2024年春季学期
计算学部《软件工程》课程

实验报告

Lab3 代码评审与单元测试

姓名	学号	联系方式
张智雄	2021112845	1557588606@qq.com
韩晨烨	2021111247	hanchenye1122@gmail.com

目　录

1 实验要求

代码审计：

1) 按照 Lab1 分组，两人共同完成实验。
2) 针对 Lab1 所完成的代码，进行代码评审(走查)，使用 Checkstyle 和 SpotBugs 从代码规范性和正确性角度对代码进行评价。

单元测试

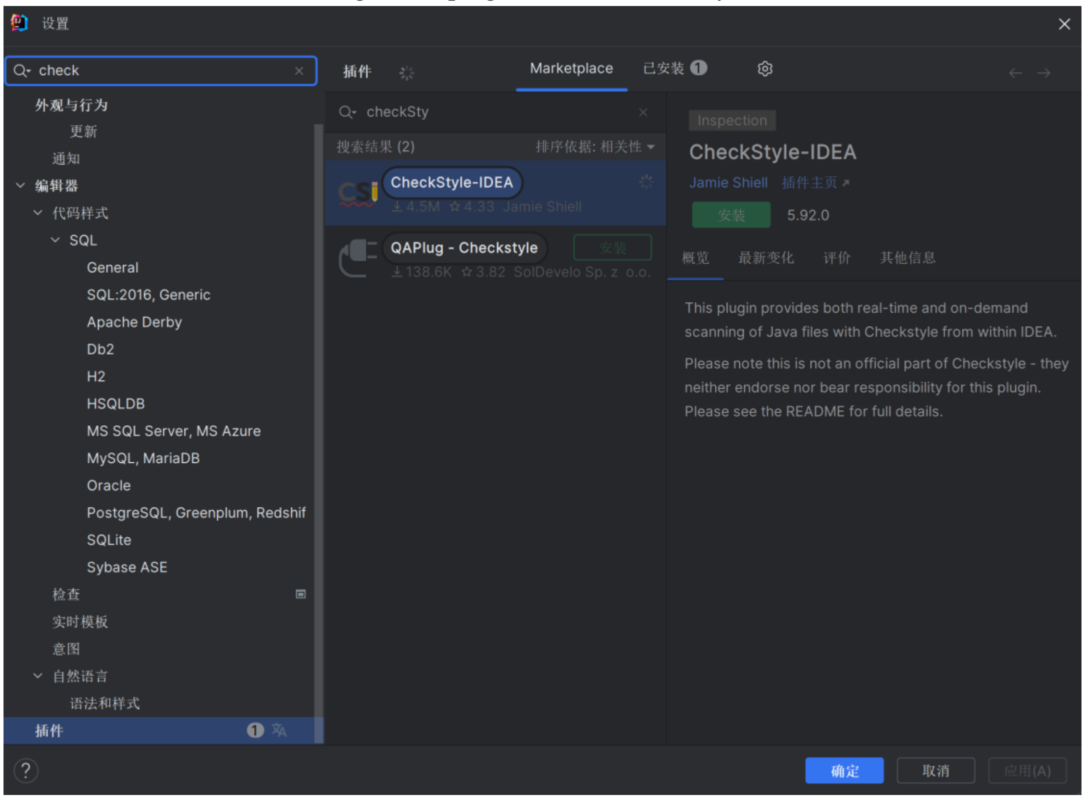
1) 按 Lab1 分组，两人共同完成实验。
2) 设计黑盒测试用例和白盒测试用例。
3) 在 JUnit 环境下撰写测试代码并执行测试。
4) 使用 EclEmma 或 IDE 自带工具统计测试的覆盖度。

2 在 IDE 中配置代码审查与分析工具

简要描述在 IDE 中安装和配置 Checkstyle、SpotBugs、EclEmma、JUnit 等插件或 IDE 自带插件的过程。

2.1 Checkstyle

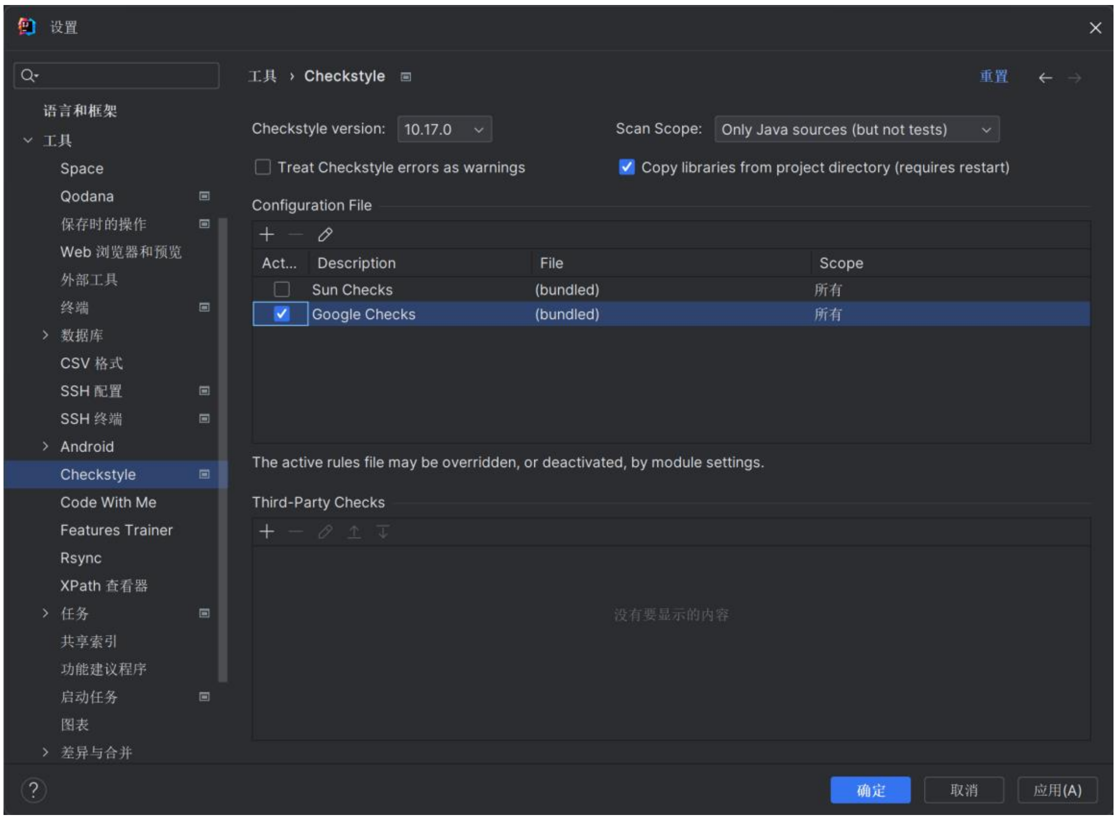
打开 IDEA 菜单栏 Settings 中的 plugins，搜索 CheckStyle-IDEA 并安装:



This plugin provides both real-time and on-demand scanning of Java files with Checkstyle from within IDEA.

Please note this is not an official part of Checkstyle - they neither endorse nor bear responsibility for this plugin. Please see the README for full details.

