Java 核心技术(进阶)

第二章 单元测试和JUnit

第一节 单元测试

华东师范大学 陈良育

软件测试

- 软件测试的经典定义是：在规定的条件下对程序进行操作，以发现程序错误，衡量软件质量，并对其是否能满足设计要求进行评估的过程。
- 软件测试分类
  - 单元 vs 集成测试
  - 白盒 vs 黑盒测试
  - 自动 vs 手动测试
  - 回归测试
  - 压力测试
  - 。。。。。。

单元和集成测试

- 单元测试（unit testing），是指对软件中的最小可测试单元进行检查和验证。通常是一个函数/方法。
- 单元测试是已知代码结构进行的测试，属于白盒测试。

- 集成测试是将多个单元相互作用，形成一个整体，对整体协调性进行测试。
- 一般从构成系统的最小单元开始，持续推进到单元之间的接口直到集成为一个完成的软件系统为止。

白盒和黑盒测试

- 白盒测试 (white-box testing)，全面了解程序内部逻辑结构，对所有的逻辑路径都进行测试。一般由程序员完成。

- 黑盒测试 (black-box testing)，又名功能测试，将程序视为一个不能打开的黑盒子。在完全不考虑程序内部结构和内部特性的情况下，检查程序功能是否按照需求规格说明书的规定正常使用。一般由独立的使用者完成。

自动和手动测试

- 自动测试：用测试程序批量、反复测试功能程序，并可自动检查功能程序输出结果是否满足预定的要求。

- 手动测试：手动执行程序，手动输入所需要的参数，手动检查程序结果是否满足预定的要求。

回归测试

- 回归测试：修改旧代码后，重新进行测试以确认修改没有引入新的错误或导致其他代码产生错误。

- 回归测试在整个软件测试过程中占有很大的比重。软件快速迭代开发过程中，新版本的连续发布(Daily/Nightly Build)使得回归测试进行的更加频繁。

测试策略

- 基于main函数的策略
  - 优点：简单
  - 缺点：
    - 无法自动判断被测对象的行为是否符合预期
    - main方法需要添加大量的代码，这些代码在发布时候也需要手动删除
    - 分散程序员在开发时的关注点
- 基于自动化测试框架的策略
  - 初始化->输入测试数据执行被测代码->获取系统实际结果->比较结果是否一致->输出测试结论

main策略测试的例子

```java
public class Calculator {

    public int add(int a,int b) {
        return a + b;
    }

    public int subtract(int a, int b) {
        return a - b;
    }

    public int multiply(int a,int b) {
        return a * b;
    }

    public int divide(int a ,int b) {
        return a / b;
    }
}
```

```java
public class CalculatorTest {

    public static void main(String[] args) {
        Calculator c = new Calculator();
        System.out.println(3 == c.add(1, 2));
        System.out.println(-1 == c.subtract(1, 2));
        System.out.println(3 == c.multiply(1, 2));
        System.out.println(0 == c.divide(1, 2));
    }
}
```

