**《软件测试》**

**软件测试实验二：边界值分析方法**

**姓 名： 胡国昌 学 号： 202210120518**

**院 系： 计算机与信息学院 专 业：计算机与科学**

**实 验 室： B1-119 实验日期： 2025.3.1**

**总评成绩： 审阅教师： 杨青**

# 一、实验目的：

1. 掌握 git 协作式管理工具；
2. 掌握 gradle 或 Maven 项目构建工具基本操作
3. 淬炼基本代码编程能力
4. 掌握单元测试框架Junit基本操作

# 二、课程目标：

# 目标2： 能够运用功能测试、基于控制流和数据流的测试等软件测试的核心技术和原理，结合相关文献，对测试问题及其影响因素进行分析或计算，得出有效结论。

# 三、实验要求：

1. 学习Junit5 参数化测试方法，参考[Junit5 五分钟教程](https://www.jianshu.com/p/3183fb69fd74" \o "Junit5 五分钟教程)
2. 针对实验一实现判断三角形类型程序Triangle，采用边界值分析方法设计测试用例，并采用Junit5 编写完成三角形类型程序测试用例，并提交到代码仓库
3. 按照[实验报告模板](https://star.jmhui.com.cn/u/cms/www/202203/06160105hi3m.docx) 编写实验报告，以“学号-姓名-软件测试实验二”命名，提交到雨课堂“软件测试实验二”

# 四、实验步骤与内容

## 题目：三角形题目

**输入**3个整数a、b和c分别作为三角形的三条边，要求a、b和c必须满足以下条件：

1、整数

2、3个数

3、边长大于等于1，小于等于100

4、任意两边之和大于第三边

**输出**为5种情况之一：

如果不满足条件1、2、3，则程序输出为“输入错误”。

如果不满足条件4，则程序输出为“非三角形”。

如果三条边相等，则程序输出为“等边三角形”。

如果恰好有两条边相等，则程序输出为“等腰三角形”。

如果三条边都不相等，则程序输出为“一般三角形”。

## 代码实现：

### 2.1Java部分

#### 生成测试代码部分

##### 2.1.1.1生成一般边界值测试数据

package com.ctgu.api.conditionaljudgment;

import java.util.ArrayList;

import java.util.List;

import static com.ctgu.api.conditionaljudgment.ParameterClassification.isTriangle;

import static com.ctgu.api.conditionaljudgment.ParameterClassification.isValid;

import static com.ctgu.api.growthparameter.WriteToFile.writeToFile;

public class GeneralBoundary {

private static final int NORMAL = 50; // 正常值

private static final int[] GENERAL\_BOUNDARIES = {

1,

2,

50,

99,

100

}; // 一般边界值

/\*\*

\* 生成一般边界值测试数据（13条用例）

\*/

public static void generateGeneralBoundaryData() {

List < String > testCases = new ArrayList < > ();

// 分别测试a、b、c的边界值，其他参数固定为NORMAL

for (int boundary: GENERAL\_BOUNDARIES) {

testCases.add(String.format("%d,%d,%d,%b,%b", boundary, NORMAL, NORMAL, isValid(boundary, NORMAL, NORMAL), isTriangle(boundary, NORMAL, NORMAL)));

testCases.add(String.format("%d,%d,%d,%b,%b", NORMAL, boundary, NORMAL, isValid(NORMAL, boundary, NORMAL), isTriangle(boundary, NORMAL, NORMAL)));

testCases.add(String.format("%d,%d,%d,%b,%b", NORMAL, NORMAL, boundary, isValid(NORMAL, NORMAL, boundary), isTriangle(boundary, NORMAL, NORMAL)));

}

// 添加一个全正常值的用例

testCases.add(String.format("%d,%d,%d,%b,%b", NORMAL, NORMAL, NORMAL, true, true));

writeToFile("GeneralBoundary.csv", testCases);

}

}

##### 2.1.1.2生成健壮性边界值测试数据

package com.ctgu.api.conditionaljudgment;

import java.util.ArrayList;

import java.util.List;

import static com.ctgu.api.conditionaljudgment.ParameterClassification.isTriangle;

import static com.ctgu.api.conditionaljudgment.ParameterClassification.isValid;

import static com.ctgu.api.growthparameter.WriteToFile.writeToFile;

public class RobustBoundary {

private static final int[] ROBUST\_BOUNDARIES = {

0,

1,

2,

50,

99,

100,

101

}; // 健壮性边界值

/\*\*

\* 生成健壮性边界值测试数据（19条用例）

\*/

public static void generateRobustBoundaryData() {

List < String > testCases = new ArrayList < > ();

// 分别测试a、b、c的越界值

for (int boundary: ROBUST\_BOUNDARIES) {

testCases.add(String.format("%d,%d,%d,%b,%b", boundary, 50, 50, isValid(boundary, 50, 50), isTriangle(boundary, 50, 50)));

testCases.add(String.format("%d,%d,%d,%b,%b", 50, boundary, 50, isValid(50, boundary, 50), isTriangle(boundary, 50, 50)));

testCases.add(String.format("%d,%d,%d,%b,%b", 50, 50, boundary, isValid(50, 50, boundary), isTriangle(boundary, 50, 50)));

}

writeToFile("RobustBoundary.csv", testCases);