选择内置的 google checks 作为插件的配置文件


设置

工具 › Checkstyle

Checkstyle version: 10.17.0 Scan Scope: Only Java sources (but not tests)

☐ Treat Checkstyle errors as warnings
☑ Copy libraries from project directory (requires restart)

Configuration File

Act...	Description	File	Scope
☐	Sun Checks	(bundled)	所有
☑	Google Checks	(bundled)	所有

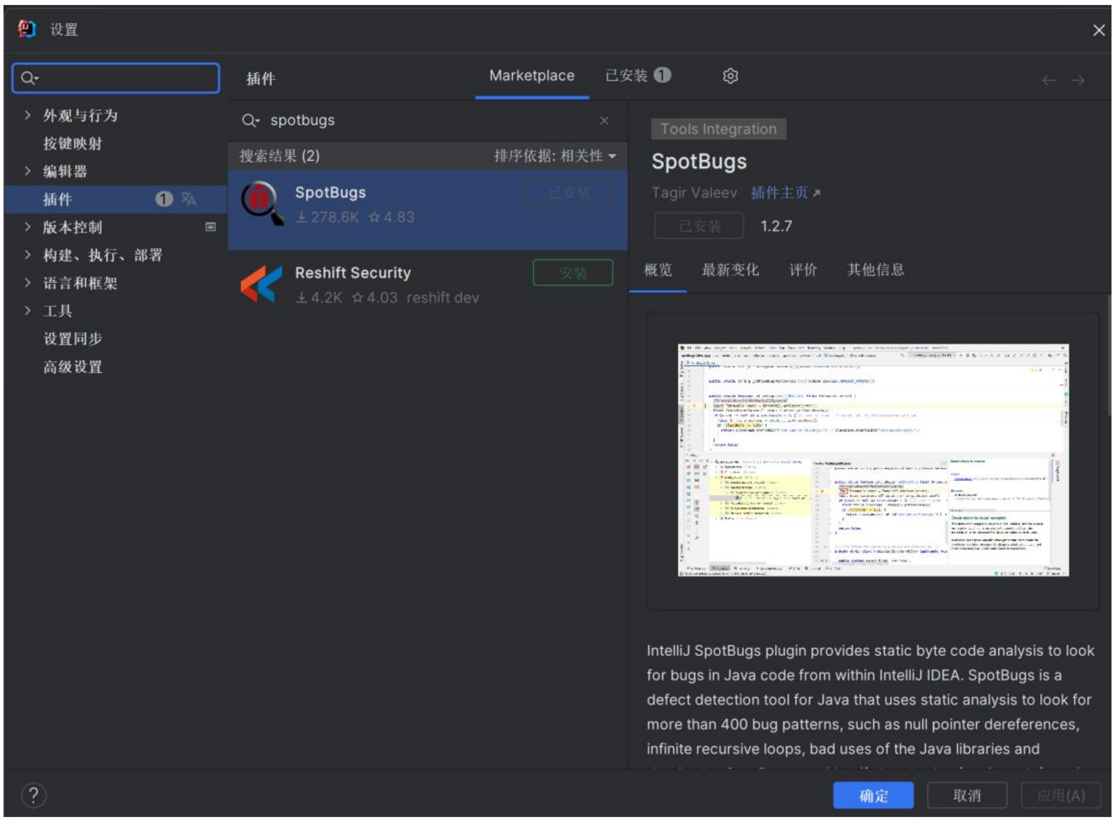
The active rules file may be overridden, or deactivated, by module settings.

Third-Party Checks

没有要显示的内容

2.2 SpotBugs

搜索并下载 spotbugs 插件


设置

插件 Marketplace 已安装 1

spotbugs

搜索结果 (2) 排序依据: 相关性

SpotBugs ⤓278.6K ☆4.83 已安装

Reshift Security ⤓4.2K ☆4.03 reshift dev 安装

Tools Integration

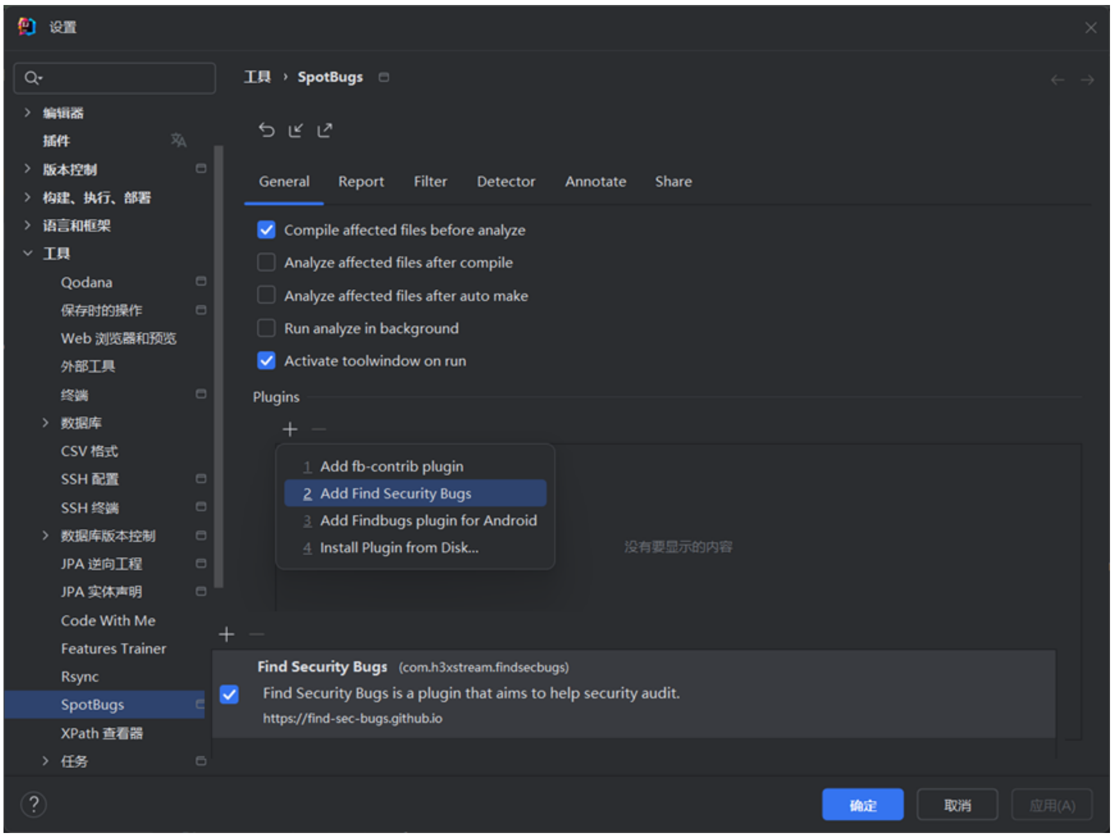
SpotBugs

Tagir Valeev 插件主页 ↗

已安装 1.2.7

IntelliJ SpotBugs plugin provides static byte code analysis to look for bugs in Java code from within IntelliJ IDEA. SpotBugs is a defect detection tool for Java that uses static analysis to look for more than 400 bug patterns, such as null pointer dereferences, infinite recursive loops, bad uses of the Java libraries and

并点击加号配置相关规则



- [✓] Compile affected files before analyze
- [ ] Analyze affected files after compile
- [ ] Analyze affected files after auto make
- [ ] Run analyze in background
- [✓] Activate toolwindow on run

