**《软件测试》**

**软件测试实验四：决策表分析方法**

**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： 2025.3.14**

**总评成绩： 审阅教师： 杨青**

# 实验目的：

1. 巩固 git 协作式管理工具的使用；
2. 巩固 gradle 项目构建 Java 应用的基本操作
3. 掌握决策表分析方法设计测试用例的方法

# **实验要求**

1. 实现根据输入日期，计算输出其前一天日期的功能
2. 采用决策表分析方法设计测试用例，并采用Junit5 编写YesterDate程序测试用例，并提交到代码仓库
3. 使用checkstyle生成代码规范检查报告
4. 使用jacoco生成测试覆盖率报告
5. 使用spotbugs插件检查代码，生成代码安全检查报告，并修复检查出来的bug
6. 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，并把代码规范检查报告，jacoco测试覆盖率报告、spotbugs报告一同打包，以“学号-姓名-软件测试实验四”命名，提交到雨课堂“软件测试实验四”

# 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

# 实验要求：

1. 实现电话账单收费功能
2. 采用边等价类分析方法设计测试用例，并采用Junit5 编写电话账单收费程序测试用例，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-软件测试实验三”命名，提交到雨课堂“软件测试实验三”

# 实验步骤与内容

## 需求：根据输入日期，计算输出其前一天的日期

YesterDate函数（计算前一天日期）

对给定的year、month、day，YesterDate函数应返回这一天的前一天的日期。

例如：

输入日期为2025年3月10日，则输出应该为2025年3月9日。

输入日期为2025年3月1日，则输出应该为2025年2月28日。

输入日期为2025年1月1日，则输出应该为2024年12月31日。

### **采用如下规则：**

规则1：1 ≤ month ≤ 12

规则2：1 ≤ day ≤ 31

规则3：1900 ≤ year ≤ 2050

## 代码实现：

| **用例编号** | **输入日期** | **预期输出** | **说明** |  |
| --- | --- | --- | --- | --- |
| 1 | 2025-05-20 | 2025-05-19 | 普通日期减一 |  |
| 2 | 2025-05-01 | 2025-04-30 | 跨月到小月（30天） |  |
| 3 | 2025-08-01 | 2025-07-31 | 跨月到大月（31天） |  |
| 4 | 202-03-01 | 2024-02-29 | 闰年2月 |  |
| 5 | 2025-03-01 | 2025-02-28 | 非闰年2月 |  |
| 6 | 2025-01-01 | 2024-12-31 | 跨年 |  |
| 7 | 2025-12-31 | 2025-12-30 | 12月末日（非首日） |  |
| 8 | 2024-02-29 | 2024-02-28 | 闰年2月末日 |  |
| 9 | 2025-02-28 | 2025-02-27 | 非闰年2月末日 |  |

### Java部分

**YesterdayDate.java**

package cn.edu.ctgu;

public class YesterdayDate {

public static String getYesterday(int year, int month, int day) {

// 输入合法性检测

validateInput(year, month, day);

if (day > 1) {

day--;

} else {

// 处理跨月或跨年

if (month == 1) {

// 上一年的12月

year--;

month = 12;

day = 31;

} else {

month--;

// 获取上个月的天数

day = getDaysInMonth(year, month);

}

}

return String.format("%d-%02d-%02d", year, month, day);

}

private static void validateInput(int year, int month, int day) {

// 检测年份合法性

if (year < 0) {

throw new IllegalArgumentException("年份不能为负数: " + year);

}

// 检测月份合法性

if (month < 1 || month > 12) {

throw new IllegalArgumentException("月份必须为1-12之间的整数: " + month);

}

// 检测日期合法性

int maxDay = getDaysInMonth(year, month);

if (day < 1 || day > maxDay) {

throw new IllegalArgumentException(

String.format("日期无效：%d年%d月不存在第%d天", year, month, day)

);

}

}

private static int getDaysInMonth(int year, int month) {

switch (month) {

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12:

return 31;

case 4:

case 6:

case 9:

case 11:

return 30;

case 2:

return isLeapYear(year) ? 29 : 28;

default:

throw new IllegalArgumentException("Invalid month: " + month);