}

}

##### 2.1.1.3生成最坏情况一般边界值测试数据

package com.ctgu.api.conditionaljudgment;

import java.util.ArrayList;

import java.util.List;

import static com.ctgu.api.conditionaljudgment.ParameterClassification.isTriangle;

import static com.ctgu.api.conditionaljudgment.ParameterClassification.isValid;

import static com.ctgu.api.growthparameter.WriteToFile.writeToFile;

public class WorstCaseGeneral {

/\*\*

\* 生成最坏情况一般边界值测试数据（125条用例）

\*/

public static void generateWorstCaseGeneralData() {

List < String > testCases = new ArrayList < > ();

int[] boundaries = {

1,

2,

50,

99,

100

};

for (int a: boundaries) {

for (int b: boundaries) {

for (int c: boundaries) {

boolean valid = isValid(a, b, c);

boolean triangle = isTriangle(a, b, c);

testCases.add(String.format("%d,%d,%d,%b,%b", a, b, c, valid, triangle));

}

}

}

writeToFile("WorstCaseGeneral.csv", testCases);

}

}

##### 2.1.1.4生成最坏情况健壮性边界值测试数据

package com.ctgu.api.conditionaljudgment;

import java.util.ArrayList;

import java.util.List;

import static com.ctgu.api.conditionaljudgment.ParameterClassification.isTriangle;

import static com.ctgu.api.conditionaljudgment.ParameterClassification.isValid;

import static com.ctgu.api.growthparameter.WriteToFile.writeToFile;

public class WorstCaseRobust {

/\*\*

\* 生成最坏情况健壮性边界值测试数据（343条用例）

\*/

public static void generateWorstCaseRobustData() {

List < String > testCases = new ArrayList < > ();

int[] boundaries = {

0,

1,

2,

50,

99,

100,

101

};

for (int a: boundaries) {

for (int b: boundaries) {

for (int c: boundaries) {

boolean valid = isValid(a, b, c);

boolean triangle = isTriangle(a, b, c);

testCases.add(String.format("%d,%d,%d,%b,%b", a, b, c, valid, triangle));

}

}

}

writeToFile("WorstCaseRobust.csv", testCases);

}

}

##### 2.1.1.5启动函数和写入函数

package com.ctgu.api.growthparameter;

import static com.ctgu.api.conditionaljudgment.GeneralBoundary.generateGeneralBoundaryData;

import static com.ctgu.api.conditionaljudgment.RobustBoundary.generateRobustBoundaryData;

import static com.ctgu.api.conditionaljudgment.WorstCaseGeneral.generateWorstCaseGeneralData;

import static com.ctgu.api.conditionaljudgment.WorstCaseRobust.generateWorstCaseRobustData;

import static com.ctgu.api.conditionaljudgment.WorstCaseRobustType.generateWorstCaseRobustTypeData;

public class TestDataGenerator {

public static void main(String[] args) {

generateGeneralBoundaryData();

generateRobustBoundaryData();

generateWorstCaseGeneralData();

generateWorstCaseRobustData();

generateWorstCaseRobustTypeData();

}

}

package com.ctgu.api.growthparameter;

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

import java.util.List;

public class WriteToFile {

// 辅助方法：写入CSV文件

public static void writeToFile(String fileName, List < String > lines) {

try (BufferedWriter writer = new BufferedWriter(new FileWriter("src/test/resources/" + fileName))) {

writer.write("a,b,c,isValid,isTriangle");

writer.newLine();

for (String line: lines) {

writer.write(line);

writer.newLine();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

##### 2.1.1.6 测试用例 测试函数

**TriangleChecker.java**

package com.ctgu.api;

public class TriangleChecker {

// 解析并验证输入参数

public static int[] parseAndValidate(String[] args) throws IllegalArgumentException {

if (args.length != 3) {

throw new IllegalArgumentException("参数不足");

}

try {

int a = Integer.parseInt(args[0]);

int b = Integer.parseInt(args[1]);

int c = Integer.parseInt(args[2]);

if (a < 1 || a > 100 || b < 1 || b > 100 || c < 1 || c > 100) {

throw new IllegalArgumentException("参数超出范围时抛出异常");

}

return new int[] {

a,

b,

c

};

} catch (NumberFormatException e) {

throw new IllegalArgumentException("非整数参数时抛出异常");

}

}

// 判断是否为三角形

public static boolean isTriangle(int a, int b, int c) {

return a + b > c && a + c > b && b + c > a;

}

// 判断三角形类型

public static String determineTriangleType(int a, int b, int c) {

if (a == b && b == c) {

return "等边三角形";

} else if (a == b || a == c || b == c) {

return "等腰三角形";

} else {

return "一般三角形";

}

}

}

**TriangleCheckerTest.java**

package com.ctgu.api;

import com.ctgu.api.testcases.TriangleChecker;

import org.junit.jupiter.api.DisplayName;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.CsvFileSource;

import org.junit.jupiter.params.provider.CsvSource;

import org.junit.jupiter.params.provider.ValueSource;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import static org.junit.jupiter.api.Assertions.\*;

public class TriangleCheckerTest {

private static final Logger logger = LoggerFactory.getLogger(TriangleCheckerTest.class);

// 参数不足时抛出异常

@ParameterizedTest

@DisplayName("参数不足时抛出异常")

// @ValueSource(strings = {"1", "1,2", "1,2,3,4"})

@CsvFileSource(resources = "/InsufficientParameters.csv")

public void testParseAndValidate\_InsufficientParameters(String input) {

String[] args = input.split(",");

IllegalArgumentException exception = assertThrows(

IllegalArgumentException.class,

() -> TriangleChecker.parseAndValidate(args)