Plugins

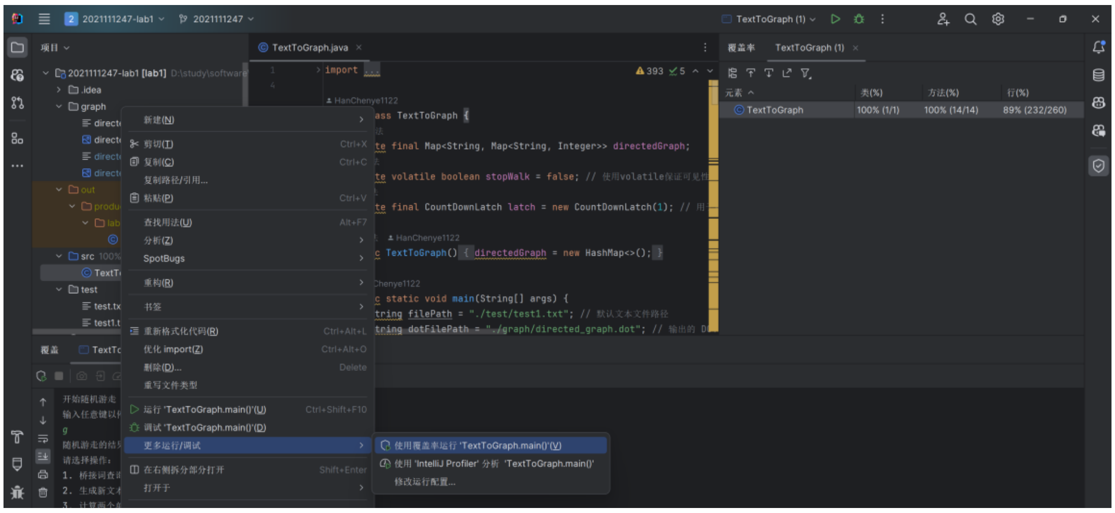
1. Add fb-contrib plugin
2. Add Find Security Bugs
3. Add Findbugs plugin for Android
4. Install Plugin from Disk...

没有要显示的内容

[✓] **Find Security Bugs** (com.h3xstream.findsecbugs)
Find Security Bugs is a plugin that aims to help security audit.
https://find-sec-bugs.github.io

2.3 EclEmma

IDEA 使用自带的覆盖率测试软件


2021111247-lab1
2021111247
TextToGraph (1)
项目
TextToGraph.java
覆盖率
新建(N)
剪切(T) Ctrl+X
复制(C) Ctrl+C
复制路径/引用...
粘贴(P) Ctrl+V
查找用法(U) Alt+F7
分析(Z)
SpotBugs
重构(R)
书签
重新格式化代码(R) Ctrl+Alt+L
优化 import(Z) Ctrl+Alt+O
删除(D)... Delete
重写文件类型
运行 'TextToGraph.main()'(U) Ctrl+Shift+F10
调试 'TextToGraph.main()'(D)
更多运行/调试
在右侧拆分部分打开 Shift+Enter
打开于
使用覆盖率运行 'TextToGraph.main()'(V)
使用 'IntelliJ Profiler' 分析 'TextToGraph.main()'
修改运行配置...

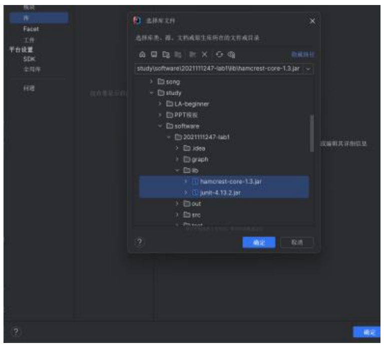
```
import ...

ass TextToGraph {
te final Map<String, Map<String, Integer>> directedGraph;
te volatile boolean stopWalk = false; // 使用volatile保证可见性
te final CountDownLatch latch = new CountDownLatch(1); // 用
c TextToGraph() { directedGraph = new HashMap<>(); }
c static void main(String[] args) {
tring filePath = "./test/test1.txt"; // 默认文本文件路径
tring dotFilePath = "./graph/directed_graph.dot"; // 输出的 D
```

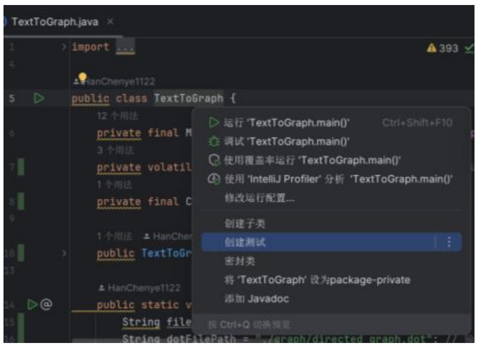
```
开始随机游走
输入任意键以
g
随机游走的结
请选择操作:
1. 桥接词查
2. 生成新文
```

元素	类(%)	方法(%)	行(%)
TextToGraph	100% (1/1)	100% (14/14)	89% (232/260)

2.4 Junit

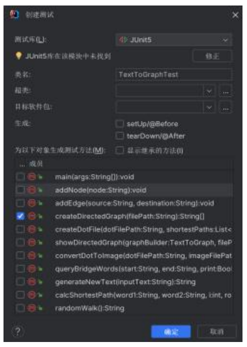
将下载的 junit.jar 和 hamcrest-core.jar ，加入项目的 Libraries 中




TextToGraph.java

```
import ...

public class TextToGraph {
    private final M
    private volatil
    private final C

    public TextToGr

    public static v
        String file
        String dotFilePath = "./graph/directed_graph.dot"; //
```

HanChenye1122
12 个用法
3 个用法
1 个用法
▲393
运行 'TextToGraph.main()'　Ctrl+Shift+F10
调试 'TextToGraph.main()'
使用覆盖率运行 'TextToGraph.main()'
使用 'IntelliJ Profiler' 分析 'TextToGraph.main()'
修改运行配置...
创建子类
创建测试
密封类
将 'TextToGraph' 设为package-private
添加 Javadoc
按 Ctrl+Q 切换预览


创建测试

测试库(L): JUnit5
JUnit5库在该模块中未找到　修正
类名: TextToGraphTest
超类:
目标软件包:
生成: ☐ setUp/@Before　☐ tearDown/@After
为以下对象生成测试方法(M): ☐ 显示继承的方法(I)

	成员
☐	main(args:String[]):void
☐	addNode(node:String):void
☐	addEdge(source:String, destination:String):void
✓	createDirectedGraph(filePath:String):String[]
☐	createDotFile(dotFilePath:String, shortestPaths:List<
☐	showDirectedGraph(graphBuilder:TextToGraph, fileP
☐	convertDotToImage(dotFilePath:String, imageFilePat
☐	queryBridgeWords(start:String, end:String, print:Boo
☐	generateNewText(inputText:String):String
☐	calcShortestPath(word1:String, word2:String, i:int, ro
☐	randomWalk():String