}

}

private static boolean isLeapYear(int year) {

return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);

}

}

**YesterdayDateTest.java**

package cn.edu.ctgu;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.junit.jupiter.api.Assertions.assertThrows;

public class YesterdayDateTest {

@Test

void testRegularDay() {

assertEquals("2025-05-19", YesterdayDate.getYesterday(2025, 5, 20));

}

@Test

void testCrossTo30DayMonth() {

assertEquals("2025-04-30", YesterdayDate.getYesterday(2025, 5, 1));

}

@Test

void testCrossTo31DayMonth() {

assertEquals("2025-07-31", YesterdayDate.getYesterday(2025, 8, 1));

}

@Test

void testLeapYearFebruary() {

assertEquals("2024-02-29", YesterdayDate.getYesterday(2024, 3, 1));

}

@Test

void testNonLeapYearFebruary() {

assertEquals("2025-02-28", YesterdayDate.getYesterday(2025, 3, 1));

}

@Test

void testCrossYear() {

assertEquals("2024-12-31", YesterdayDate.getYesterday(2025, 1, 1));

}

@Test

void testEndOfDecember() {

assertEquals("2025-12-30", YesterdayDate.getYesterday(2025, 12, 31));

}

@Test

void testLeapYearEndOfFebruary() {

assertEquals("2024-02-28", YesterdayDate.getYesterday(2024, 2, 29));

}

@Test

void testNonLeapYearEndOfFebruary() {

assertEquals("2025-02-27", YesterdayDate.getYesterday(2025, 2, 28));

}

// 新增测试：输入非法月份

@Test

void testInvalidMonth() {

assertThrows(IllegalArgumentException.class, () ->

YesterdayDate.getYesterday(2025, 0, 1),

"月份为0时应抛出异常"

);

assertThrows(IllegalArgumentException.class, () ->

YesterdayDate.getYesterday(2025, 13, 1),

"月份为13时应抛出异常"

);

}

// 新增测试：输入非法日期

@Test

void testInvalidDay() {

// 非闰年2月最多28天

assertThrows(IllegalArgumentException.class, () ->

YesterdayDate.getYesterday(2025, 2, 29),

"2023年2月29日不存在"

);

// 4月只有30天

assertThrows(IllegalArgumentException.class, () ->

YesterdayDate.getYesterday(2025, 4, 31),

"4月31日不存在"

);

}

// 新增测试：输入非法年份

@Test

void testInvalidYear() {

assertThrows(IllegalArgumentException.class, () ->

YesterdayDate.getYesterday(-1, 1, 1),

"年份不能为负数"

);

}

// 新增测试：边界值检测（最小合法日期）

@Test

void testMinValidDate() {

assertEquals("0-01-01", YesterdayDate.getYesterday(0, 1, 2));