);

logger.info("InsufficientParameters: {} Catch exception: {}", args, exception.getMessage());

assertEquals("参数不足", exception.getMessage());

}

// 非整数参数时抛出异常

@ParameterizedTest

@DisplayName("非整数参数时抛出异常")

@CsvSource({

"a, 2, 3",

"1, b, 3",

"1, 2, c"

})

public void testParseAndValidate\_NonIntegerParameters(String a, String b, String c) {

IllegalArgumentException exception = assertThrows(

IllegalArgumentException.class,

() -> TriangleChecker.parseAndValidate(new String[] {

a,

b,

c

})

);

logger.info("NonIntegerParameters :{} Catch exception: {}", new String[] {

a,

b,

c

}, exception.getMessage());

assertEquals("非整数参数时抛出异常", exception.getMessage());

}

// 参数超出范围时抛出异常

@ParameterizedTest

@DisplayName("参数超出范围时抛出异常")

@CsvSource({

"0, 10, 20",

"10, 0, 30",

"10, 20, 0",

"101, 20, 30",

"10, 200, 101",

"10, 20, 101"

})

public void testParseAndValidate\_OutOfRangeParameters(String a, String b, String c) {

IllegalArgumentException exception = assertThrows(

IllegalArgumentException.class,

() -> TriangleChecker.parseAndValidate(new String[] {

a,

b,

c

})

);

logger.info(" OutOfRangeParameters :{} Catch exception: {}", new String[] {

a,

b,

c

}, exception.getMessage());

assertEquals("参数超出范围时抛出异常", exception.getMessage());

}

@Test

@DisplayName("合法参数解析正确")

public void testParseAndValidate\_ValidParameters() {

int[] expected = {

3,

4,

5

};

logger.info("testParseAndValidate\_ValidParameters : {}", TriangleChecker.parseAndValidate(new String[] {

"3",

"4",

"5"

}));

assertArrayEquals(expected, TriangleChecker.parseAndValidate(new String[] {

"3",

"4",

"5"

}));

}

@ParameterizedTest

@DisplayName("非三角形判断")

@CsvSource({

"1, 2, 3",

"1, 1, 2",

"10, 1, 1"

})

public void testIsTriangle\_NonTriangle(int a, int b, int c) {

logger.info("NonTriangle :{} testIsTriangle\_NonTriangle: {}", new int[] {

a,

b,

c

}, TriangleChecker.isTriangle(a, b, c));

assertFalse(TriangleChecker.isTriangle(a, b, c));

}

@ParameterizedTest

@DisplayName("合法三角形判断")

@CsvSource({

"3, 4, 5",

"5, 12, 13",

"7, 24, 25"

})

public void testIsTriangle\_ValidTriangle(int a, int b, int c) {

logger.info("ValidTriangle: {}testIsTriangle\_ValidTriangle: {}", new int[] {

a,

b,

c

}, TriangleChecker.isTriangle(a, b, c));

assertTrue(TriangleChecker.isTriangle(a, b, c));

}

// 等边三角形判断

@ParameterizedTest

@DisplayName("等边三角形判断")

@ValueSource(ints = {

3,

5,

7

})

public void testDetermineTriangleType\_Equilateral(int side) {

logger.info("Equilateral: {}testDetermineTriangleType\_Equilateral: {}", new int[] {

side,

side,

side

}, TriangleChecker.determineTriangleType(side, side, side));

assertEquals("等边三角形", TriangleChecker.determineTriangleType(side, side, side));

}

@ParameterizedTest

@DisplayName("等腰三角形判断")

@CsvSource({

"3, 3, 4",

"5, 5, 8",

"7, 7, 10"

})

public void testDetermineTriangleType\_Isosceles(int a, int b, int c) {

logger.info("Isosceles: {} testDetermineTriangleType\_Isosceles: {}", new int[] {

a,

b,

c

}, TriangleChecker.determineTriangleType(a, b, c));

assertEquals("等腰三角形", TriangleChecker.determineTriangleType(a, b, c));

}

@ParameterizedTest

@DisplayName("一般三角形判断")

@CsvSource({

"3, 4, 5",

"5, 12, 13",

"7, 24, 25"

})

public void testDetermineTriangleType\_Scalene(int a, int b, int c) {

logger.info("Scalene: {} testDetermineTriangleType\_Scalene: {}", new int[] {

a,

b,

c

}, TriangleChecker.determineTriangleType(a, b, c));

assertEquals("一般三角形", TriangleChecker.determineTriangleType(a, b, c));

}

@ParameterizedTest

@DisplayName("最坏情况健壮性边界值分析")

// @CsvFileSource(resources = "/GeneralBoundary.csv",numLinesToSkip = 1)

// @CsvFileSource(resources = "/RobustBoundary.csv",numLinesToSkip = 1)

// @CsvFileSource(resources = "/WorstCaseGeneral.csv",numLinesToSkip = 1)

@CsvFileSource(resources = "/WorstCaseRobust.csv", numLinesToSkip = 1)

public void testGeneralBoundary(int a, int b, int c, boolean isValid) {

if (isValid) {

assertDoesNotThrow(() -> TriangleChecker.parseAndValidate(new String[] {

String.valueOf(a), String.valueOf(b), String.valueOf(c)

}));

} else {

assertThrows(IllegalArgumentException.class, () -> TriangleChecker.parseAndValidate(new String[] {

String.valueOf(a), String.valueOf(b), String.valueOf(c)

}));