在需要测试的类或接口名称上 Alt+Enter 选择创建测试，并选择我们需要测试的函数，此处我们选择 createDirectedGraph 来进行测试。

3　Checkstyle 所发现的代码问题清单及原因分析

针对同种类型的问题，只需要列出一个典型代表即可。

编号	问题描述	类型	所在代码行号	修改策略
1	不应使用'.*'形式的导入-java.io.*。	AvoidStarImport	1	将*改为明确导入具体类
2	'member def modifier' 缩进了 4 个缩进符，应为 2 个。	Indentation	6	修改缩进符数量
3	缺少 Javadoc 。	Missing Javadoc Method	14	在代码中添加 Javadoc 注释
4	本行字符数 106 个，最多:100 个。	LineLength	25	对代码进行适当的换行和缩进调整
5	第 17 个字符'{应位于前一行。	LeftCurly	34	调整代码中的{位置

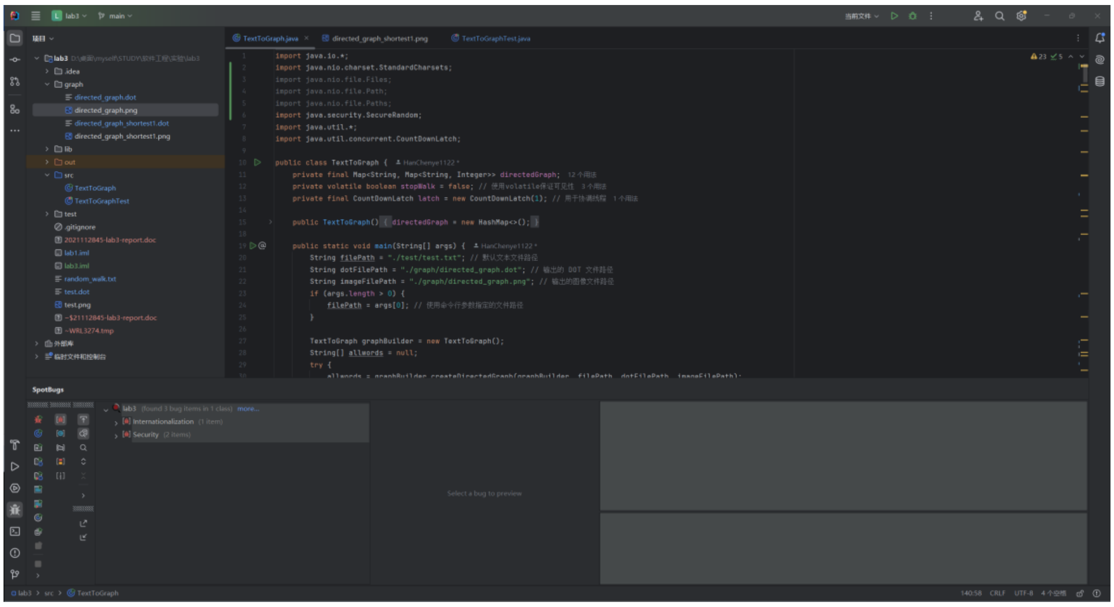
4　SpotBugs 所发现的代码问题清单及原因分析

针对同种类型的问题，只需要列出一个典型代表即可。

优先级	问题描述	违反的规则集	所在代码行号	修改策略
中 Medium Confidence Correctness	异常路径上所有单词可能的空指针解引用	Correctness(Null pointer dereference) NP_NULL_ON_SOME_PATH_EXCEPTION (Possible null pointer dereference in method on exception path)	29	添加必要的空指针检查以防止在异常路径中发生空指针解引用。
中 Medium Confidence Correctness	这个随机生成器(java.util.Random)是可预测的	Security(Predictable Pseudo Random Generator) PREDICTABLE_RANDOM (Predictable pseudorandom number generator)	313 376	使用更安全的随机数生成器，例如java.security.SecureRandom

中 Medium Confidence Correctness	这个 API (java/io/FileReader.(Ljava/lang/String;)V)读取位置可能由用户输入指定的文件	Security (Potential Path Traversal (file read)) PATH_TRAVERSAL_IN (Potential Path Traversal (file read))	149	验证并清理用户输入的文件路径；限制文件读取的目录范围，确保不能访问系统敏感文件。
中 Medium Confidence Correctness	java/lang/ProcessBuilder.([Ljava/lang/String;)V 的用法可能容易受到命令注入的攻击	Security (Command Injection) COMMAND_INJECTION (Potential Command Injection)	219	避免使用用户输入直接构建命令；使用安全的构建命令的方法，验证和清理用户输入。
中 Medium Confidence Dodgy code	存储在 randomWalk()方法的 visitedEdges 变量中的无用对象	Dodgy code (Useless code) UC_USELESS_OBJECT (Useless object created)	383	删除未使用的变量或确保变量被正确使用。
高 High Confidence Dodgy code	死仓库到路径	Dodgy code (Dead local store) DLS_DEAD_LOCAL_STORE (Dead store to local variable)	234	删除未使用的变量或确保变量被正确使用。
高 High Confidence Internationalization	发现对默认编码的依赖:new java.util.Scanner(InputStream)	Internationalization (Dubious method used) DM_DEFAULT_ENCODING (Reliance on default encoding)	30 149 173 409	明确指定编码以避免依赖默认编码。

经修改后，spotsbug 发现的代码问题大大减少，余下均为中等风险且无法在不改变现有代码逻辑的情形下完成：



```
import java.io.*;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.security.SecureRandom;
import java.util.*;
import java.util.concurrent.CountDownLatch;

public class TextToGraph {
    private final Map<String, Map<String, Integer>> directedGraph;
    private volatile boolean stopWalk = false; // 使用volatile保证可见性
    private final CountDownLatch latch = new CountDownLatch(1); // 用于协调线程

    public TextToGraph() { directedGraph = new HashMap<>(); }

    public static void main(String[] args) {
        String filePath = "./test/test.txt"; // 默认文本文件路径
        String dotFilePath = "./graph/directed_graph.dot"; // 输出的 DOT 文件路径
        String imageFilePath = "./graph/directed_graph.png"; // 输出的图像文件路径
        if (args.length > 0) {
            filePath = args[0]; // 使用命令行参数指定的文件路径
        }

        TextToGraph graphBuilder = new TextToGraph();
        String[] allwords = null;
        try {
            allwords = graphBuilder.createDirectedGraph(graphBuilder, filePath, dotFilePath, imageFilePath);
```

```
lab3 (found 3 bug items in 1 class) more...
  Internationalization (1 item)
  Security (2 items)
```

5 针对 Lab1 的黑盒测试

5.1 所选的被测函数及其需求规约

函数名称：createDirectedGraph

函数功能：该函数读取指定路径的文本文件，并将文本数据解析为有向图。图的节点为文本中出现的单词（不区分大小写），边表示两个单词在文本中相邻出现的关系，边的权重表示这对单词相邻出现的次数。

输入描述：filePath-字符串类型，表示文本文件的路径。由字符串组成的文件，字符串包含英文字母大小写和符号

输出描述：由该文本文件生成的有向图。

5.2 等价类划分结果

约束条件说明	有效等价类及其编号		无效等价类及其编号	
文本中的换行符应被视为空格。	文本中包含多个换行符。	(1)	文本中没有换行符。	(4)
文本中的任何标点符号应被视为空格。	文本中包含多种标点符号（例如，逗号、句号、感叹号等）。	(2)	文本中没有标点符号。	(5)
文本中的非字母(A-Z, a-z)字符应被忽略。	文本中包含数字和特殊字符。	(3)	文本中仅包含字母字符。	(6)