assertThrows(IllegalArgumentException.class, () ->

YesterdayDate.getYesterday(0, 1, 0),

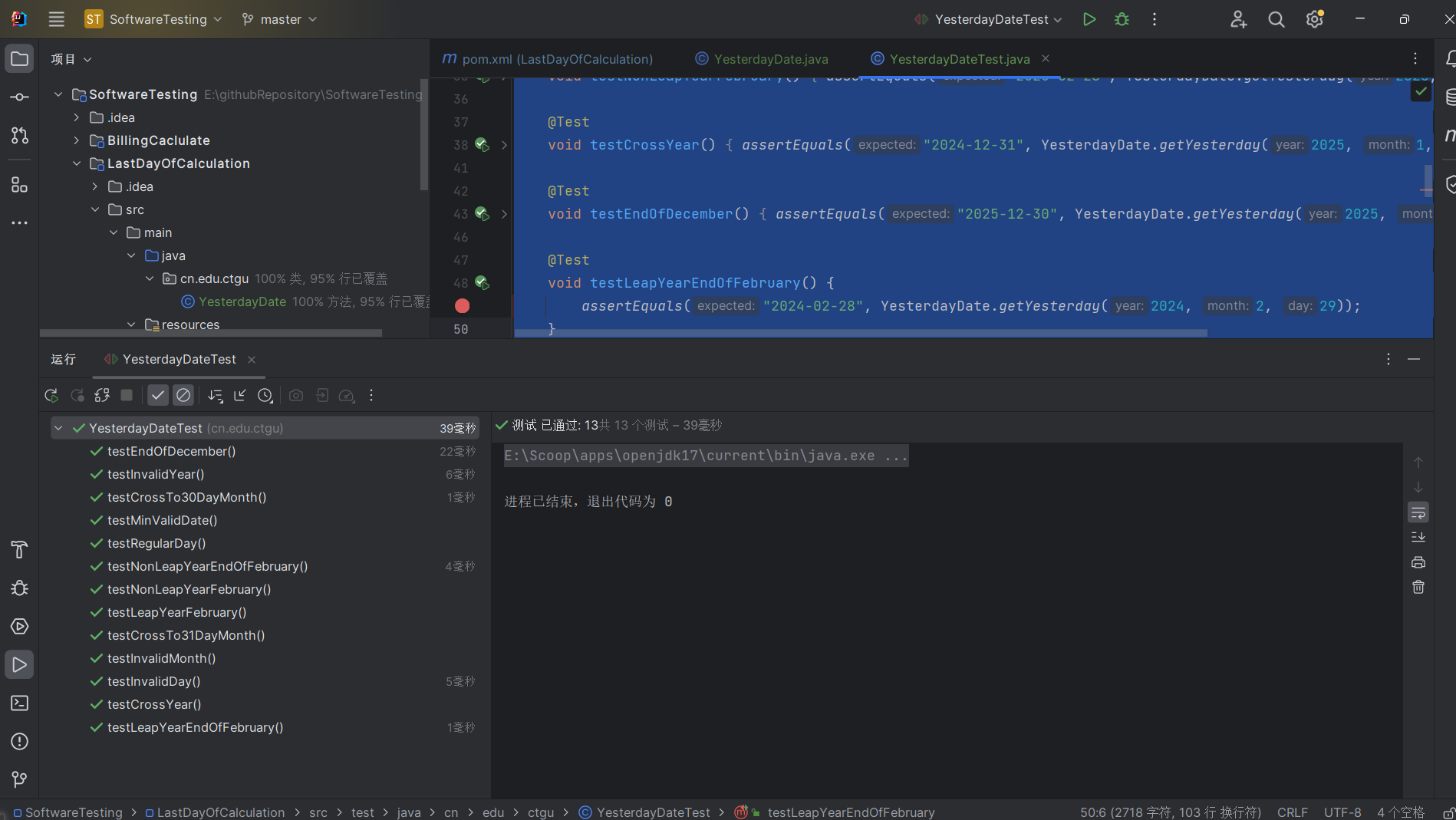
"日期不能为0"

);