}

}

@ParameterizedTest

@DisplayName("最坏情况健壮性边界值分析")

@CsvFileSource(resources = "/GeneralBoundary.csv", numLinesToSkip = 1)

// @CsvFileSource(resources = "/RobustBoundary.csv",numLinesToSkip = 1)

// @CsvFileSource(resources = "/WorstCaseGeneral.csv",numLinesToSkip = 1)

// @CsvFileSource(resources = "/WorstCaseRobust.csv",numLinesToSkip = 1)

public void testIsTriangle(int a, int b, int c, boolean isValid, boolean isTriangle) {

if (isTriangle) {

assertTrue(TriangleChecker.isTriangle(a, b, c));

} else {

assertFalse(TriangleChecker.isTriangle(a, b, c));

}

}

@ParameterizedTest

@DisplayName("最坏情况健壮性边界值分析")

@CsvFileSource(resources = "/WorstCaseRobustType.csv", numLinesToSkip = 1)

public void testDetermineTriangleType(int a, int b, int c, String expected) {

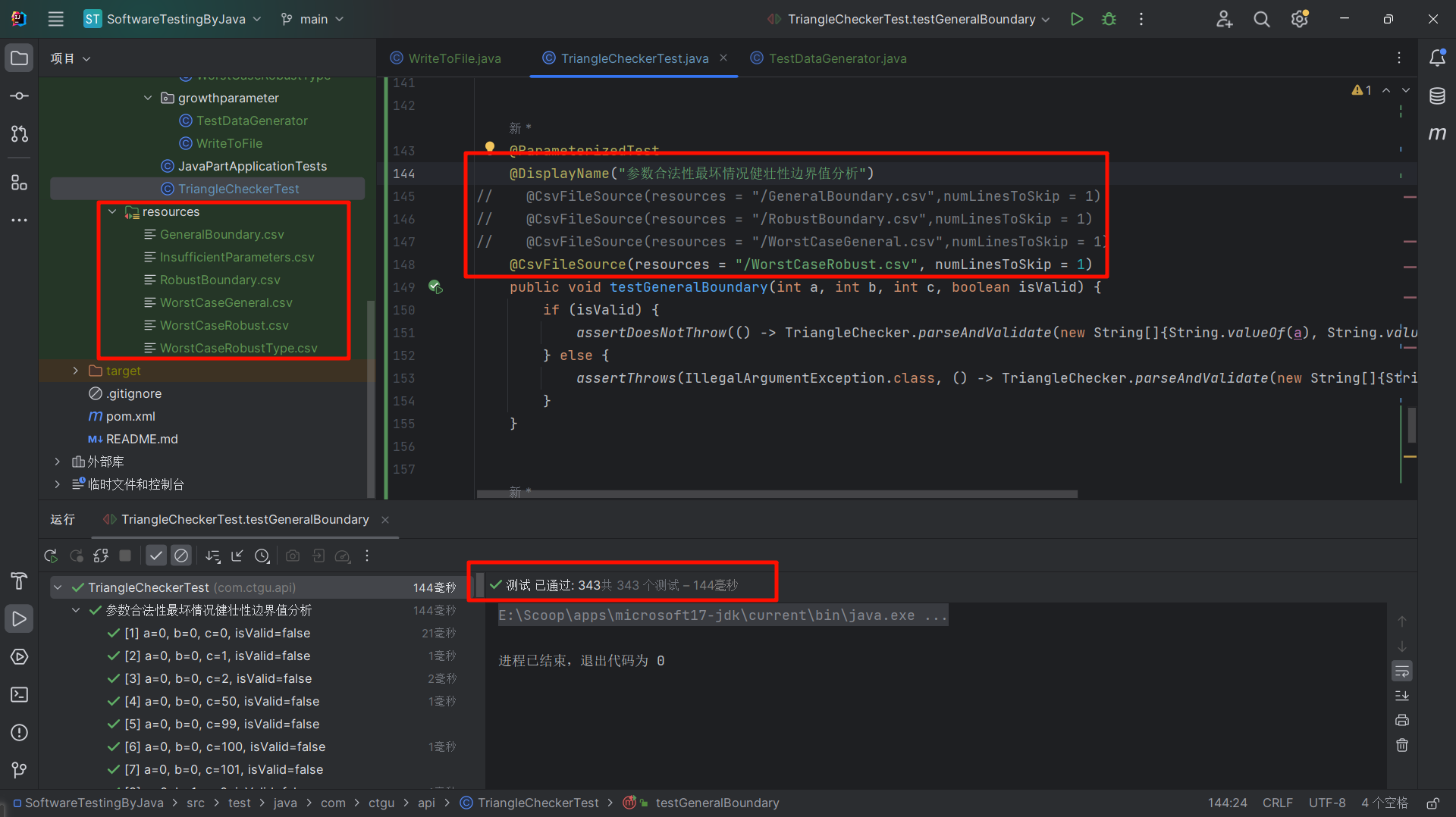
assertEquals(expected, TriangleChecker.determineTriangleType(a, b, c));