5.3 测试用例设计

测试用例编号	输入	期望输出	所覆盖的等价类编号

1.	This is the first sentence.
	This is the second sentence!
	And this is the third sentence.
		digraph G {
		this [style=filled, fillcolor=lightgray];
		the -> third [label="1"];
		the -> first [label="1"];
		the -> second [label="1"];
		sentence -> and [label="1"];
		sentence -> this [label="1"];
		third -> sentence [label="1"];
		and -> this [label="1"];
		this -> is [label="3"];
		is -> the [label="3"];
		first -> sentence [label="1"];
		second -> sentence [label="1"];
		}
			(1)

2.	Hello, world! This is an example: should work corr--ectly.	digraph G { hello [style=filled, fillcolor=lightgray]; world -> this [label="1"]; work -> correctly [label="1"]; this -> is [label="1"]; should -> work [label="1"]; is -> an [label="1"]; hello -> world [label="1"]; an -> example [label="1"]; example -> should [label="1"]; }	(2)
3.	Text with numbers 123 and symbols #$@!	digraph G { text [style=filled, fillcolor=lightgray]; with -> numbers [label="1"]; and -> symbols [label="1"]; numbers -> and [label="1"]; text -> with [label="1"]; }	(3)
4.	Continuous text without any newlines just spaces.	digraph G { continuous [style=filled, fillcolor=lightgray]; newlines -> just [label="1"]; continuous -> text [label="1"]; text -> without [label="1"]; any -> newlines [label="1"]; just -> spaces [label="1"]; without -> any [label="1"]; }	(4)
5.	No @ punctuation here just words	digraph G { no [style=filled, fillcolor=lightgray]; here -> just [label="1"]; no -> punctuation [label="1"]; punctuation -> here [label="1"]; just -> words [label="1"]; }	(5)
6.	Just letters no numbers or symbols	digraph G { just [style=filled, fillcolor=lightgray]; no -> numbers [label="1"]; or -> symbols [label="1"]; numbers -> or [label="1"]; just -> letters [label="1"]; letters -> no [label="1"]; }	(6)

5.4 JUnit 测试代码

测试用例编号	JUnit 测试代码
1.	

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

public class TextToGraphTest {

    @Test
    public void testCreateDirectedGraph_case1() throws IOException {
        String input = "This is the first sentence.\n" +
                "This is the second sentence!\n" +
                "And this is the third sentence.\n";

        String expectedOutput = "digraph G {\n" +
                "\tthis [style=filled, fillcolor=lightgray];\n" +
                "\tthe -> third [label=\"1\"];\n" +
                "\tthe -> first [label=\"1\"];\n" +
                "\tthe -> second [label=\"1\"];\n" +
                "\tsentence -> and [label=\"1\"];\n" +
                "\tsentence -> this [label=\"1\"];\n" +
                "\tthird -> sentence [label=\"1\"];\n" +
                "\tand -> this [label=\"1\"];\n" +
                "\tthis -> is [label=\"3\"];\n" +
                "\tis -> the [label=\"3\"];\n" +
                "\tfirst -> sentence [label=\"1\"];\n" +
                "\tsecond -> sentence [label=\"1\"];\n" +
                "}\n";

        // Create a temporary file for testing
        File tempFile = createTempFile(input);

        try {
            TextToGraph graphBuilder = new TextToGraph();
            String[] words = graphBuilder.createDirectedGraph(graphBuilder, tempFile.getAbsolutePath(), "test.dot", "test.png");

            // Read the generated DOT file
            String actualOutput = readFile("test.dot");
            System.out.println("actualOutput: " + actualOutput);

            // Assert the expected output matches the actual output
            assertEquals(expectedOutput, actualOutput);
        } finally {
            // Clean up: delete temporary file
            if (tempFile.exists()) {
                tempFile.delete();
            }
        }
    }


    // Helper method to create a temporary file with given content
    private File createTempFile(String content) throws IOException {
        File tempFile = File.createTempFile("temp", ".txt");
        tempFile.deleteOnExit();

        try (BufferedWriter writer = new BufferedWriter(new FileWriter(tempFile))) {
            writer.write(content);
        }

        return tempFile;
    }

    // Helper method to read content from a file
    private String readFile(String filePath) throws IOException {
        StringBuilder content = new StringBuilder();
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                content.append(line).append("\n");
            }
        }
        return content.toString();
    }
}
```

2.

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

```

```java
public class TextToGraphTest {

    @Test
    public void testCreateDirectedGraph_case2() throws IOException {
        String input = "Hello, world! This is an example: should work correctly.\n";

        String expectedOutput = "digraph G {\n" +
                "\thello [style=filled, fillcolor=lightgray];\n" +
                "\tworld -> this [label=\"1\"];\n" +
                "\twork -> correctly [label=\"1\"];\n" +
                "\tthis -> is [label=\"1\"];\n" +
                "\tshould -> work [label=\"1\"];\n" +
                "\tis -> an [label=\"1\"];\n" +
                "\thello -> world [label=\"1\"];\n" +
                "\tan -> example [label=\"1\"];\n" +
                "\texample -> should [label=\"1\"];\n" +
                "}\n";

        // Create a temporary file for testing
        File tempFile = createTempFile(input);

        try {
            TextToGraph graphBuilder = new TextToGraph();
            String[] words = graphBuilder.createDirectedGraph(graphBuilder, tempFile.getAbsolutePath(), "test.dot", "test.png");

            // Read the generated DOT file
            String actualOutput = readFile("test.dot");

            // Assert the expected output matches the actual output
            assertEquals(expectedOutput, actualOutput);
            System.out.println("测试用例 2 已通过");
        } finally {
            // Clean up: delete temporary file
            if (tempFile.exists()) {
                tempFile.delete();
            }
        }
    }


    // Helper method to create a temporary file with given content
    private File createTempFile(String content) throws IOException {
        File tempFile = File.createTempFile("temp", ".txt");
        tempFile.deleteOnExit();

        try (BufferedWriter writer = new BufferedWriter(new FileWriter(tempFile))) {
            writer.write(content);
        }

        return tempFile;
    }

    // Helper method to read content from a file
    private String readFile(String filePath) throws IOException {
        StringBuilder content = new StringBuilder();
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                content.append(line).append("\n");
            }
        }
        return content.toString();
    }
}
```

3.