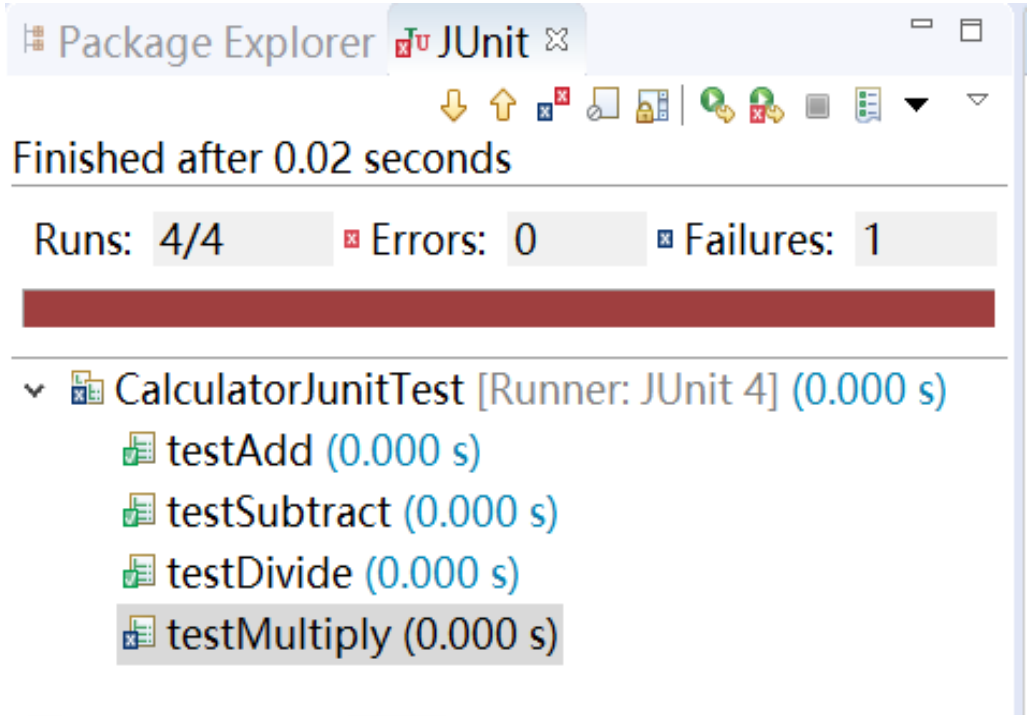
Problems @ Javadoc Declaration Console

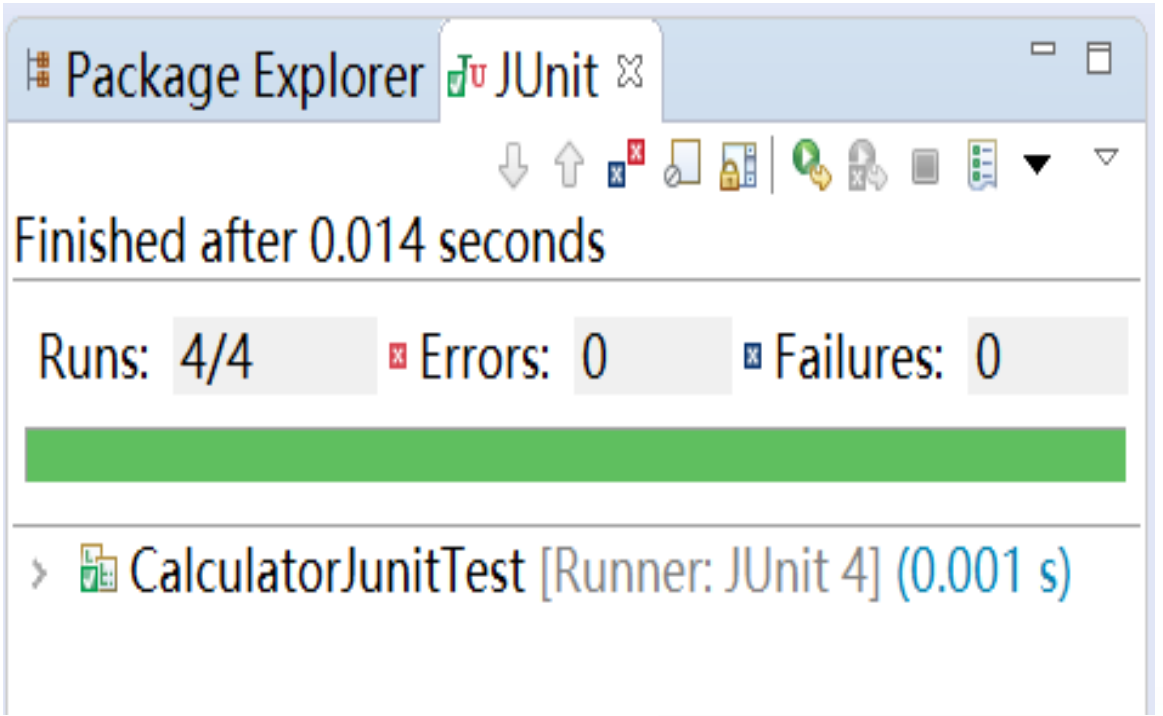
<terminated> CalculatorTest [Java Application] E:\java\jc

```
true
true
false
true
```

JUnit

- JUnit是一个Java语言的单元测试框架





推荐书籍

- 软件测试--基于问题驱动模式，朱少民，高等教育出版社，2016。
- 软件测试(第二版)，Rom Patton著，张小松等译，机械工业出版社，2006。
- 从菜鸟到测试架构师——一个测试工程师的成长日记，孙磊等著，电子工业出版社，2013。

单元测试

- 总结
  - 理解测试在软件开发中的意义
  - 厘清软件测试概念和分类
  - 了解手动测试和自动测试

代码(1) Calculator/CalculatorTest


```java
public class Calculator {

    public int add(int a,int b) {
        return a + b;
    }

    public int subtract(int a, int b) {
        return a - b;
    }

    public int multiply(int a,int b) {
        return a * b;
    }

    public int divide(int a ,int b) {
        return a / b;
    }
}
```

```java
public class CalculatorTest {

    public static void main(String[] args) {
        Calculator c = new Calculator();
        System.out.println(3 == c.add(1, 2));
        System.out.println(-1 == c.subtract(1, 2));
        System.out.println(3 == c.multiply(1, 2));
        System.out.println(0 == c.divide(1, 2));
    }
}
```

代码(2) CalculatorJunitTest


```
import static org.junit.Assert.*;

import org.junit.Test;

public class CalculatorJunitTest {
    @Test
    public void testAdd() {
        assertEquals(3, new Calculator().add(1,2));
    }

    @Test
    public void testSubtract() {
        assertEquals(-1, new Calculator().subtract(1,2));
    }

    @Test
    public void testMultiply() {
        assertEquals(2, new Calculator().multiply(1,2));
    }

    @Test
    public void testDivide() {
        assertEquals(0, new Calculator().divide(1,2));
    }
}
```

谢 谢！