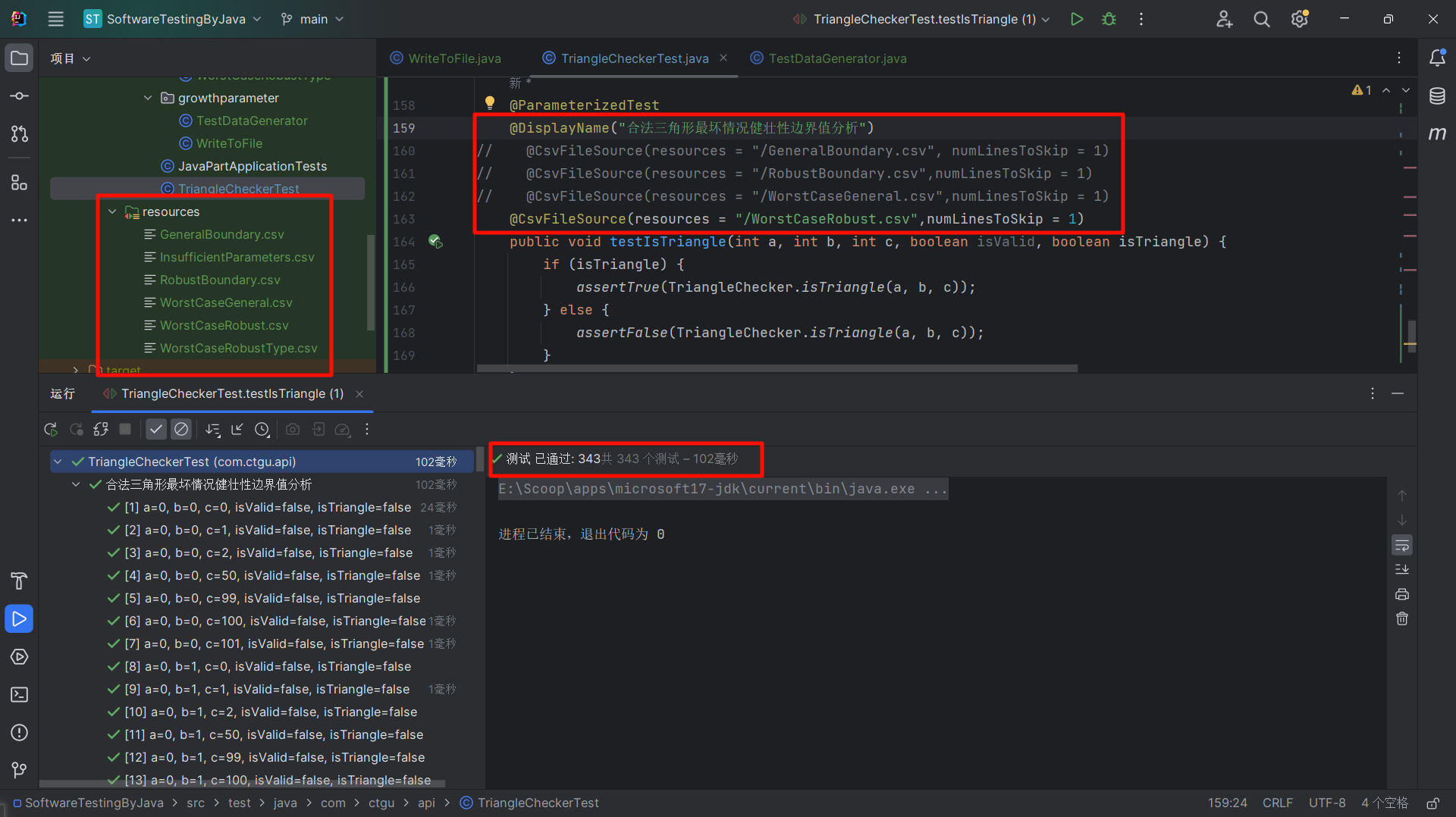
}

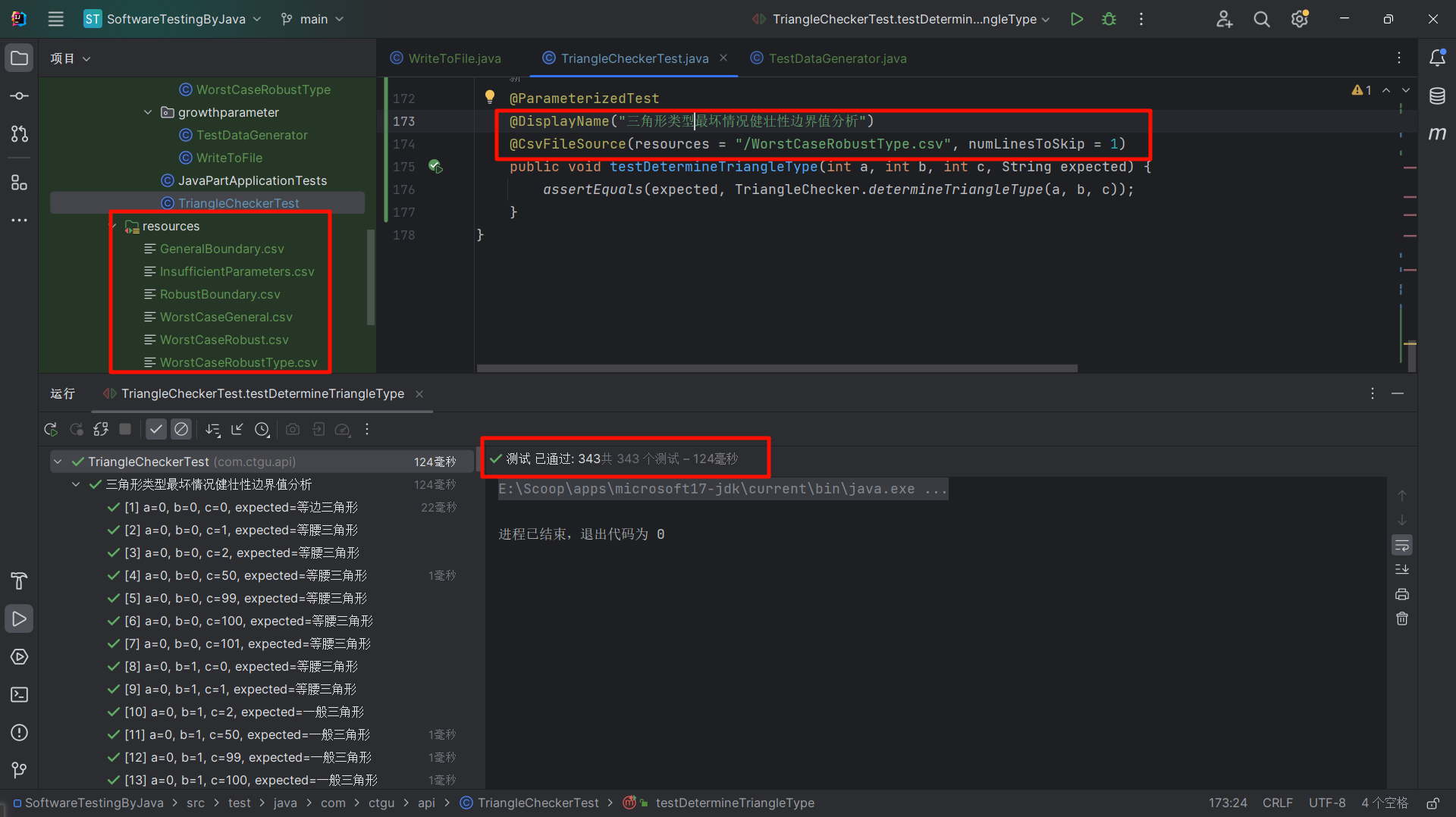
}