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

public class TextToGraphTest {

    @Test
    public void testCreateDirectedGraph_case3() throws IOException {
        String input = "Text with numbers 123 and symbols #$@!\n";

        String expectedOutput = "digraph G {\n" +
                "\ttext [style=filled, fillcolor=lightgray];\n" +
                "\twith -> numbers [label=\"1\"];\n" +
                "\tand -> symbols [label=\"1\"];\n" +
                "\tnumbers -> and [label=\"1\"];\n" +
                "\ttext -> with [label=\"1\"];\n" +
                    "}\n";

        // Create a temporary file for testing
        File tempFile = createTempFile(input);

        try {
            TextToGraph graphBuilder = new TextToGraph();
            String[] words = graphBuilder.createDirectedGraph(graphBuilder, tempFile.getAbsolutePath(), "test.dot", "test.png");

            // Read the generated DOT file
            String actualOutput = readFile("test.dot");

            // Assert the expected output matches the actual output
            assertEquals(expectedOutput, actualOutput);
            System.out.println("测试用例 3 已通过");
        } finally {
            // Clean up: delete temporary file
            if (tempFile.exists()) {
                tempFile.delete();
            }
        }
    }


    // Helper method to create a temporary file with given content
    private File createTempFile(String content) throws IOException {
        File tempFile = File.createTempFile("temp", ".txt");
        tempFile.deleteOnExit();

        try (BufferedWriter writer = new BufferedWriter(new FileWriter(tempFile))) {
            writer.write(content);
        }

        return tempFile;
    }

    // Helper method to read content from a file
    private String readFile(String filePath) throws IOException {
        StringBuilder content = new StringBuilder();
        try (BufferedReader reader = new BufferedReader(new FileRe
ader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                content.append(line).append("\n");
            }
        }
        return content.toString();
    }
}
```

4.

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

public class TextToGraphTest {

    @Test
    public void testCreateDirectedGraph_case4() throws IOException
 {
        String input = "Continuous text without any newlines just
spaces.\n";

        String expectedOutput = "digraph G {\n" +
                "\tcontinuous [style=filled, fillcolor=lightgray];
\n" +
                "\tnewlines -> just [label=\"1\"];\n" +
                "\tcontinuous -> text [label=\"1\"];\n" +
                "\ttext -> without [label=\"1\"];\n" +
                "\tany -> newlines [label=\"1\"];\n" +
                "\tjust -> spaces [label=\"1\"];\n" +
                "\twithout -> any [label=\"1\"];\n" +
                "}\n";


        // Create a temporary file for testing
        File tempFile = createTempFile(input);

        try {
            TextToGraph graphBuilder = new TextToGraph();
            String[] words = graphBuilder.createDirectedGraph(grap
hBuilder, tempFile.getAbsolutePath(), "test.dot", "test.png");

            // Read the generated DOT file
            String actualOutput = readFile("test.dot");

```

```java
            // Assert the expected output matches the actual output
            assertEquals(expectedOutput, actualOutput);
            System.out.println("测试用例 4 已通过");
        } finally {
            // Clean up: delete temporary file
            if (tempFile.exists()) {
                tempFile.delete();
            }
        }
    }


    // Helper method to create a temporary file with given content
    private File createTempFile(String content) throws IOException {
        File tempFile = File.createTempFile("temp", ".txt");
        tempFile.deleteOnExit();

        try (BufferedWriter writer = new BufferedWriter(new FileWriter(tempFile))) {
            writer.write(content);
        }

        return tempFile;
    }

    // Helper method to read content from a file
    private String readFile(String filePath) throws IOException {
        StringBuilder content = new StringBuilder();
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                content.append(line).append("\n");
            }
        }
        return content.toString();
    }
}
```

5.

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

```

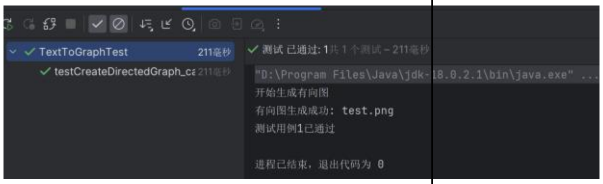
```java
public class TextToGraphTest {


    @Test
    public void testCreateDirectedGraph_case5() throws IOException {
        String input = "No @ punctuation here just words\n";

        String expectedOutput = "digraph G {\n" +
                "\tno [style=filled, fillcolor=lightgray];\n" +
                "\there -> just [label=\"1\"];\n" +
                "\tno -> punctuation [label=\"1\"];\n" +
                "\tpunctuation -> here [label=\"1\"];\n" +
                "\tjust -> words [label=\"1\"];\n" +
                "}\n";


        // Create a temporary file for testing
        File tempFile = createTempFile(input);

        try {
            TextToGraph graphBuilder = new TextToGraph();
            String[] words = graphBuilder.createDirectedGraph(graphBuilder, tempFile.getAbsolutePath(), "test.dot", "test.png");

            // Read the generated DOT file
            String actualOutput = readFile("test.dot");

            // Assert the expected output matches the actual output
            assertEquals(expectedOutput, actualOutput);
            System.out.println("测试用例 5 已通过");
        } finally {
            // Clean up: delete temporary file
            if (tempFile.exists()) {
                tempFile.delete();
            }
        }
    }


    // Helper method to create a temporary file with given content
    private File createTempFile(String content) throws IOException {
        File tempFile = File.createTempFile("temp", ".txt");
        tempFile.deleteOnExit();

        try (BufferedWriter writer = new BufferedWriter(new FileWriter(tempFile))) {
            writer.write(content);
        }

        return tempFile;
    }

    // Helper method to read content from a file
    private String readFile(String filePath) throws IOException {
        StringBuilder content = new StringBuilder();
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                content.append(line).append("\n");
            }
        }
        return content.toString();
    }
}
```

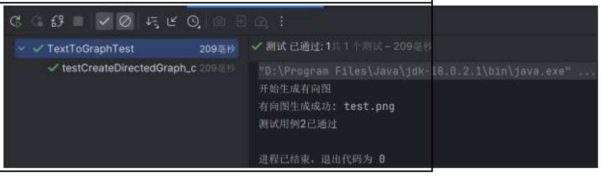
6.

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

public class TextToGraphTest {


    @Test
    public void testCreateDirectedGraph_case6() throws IOException {
        String input = "Just letters no numbers or symbols\n";

        String expectedOutput = "digraph G {\n" +
                "\tjust [style=filled, fillcolor=lightgray];\n" +
                "\tno -> numbers [label=\"1\"];\n" +
                "\tor -> symbols [label=\"1\"];\n" +
                "\tnumbers -> or [label=\"1\"];\n" +
                "\tjust -> letters [label=\"1\"];\n" +
                "\tletters -> no [label=\"1\"];\n" +
                "}\n";


        // Create a temporary file for testing
        File tempFile = createTempFile(input);

        try {
            TextToGraph graphBuilder = new TextToGraph();
            String[] words = graphBuilder.createDirectedGraph(graphBuilder, tempFile.getAbsolutePath(), "test.dot", "test.png");

            // Read the generated DOT file
            String actualOutput = readFile("test.dot");

            // Assert the expected output matches the actual output
            assertEquals(expectedOutput, actualOutput);
            System.out.println("测试用例 6 已通过");
        } finally {
            // Clean up: delete temporary file
            if (tempFile.exists()) {
                tempFile.delete();
            }
        }
    }


    // Helper method to create a temporary file with given content
    private File createTempFile(String content) throws IOException {
        File tempFile = File.createTempFile("temp", ".txt");
        tempFile.deleteOnExit();

        try (BufferedWriter writer = new BufferedWriter(new FileWriter(tempFile))) {
            writer.write(content);
        }

        return tempFile;
    }

    // Helper method to read content from a file
    private String readFile(String filePath) throws IOException {
        StringBuilder content = new StringBuilder();
        try (BufferedReader reader = new BufferedReader(new FileRe
ader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                content.append(line).append("\n");
            }
        }
        return content.toString();
    }
}
```