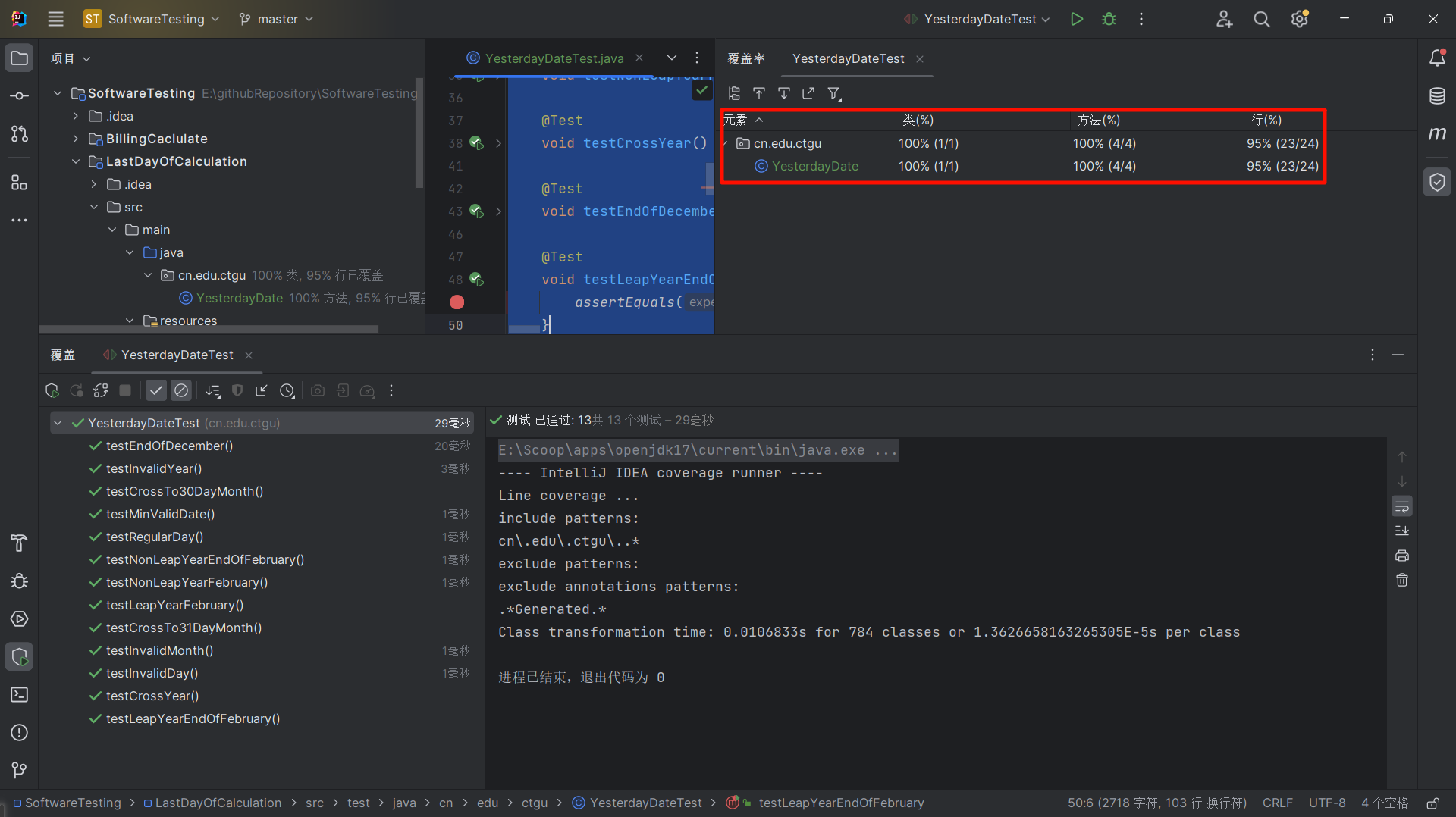
}

}

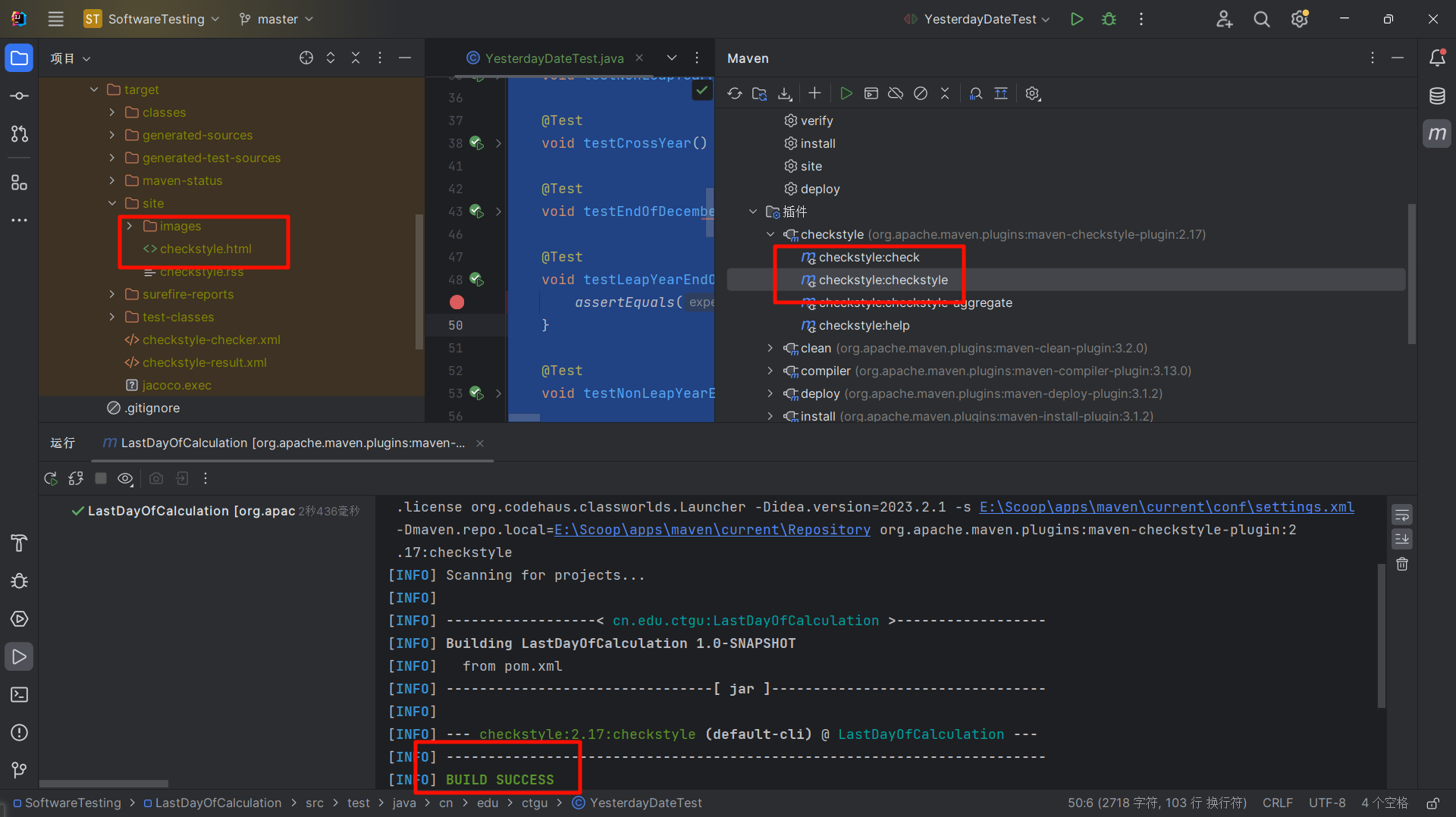
#### Idea工具测试用例 覆盖率结果 checkstyle jacoco

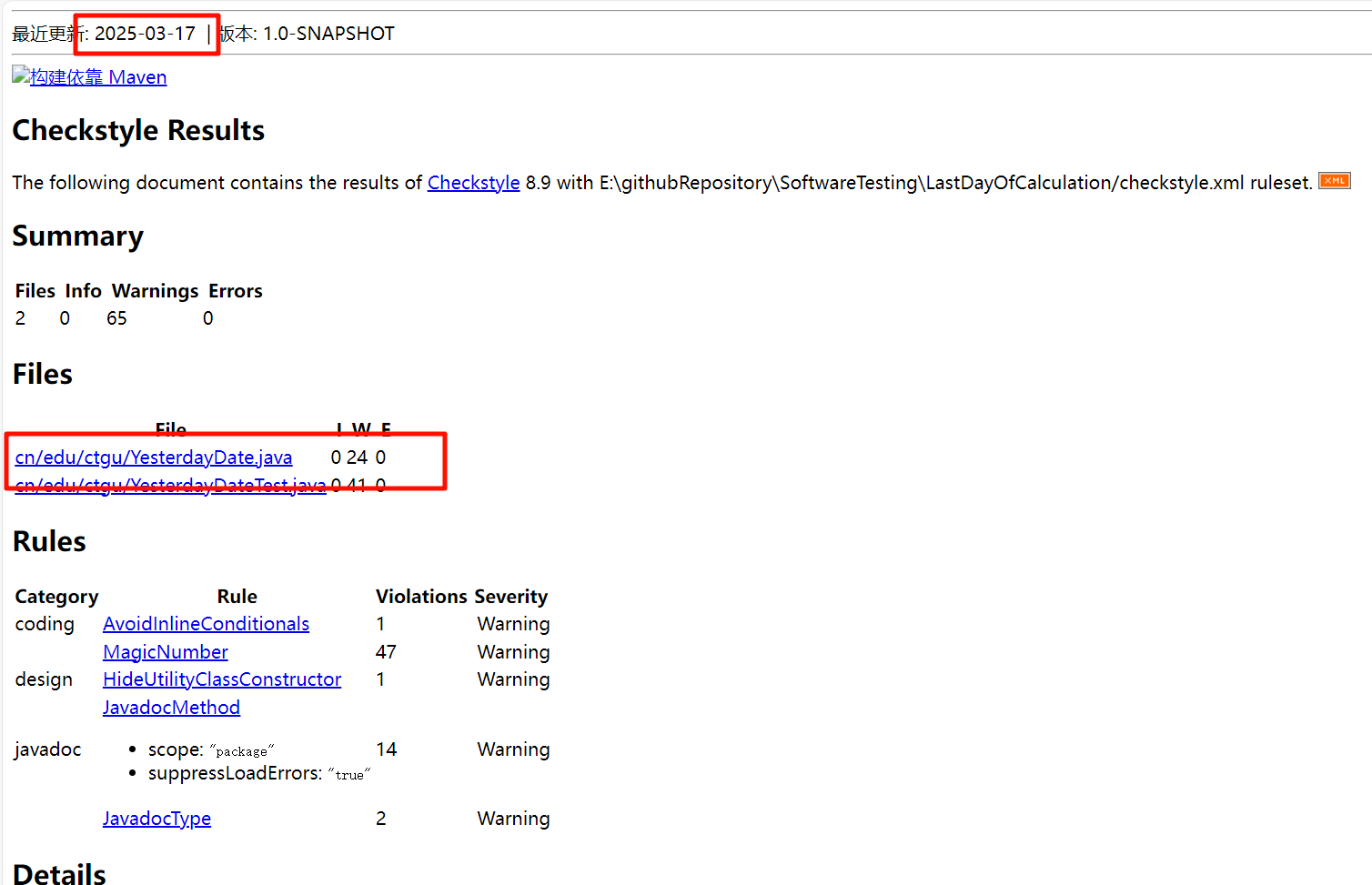