#### Idea工具测试用例和覆盖率结果

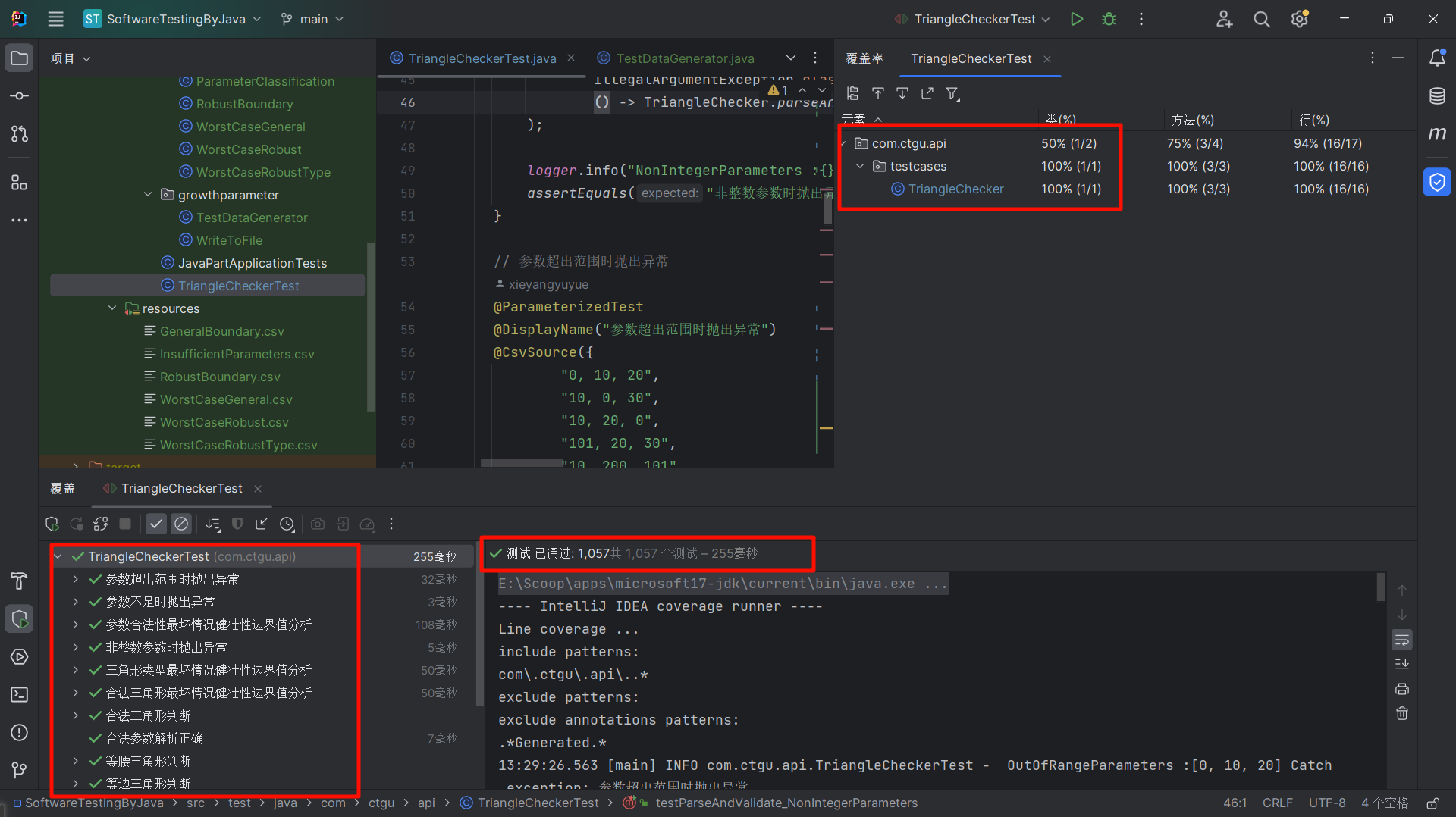
##### 2.1.2.1对于边界值分析方法 我主要示范最坏情况健壮性边界值分析方法