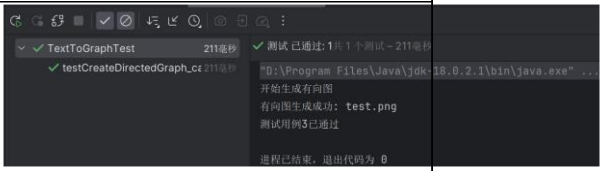
5.5 JUnit 单元测试结果

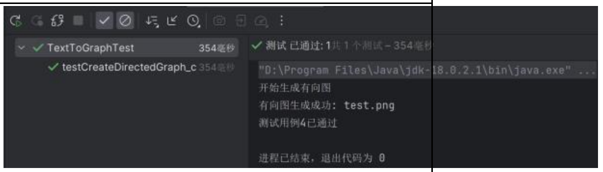
测试用例编号	期望输出	实际输出	是否通过测试，请给出屏幕截图

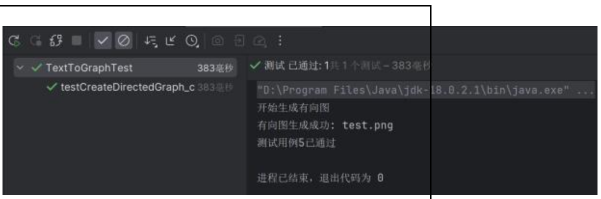
1.	digraph G {
	this [style=filled, fillcolor=lightgray];
	the -> third [label="1"];
	the -> first [label="1"];
	the -> second [label="1"];
	sentence -> and [label="1"];
	sentence -> this [label="1"];
	third -> sentence [label="1"];
	and -> this [label="1"];
	this -> is [label="3"];
	is -> the [label="3"];
	first -> sentence [label="1"];
	second -> sentence [label="1"];
	}
		digraph G {
		this [style=filled, fillcolor=lightgray];
		the -> third [label="1"];
		the -> first [label="1"];
		the -> second [label="1"];
		sentence -> and [label="1"];
		sentence -> this [label="1"];
		third -> sentence [label="1"];
		and -> this [label="1"];
		this -> is [label="3"];
		is -> the [label="3"];
		first -> sentence [label="1"];
		second -> sentence [label="1"];
		}
			 ✓ TextToGraphTest 211毫秒
			✓ testCreateDirectedGraph_ca 211毫秒
			✓ 测试 已通过: 1共 1 个测试 – 211毫秒
			"D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ...
			开始生成有向图
			有向图生成成功: test.png
			测试用例1已通过
			进程已结束，退出代码为 0

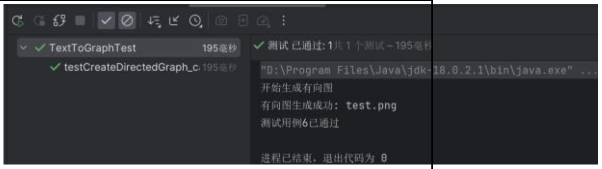
2.	digraph G {
	hello [style=filled, fillcolor=lightgray];
	world -> this
		digraph G {
		hello [style=filled, fillcolor=lightgray];
		world -> this
			 ✓ TextToGraphTest 209毫秒
			✓ testCreateDirectedGraph_c 209毫秒
			✓ 测试 已通过: 1共 1 个测试 – 209毫秒
			"D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ...
			开始生成有向图
			有向图生成成功: test.png
			测试用例2已通过
			进程已结束，退出代码为 0

	[label="1"];
	work -> correctly [label="1"];
	this -> is [label="1"];
	should -> work [label="1"];
	is -> an [label="1"];
	hello -> world [label="1"];
	an -> example [label="1"];
	example -> should [label="1"];
	}
		[label="1"];
		work -> correctly [label="1"];
		this -> is [label="1"];
		should -> work [label="1"];
		is -> an [label="1"];
		hello -> world [label="1"];
		an -> example [label="1"];
		example -> should [label="1"];
		}	

3.	digraph G {
	text [style=filled, fillcolor=lightgray];
	with -> numbers [label="1"];
	and -> symbols [label="1"];
	numbers -> and [label="1"];
	text -> with [label="1"];
	}
		digraph G {
		text [style=filled, fillcolor=lightgray];
		with -> numbers [label="1"];
		and -> symbols [label="1"];
		numbers -> and [label="1"];
		text -> with [label="1"];
		}
			
			TextToGraphTest 211毫秒
			testCreateDirectedGraph_ca 211毫秒
			测试 已通过: 1共 1 个测试 – 211毫秒
			"D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ...
			开始生成有向图
			有向图生成成功: test.png
			测试用例3已通过
			
			进程已结束，退出代码为 0

4.	digraph G {
	continuous [style=filled, fillcolor=lightgray];
	newlines -> just [label="1"];
	continuous -> text [label="1"];
	text -> without [label="1"];
	any -> newlines [label="1"];
	just -> spaces [label="1"];
	without -> any [label="1"];
	}
		digraph G {
		continuous [style=filled, fillcolor=lightgray];
		newlines -> just [label="1"];
		continuous -> text [label="1"];
		text -> without [label="1"];
		any -> newlines [label="1"];
		just -> spaces [label="1"];
		without -> any [label="1"];
		}
			
			TextToGraphTest 354毫秒
			testCreateDirectedGraph_c 354毫秒
			测试 已通过: 1共 1 个测试 – 354毫秒
			"D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ...
			开始生成有向图
			有向图生成成功: test.png
			测试用例4已通过
			
			进程已结束，退出代码为 0

5.	digraph G {
	no [style=filled, fillcolor=lightgray];
	here -> just [label="1"];
	no -> punctuation [label="1"];
	punctuation -> here [label="1"];
	just -> words [label="1"];
	}
		digraph G {
		no [style=filled, fillcolor=lightgray];
		here -> just [label="1"];
		no -> punctuation [label="1"];
		punctuation -> here [label="1"];
		just -> words [label="1"];
		}
			
			TextToGraphTest 383毫秒
			testCreateDirectedGraph_c 383毫秒
			测试 已通过: 1共 1 个测试 – 383毫秒
			"D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ...
			开始生成有向图
			有向图生成成功: test.png
			测试用例5已通过
			进程已结束，退出代码为 0

6.	digraph G {
	just [style=filled, fillcolor=lightgray];
	no -> numbers [label="1"];
	or -> symbols [label="1"];
	numbers -> or [label="1"];
	just -> letters [label="1"];
	letters -> no [label="1"];
	}
		digraph G {
		just [style=filled, fillcolor=lightgray];
		no -> numbers [label="1"];
		or -> symbols [label="1"];
		numbers -> or [label="1"];
		just -> letters [label="1"];
		letters -> no [label="1"];
		}
			
			TextToGraphTest 195毫秒
			testCreateDirectedGraph_c 195毫秒
			测试 已通过: 1共 1 个测试 – 195毫秒
			"D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ...
			开始生成有向图
			有向图生成成功: test.png
			测试用例6已通过
			进程已结束，退出代码为 0