**覆盖率**

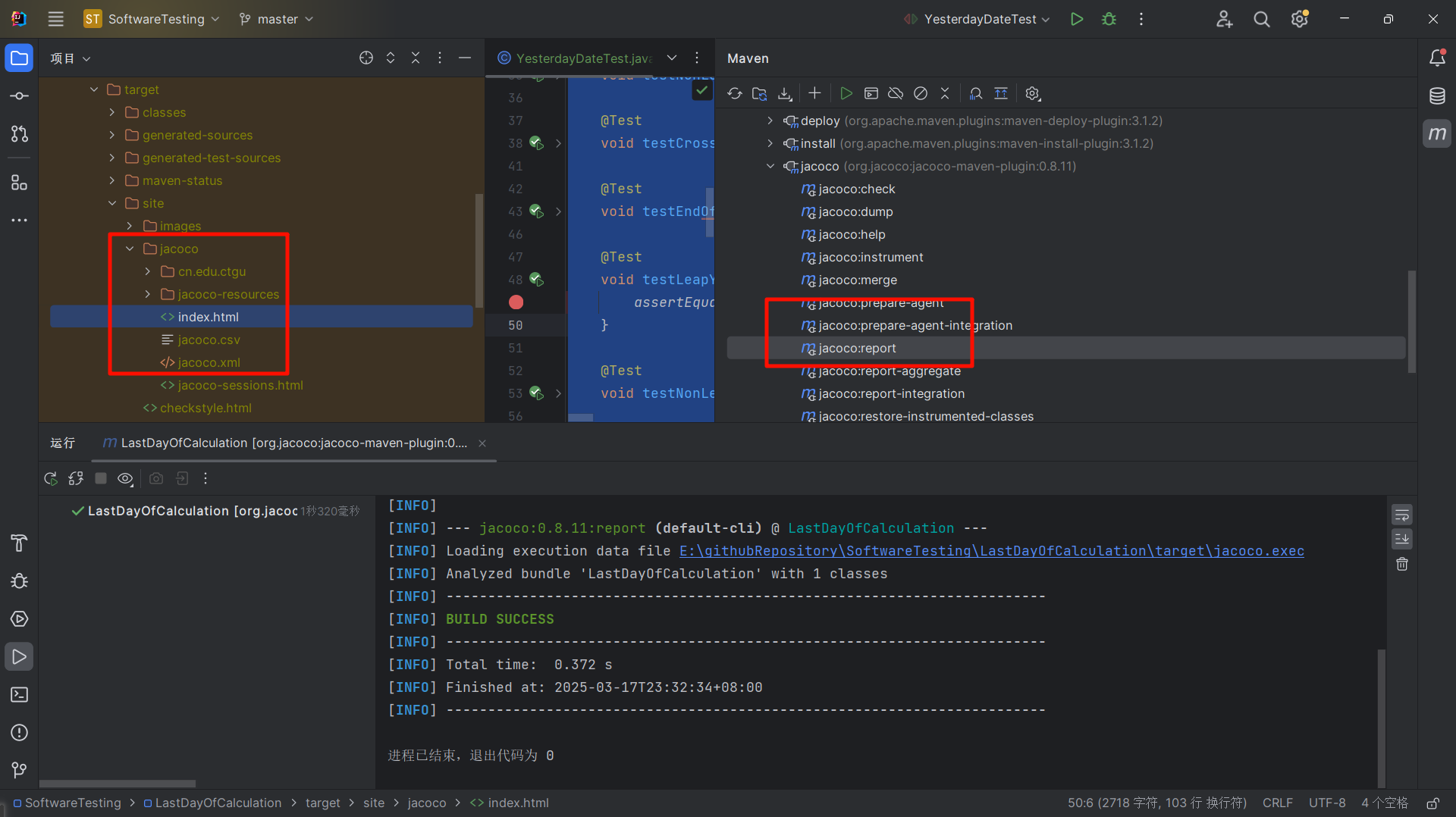


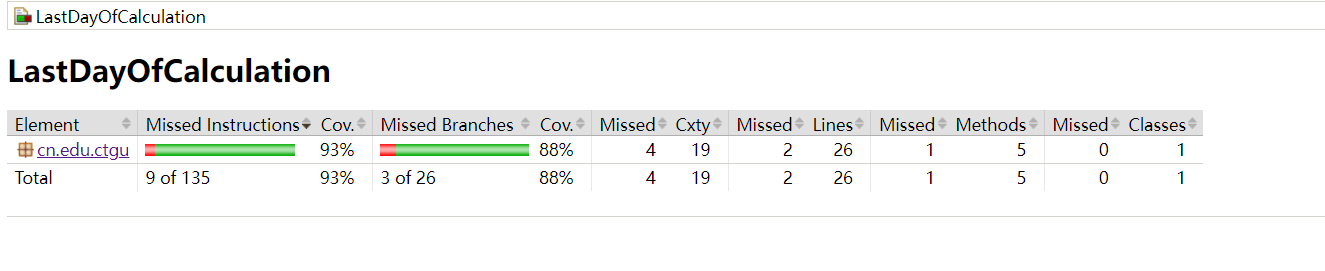
**Checkstyle**