##### 2.1.2.2参数集成化 覆盖率测试



### golang部分

#### 2.2.1导入csv工具类

package testutils

import (

"encoding/csv"

"fmt"

"os"

"strconv"

)

// AddTestCase 定义测试用例结构体

type AddTestCase struct {

Name string

A int

B int

C int

IsValid bool

IsTriangle bool

}

// LoadCSVTestCases 加载CSV测试用例

func LoadCSVTestCases(path string, skipHeader bool)([] AddTestCase, error) {

file, err: = os.Open(path)

if err != nil {

return nil, fmt.Errorf("打开文件失败: %w", err)

}

defer func(file \* os.File) {

\_ = file.Close()

}(file)

reader: = csv.NewReader(file)

lines, err: = reader.ReadAll()

if err != nil {

return nil, fmt.Errorf("读取CSV失败: %w", err)

}

var cases[] AddTestCase

startIndex: = 0

if skipHeader && len(lines) > 0 {

startIndex = 1

}

for i: = startIndex;

i < len(lines);

i++{

line: = lines[i]

if len(line) < 5 {

return nil, fmt.Errorf("第%d行数据不完整", i + 1)

}

a,

err: = strconv.Atoi(line[0])

if err != nil {

return nil, fmt.Errorf("第%d行A值转换错误: %v", i + 1, err)

}

b,

err: = strconv.Atoi(line[1])

if err != nil {

return nil, fmt.Errorf("第%d行B值转换错误: %v", i + 1, err)

}

c,

err: = strconv.Atoi(line[2])

if err != nil {

return nil, fmt.Errorf("第%d行C值转换错误: %v", i + 1, err)

}

isValid,

err: = strconv.ParseBool(line[3])

if err != nil {

return nil, fmt.Errorf("第%d行isValid转换错误: %v", i + 1, err)

}

isTriangle,

err: = strconv.ParseBool(line[4])

if err != nil {

return nil, fmt.Errorf("第%d行isTriangle转换错误: %v", i + 1, err)

}

name: = fmt.Sprintf("Test\_%d", i + 1)

cases = append(cases, AddTestCase {

Name: name,

A: a,

B: b,

C: c,

IsValid: isValid,

IsTriangle: isTriangle,

})

}

return cases, nil

}

#### 2.2.2测试用例和测试代码

package main

import (

"errors"

"strconv"

)

// parseAndValidate 解析并验证输入参数

func parseAndValidate(args[] string)(int, int, int, error) {

if len(args) != 3 {

return 0, 0, 0, errors.New("参数不足")

}

a, err1: = strconv.Atoi(args[0])

b, err2: = strconv.Atoi(args[1])

c, err3: = strconv.Atoi(args[2])

if err1 != nil || err2 != nil || err3 != nil {

return 0, 0, 0, errors.New("参数超出范围时抛出异常")

}

if a < 1 || a > 100 || b < 1 || b > 100 || c < 1 || c > 100 {

return 0, 0, 0, errors.New("非整数参数时抛出异常")

}

return a, b, c, nil

}

// isTriangle 判断是否为三角形

func isTriangle(a, b, c int) bool {

return a + b > c && a + c > b && b + c > a

}

// determineTriangleType 判断三角形类型

func determineTriangleType(a, b, c int) string {

if a == b && b == c {

return "等边三角形"

} else if a == b || a == c || b == c {

return "等腰三角形"

} else {

return "一般三角形"

}

}

package main

import (

"SoftwareTestingByGolang/testutils"

"fmt"

"path/filepath"

"testing"

)

func TestTriangle(t \* testing.T) {

//csvPath := filepath.Join("testdata", "GeneralBoundary.csv")

//csvPath := filepath.Join("testdata", "RobustBoundary.csv")

//csvPath := filepath.Join("testdata", "WorstCaseGeneral.csv")

csvPath: = filepath.Join("testdata", "WorstCaseRobust.csv")

testCases,

err: = testutils.LoadCSVTestCases(csvPath, true)

if err != nil {

t.Fatalf("加载测试数据失败: %v", err)

}

for \_,

tc: = range testCases {

t.Run(tc.Name, func(t \* testing.T) {

// 参数验证测试

\_, \_, \_, err: = parseAndValidate([] string {

fmt.Sprint(tc.A),

fmt.Sprint(tc.B),

fmt.Sprint(tc.C),

})

if tc.IsValid && err != nil {

t.Errorf("预期有效但返回错误: %v", err)

}

if !tc.IsValid && err == nil {

t.Error("预期无效但未返回错误")

}

// 三角形判断测试

if tc.IsValid {

got: = isTriangle(tc.A, tc.B, tc.C)

if got != tc.IsTriangle {

t.Errorf("输入 (%d, %d, %d) 预期 %v 实际 %v",

tc.A, tc.B, tc.C, tc.IsTriangle, got)

