的弹窗中选择JUnit,VSCode会帮我们自动下载并导入所用到的包。

在lib文件夹下可以看到新增的包为: hamcrest-core-1.3.jar和junit-4.13.2.jar。

```
□ □ □ -
刘 文件(F) 编辑(E) 选择(S)
                  ರ ⊳ ♦
                              JUnit Jupiter
                              JUnit
       尚未在此工作区中找到任 TestNG
Q
                                                   ^ ©Author: 他石疋坳
* @Date: 2024-05-26 22:37:56
       Configure a test framework to see
                                                   * @LastEditors: 柚岩龙蜥
|* @LastEditTime: 2024-05-26 22:41:32
             Configure Python Tests
                                                   * @FilePath: \code\src\Calculator.java
       生成 C# 测试项目以在此处查看测
       Click below button to configure a test
       framework for your project.
Д
                Enable Java Tests
                                                      private static int result; // 静态变量,用于存储运行结果
                                              14
15
               安装其他测试扩展
                                                      public void add(int n)
                                                        result = result + n;
                                                      public void substract(int n)
                                                        result = result - 1; //Bug: 正确的应该是 result = result - n
                                            问题 18 输出 调试控制台 端口 终端
                                                                                                           + ∨ ☑ powershell Ⅲ 前 ··· ∧ ×
                                            PS D:\Users\Gtupxx\Desktop\test\code> tree
卷 Data 的文件夹 PATH 列表
卷序列号为 2001-AAA1
                                              -.vscode
                                             —lib
8
                                            PS D:\Users\Gtupxx\Desktop\test\code>
    ⊗ 18 <u>∧</u> 0 № 0 <del>></del> Java: Ready
                                                                                                     行48,列1 空格:4 UTF-8 CRLF () Java
```

3.编写测试代码

我们新建一个CalculatorTest类,用于测试Calculator类中的各种方法。

```
1
    import static org.junit.Assert.assertEquals;
2
3
    import org.junit.Before;
4
    import org.junit.Ignore;
5
    import org.junit.Test;
 6
7
    public class CalculatorTest
8
9
        private static Calculator calculator = new Calculator();
10
11
        @Before
12
        public void setUp() throws Exception //初始化操作
13
        {
14
            calculator.clear();
15
        }
16
17
        @Test
        public void testAdd() //测试加法
18
19
        {
20
            calculator.add(2);
21
            calculator.add(3);
22
            assertEquals(5, calculator.getResult());
        }
23
24
25
        @Test
        public void testSubtract() //测试减法
26
27
        {
```

```
28
            calculator.add(10);
29
            calculator.substract(2);
30
            assertEquals(8, calculator.getResult());
        }
31
32
33
        @Ignore("Multiply() Not yet implemented")
34
        @Test
35
        public void testMultiply() //测试乘法
36
37
        }
38
39
        @Test
40
        public void testDivide() //测试除法
41
42
            calculator.add(8);
43
            calculator.divide(2);
            assertEquals(4, calculator.getResult());
44
        }
45
46
47
        @Test(expected = ArithmeticException.class)
48
        public void testDivideZero() //测试除法
49
50
            calculator.add(8);
51
            calculator.divide(0);
52
        }
53
54
        @Test
        public void testSqare() //测试平方
55
56
57
            calculator.square(4);
58
            assertEquals(16, calculator.getResult());
        }
59
60
61
        @Test(timeout = 1000)
        public void testSquareRoot()
62
63
        {
64
            calculator.squareRoot(16);
65
            assertEquals(4, calculator.getResult());
66
        }
67
    }
68
```

@Test

@Test 是 JUnit 框架中的一个注解,用于标记一个方法是测试方法。在 JUnit 测试类中,所有被 @Test 注解标记的方法都会被 JUnit 框架执行,用于测试被测试类中的某个功能或方法是否符合预期。

- 1. **标记测试方法**: 使用 @Test 注解标记的方法将被 JUnit 框架识别为一个测试方法,用于执行相应的测试任务。
- 2. **测试方法命名规范**: 通常,测试方法的命名应该具有描述性,清晰地表明被测试的功能或方法以及期望的行为。例如, testAdd() 、 testSubtract() 等。
- 3. **执行顺序:** JUnit 框架会按照测试方法在源代码中的顺序依次执行测试。但是,并不保证测试方法的执行顺序,因此应该避免编写依赖于测试方法执行顺序的测试用例。

- 4. **断言验证**: 在测试方法中,通常会使用断言方法(如 assertEquals()、 assertTrue()等)来验证被测试方法的执行结果是否符合预期。
- 5. **异常测试:** 除了正常情况下的测试, @Test 注解还支持对异常情况的测试。可以通过 expected 属性来指定测试方法是否会抛出特定的异常,例如 @Test(expected = ArithmeticException.class)。

@Before

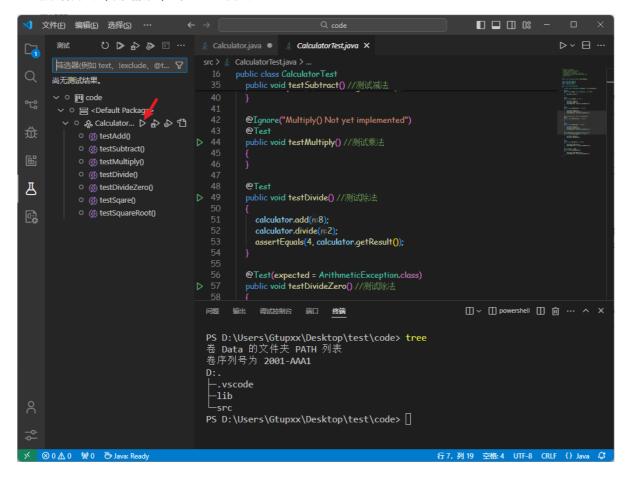
@Before 注解标记的方法会在每个测试方法执行前执行,用于进行测试的准备工作。通常,@Before 方法用于初始化测试环境、准备测试数据或执行其他必要的设置操作。例如,在测试之前清除数据库、创建临时文件、初始化测试对象等。@Before 注解使得这些操作在每个测试方法执行前都能被正确执行,确保测试的独立性和可靠性。

@Ignore

@Ignore 注解用于标记暂时不需要执行的测试方法。当一个测试方法被 @Ignore 注解标记时, JUnit 框架将会跳过执行该测试方法,直接执行其他未被标记的测试方法。通常情况下, @Ignore 注解可以用于标记一些尚未实现或者临时不可用的测试方法,避免这些方法影响整体的测试结果。在实现完毕或者问题解决后,再移除 @Ignore 注解,使得该测试方法能够被正常执行。

5.开始测试

进入Test界面,现在我们编写的CalculatorTest类中使用@Test 注解的方法将会出现在测试界面中,我们点击开始测试(下图箭头)即可完成测试。



6.测试结果

如上图,我们可以看到减法的测试没有通过,乘法的测试被跳过,除数为零正常通过,平方根的测试超时了,其余测试均通过。

上面是我们设计好的结果。