实验三 单元测试实验

1. 引言

1.1 标识

本文档适用于以下测试环境

系统：win11，version 22H2

1.2 实验要求

1　通过实验，理解单元测试原理，熟悉单元测试工具的使用。

2　编写四则运算程序或其它，确定测试单元，设计测试用例，借助某单元测试工具做单元测试。

3　具体用什么单元测试工具，根据自己情况自选，如 xUnit、TestNG、gtest、pytest、unittest 等。

4　在实验报告中，给出测试需求、测试设计、测试用例集、测试执行结果及分析。

2. 测试需求

对于如下四则运算程序进行 Junit 单元测试，其测试需求如下：

1. 测试加法方法 add(int a, int b)的计算准确性，包括正整数、负整数的相加。
2. 测试减法方法 subtract(int a, int b)的计算准确性，包括正整数、负整数的相减。
3. 测试乘法方法 multiply(int a, int b)的计算准确性，包括正整数、负整数的相乘。
4. 测试除法方法 divide(int a, int b)的计算准确性，包括正整数、负整数的相除，以及除数为零的情况。
5. 测试结果是否越界的情况，即计算结果是否在 int 类型的范围内。
6. 测试异常情况，如除数为零时是否抛出异常，以及计算结果越界时是否抛出异常。
7. 保证语句覆盖率达到 100%。

```java
public class Calculator {
    // 加法
    public static int add(int a, int b) throws Exception {
        long c = (long) a + b;
        if(check(c))
            return (int)c;
        else
            throw new Exception("结果越界！");
    }

    // 减法
    public static int subtract(int a, int b) throws Exception {
        long c = (long) a - b;
        if(check(c))
            return (int)c;
        else
            throw new Exception("结果越界！");
    }

    // 乘法
    public static int multiply(int a, int b) throws Exception {
        long c = (long) a * b;
        if(check(c))
            return (int)c;
        else
            throw new Exception("结果越界！");
    }

    // 除法
    public static int divide(int a, int b) throws Exception {
        if(b == 0)
            throw new Exception("除数为0！");
        long c = (long) a / b;
        if(check(c))
            return (int)c;
        else
            throw new Exception("结果越界！");
    }

    public static boolean check(long x) {
        return x >= Integer.MIN_VALUE && x <= Integer.MAX_VALUE;
    }
}
```

3. 测试设计

要对每个方法进行单元测试，包括 add，subtract，multiply 和 divide 方法。对于每个方法，我们需要测试以下几个方面：

1. 正常情况下的输入值测试：输入合法的整数，验证计算结果是否正确。
2. 边界情况下的输入值测试：输入边界整数值，例如 Integer.MAX_VALUE，Integer.MIN_VALUE，0 等，验证计算结果是否正确。
3. 异常输入值测试：对于 divide 方法，测试除数为 0 的情况，对于其他方法，测试输入值会导致越界的情况。
4. 异常情况下的输出值测试：对于会导致越界的输入出值，验证是否能捕获并正确抛出异常。

4. 测试用例集

待测试方法	用例序号	输入	预期输出
add	1	a = 2, b = 3	5
	2	a = 2147483646, b = 1	2147483647
	3	a = -2147483647, b = -1	-2147483648
	4	a = 2147483647, b = 1	“结果越界！”
	5	a = -2147483648, b = -1	“结果越界！”
subtract	6	a = 8, b = 3	5
	7	a = 2147483647, b = 1	2147483646
	8	a = -2147483647, b = 1	-2147483648
	9	a = -2147483648, b = 1	“结果越界！”
	10	a = 2147483647, b = -1	“结果越界！”
multiply	11	a = 3, b = 5	15
	12	a = -12345678, b = -173	2135802294
	13	a = 12345678, b = -173	-2135802294
	14	a = 214748365, b = 10	“结果越界！”

	15	a = 21474837, b = 100	“结果越界！”
divide	16	a = 15, b = 3	5
	17	a = -2147483648, b = 1	-2147483648
	18	a = 10, b = 0	“除数为 0！”
	19	a = 2147483647, b = 0	“除数为 0！”
	20	a = -2147483648, b = -1	“结果越界！”

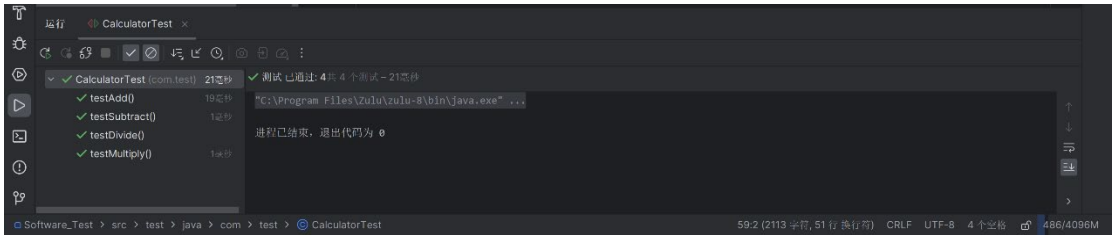
5. 测试执行结果与分析

测试代码如下：

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class CalculatorTest {

    @Test
    void testAdd() {
        try {
            assertEquals(5, Calculator.add(2, 3));
            assertEquals(2147483647, Calculator.add(2147483646, 1));
            assertEquals(-2147483648, Calculator.add(-2147483647, -1));
            assertThrows(Exception.class, () -> Calculator.add(2147483647, 1));
            assertThrows(Exception.class, () -> Calculator.add(-2147483648, -1));
        } catch (Exception e) {
            fail("Unexpected exception: " + e.getMessage());
        }
    }

    @Test
    void testSubtract() {
        try {
            assertEquals(5, Calculator.subtract(8, 3));
            assertEquals(2147483646, Calculator.subtract(2147483647, 1));
            assertEquals(-2147483648, Calculator.subtract(-2147483647, 1));
            assertThrows(Exception.class, () -> Calculator.subtract(-2147483648, 1));
            assertThrows(Exception.class, () -> Calculator.subtract(2147483647, -1));
        } catch (Exception e) {
            fail("Unexpected exception: " + e.getMessage());
```

```
    }
}

@Test
void testMultiply() {
    try {
        assertEquals(15, Calculator.multiply(3, 5));
        assertEquals(2135802294, Calculator.multiply(-12345678, -173));
        assertEquals(-2135802294, Calculator.multiply(12345678, -173));
        assertThrows(Exception.class, () -> Calculator.multiply(214748365, 10));
        assertThrows(Exception.class, () -> Calculator.multiply(21474837, 100));
    } catch (Exception e) {
        fail("Unexpected exception: " + e.getMessage());
    }
}
```

执行结果如下：



所有测试用例结果的实际输出都与预期输出相同。说明代码的正确性和质量良好，程序健壮性强，能有效处理异常输出。

6. 测试记录

测试项目	测试日期	测试时间	测试人员	测试环境
功能测试	2023.12.3	16:00	杨帅棋	华硕无畏 pro15 2022 win11，version 22H2