}

}

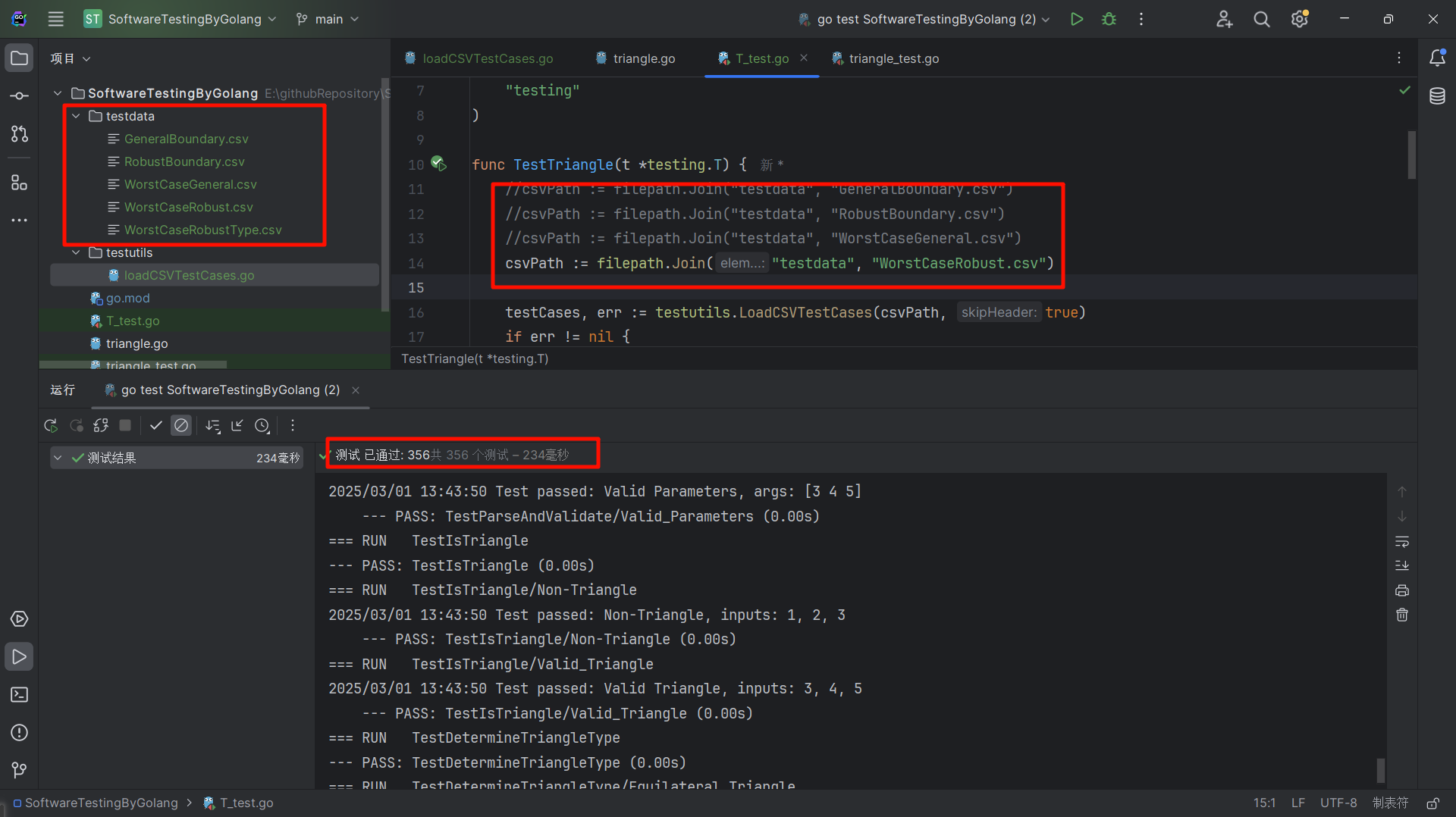
})

}

}

#### 2.2.3Goland工具测试用例和覆盖率结果

##### 2.2.3.1对于边界值分析方法 我主要示范最坏情况健壮性边界值分析方法



# 五、结论分析与体会

这次我使用csv格式导入进行一般边界值方法 健壮性边界值方法 最坏情况一般边界值分析方法 最坏情况健壮性边界值分析方法等测试 我采取使用程序生成测试数据并写入相应的目录下面 我使用的junit 可以自动化导入csv格式数据 在golang中需要自己读取文件存储数据

在测试代码中集成日志记录功能，能够帮助开发者更好地跟踪测试执行过程，尤其是在测试失败时，日志可以提供详细的上下文信息，便于快速定位问题。我们使用了 Go 标准库 log 来实现日志记录，并在测试通过或失败时分别记录了相关信息。

测试与报告的依赖关系：无论是 Go 还是 Java，生成报告的前提都是先运行测试。例如，单独执行 mvn jacoco:report 会失败，必须通过 mvn test jacoco:report 确保测试数据已生成。

Go 语言通过内置的 go test 工具实现了开箱即用的测试能力，配合 -cover 和 -json 参数可以快速生成覆盖率报告和结构化测试结果。

Java 的生态复杂性：Java 依赖 Maven/Gradle 等构建工具，通过插件（如 Surefire、Jacoco）实现测试和覆盖率统计。

# 仓库地址

https://github.com/xieyangyuyue/SoftwareTesting

[xieyangyuyue/SoftwareTesting: 软件测试--java (github.com)](https://github.com/xieyangyuyue/SoftwareTesting)