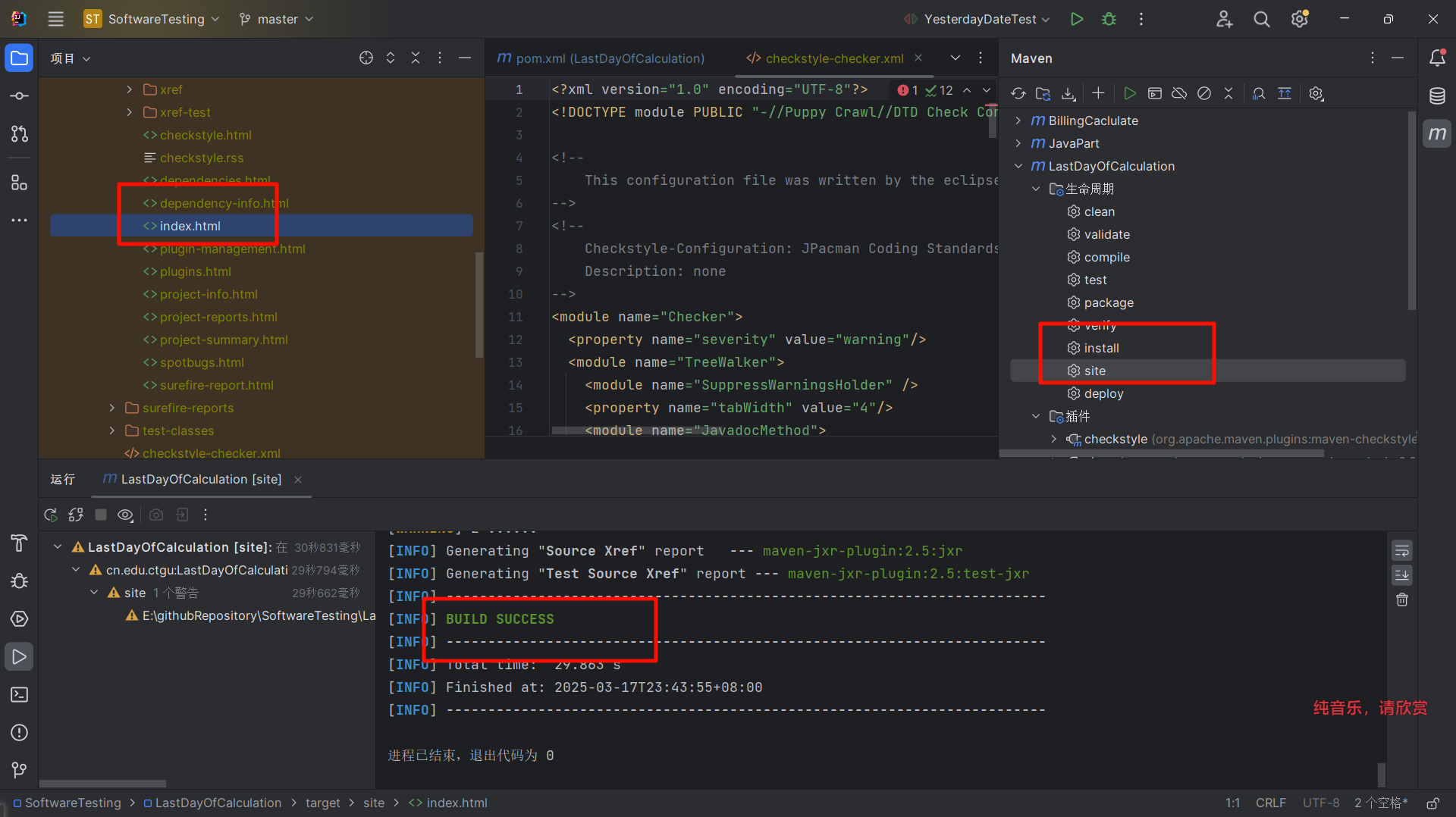


**Jacoco**





**spotbugs报告:**





# 五、结论分析与体会

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。单独执行 mvn checkstyle:checkstyle会失败，必须通过 mvn test checkstyl:checkstyle确保测试数据已生成。

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。使用chcekstyle进行静态调整代码规范

spotbugs静态代码分析 发现潜在缺陷（空指针、资源泄漏、线程安全等）。识别代码异味（冗余逻辑、低效实现）。

# 仓库地址

https://github.com/xieyangyuyue/SoftwareTesting

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