5.6 未通过测试的原因分析及代码修改

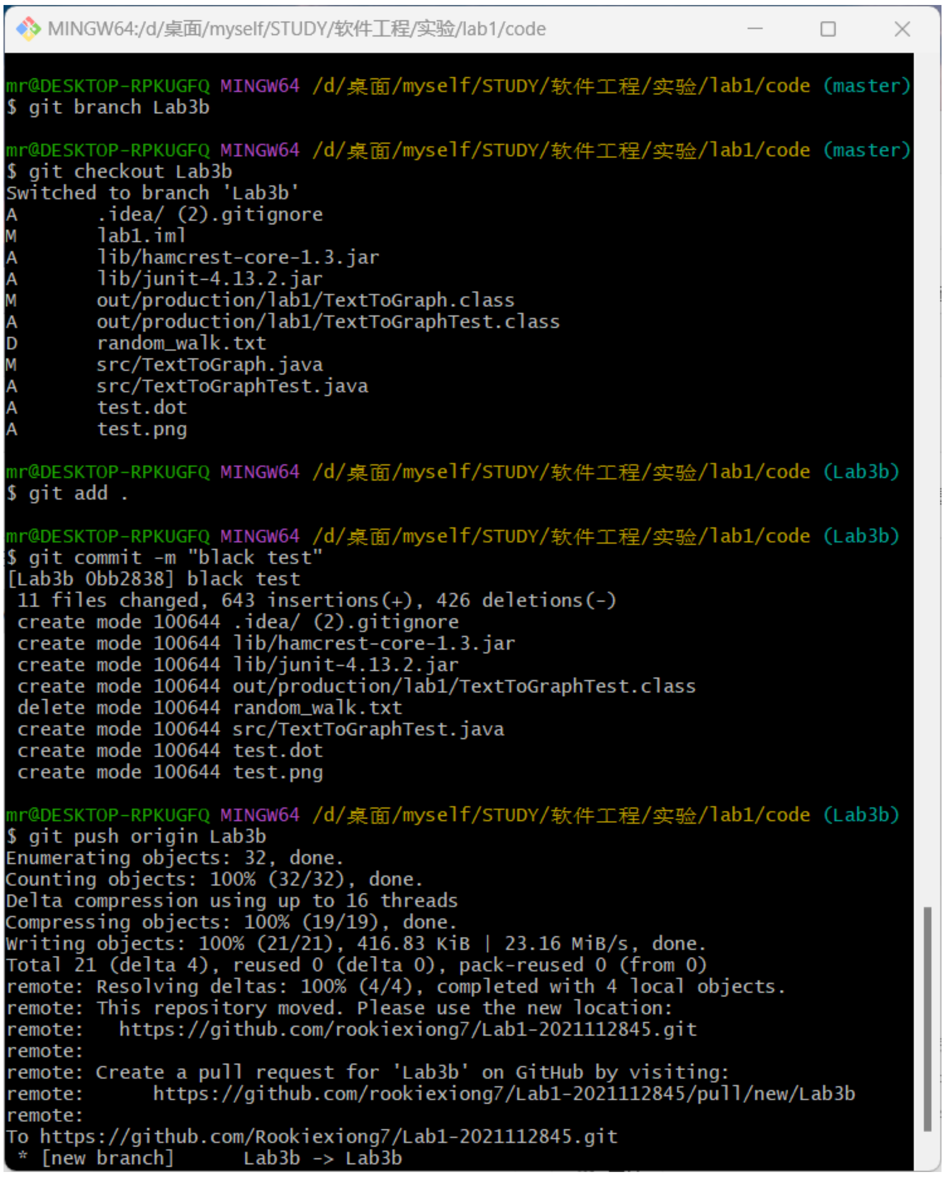
请简要分析自己的 Lab1 代码为何未通过 5.5 节表格中某些测试用例的原因，并通过修改代码消除此类不符合需求的 BUG。必要时给出修改后的代码。

若 5.5 节表格中没有未通过的测试用例，本节可空。

测试用例编号	期望输出字符串	实际输出字符串	是否通过测试，请给出屏幕截图
1.			

5.7 Git 操作记录

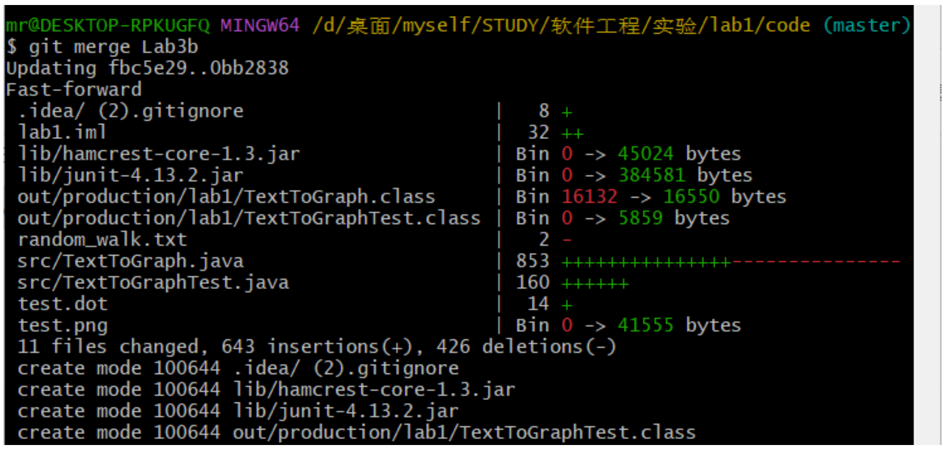
给出本地创建 Lab3b 分支，以及推送到 Github 上操作命令的截图；



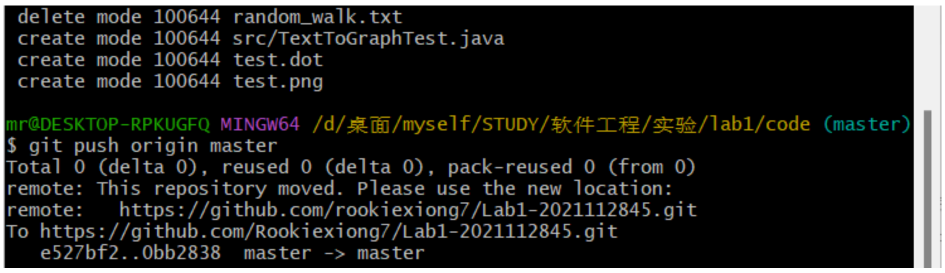
MINGW64:/d/桌面/myself/STUDY/软件工程/实验/lab1/code

```
mr@DESKTOP-RPKUGFQ MINGW64 /d/桌面/myself/STUDY/软件工程/实验/lab1/code (master)
$ git branch Lab3b

mr@DESKTOP-RPKUGFQ MINGW64 /d/桌面/myself/STUDY/软件工程/实验/lab1/code (master)
$ git checkout Lab3b
Switched to branch 'Lab3b'
A       .idea/ (2).gitignore
M       lab1.iml
A       lib/hamcrest-core-1.3.jar
A       lib/junit-4.13.2.jar
M       out/production/lab1/TextToGraph.class
A       out/production/lab1/TextToGraphTest.class
D       random_walk.txt
M       src/TextToGraph.java
A       src/TextToGraphTest.java
A       test.dot
A       test.png

mr@DESKTOP-RPKUGFQ MINGW64 /d/桌面/myself/STUDY/软件工程/实验/lab1/code (Lab3b)
$ git add .

mr@DESKTOP-RPKUGFQ MINGW64 /d/桌面/myself/STUDY/软件工程/实验/lab1/code (Lab3b)
$ git commit -m "black test"
[Lab3b 0bb2838] black test
 11 files changed, 643 insertions(+), 426 deletions(-)
 create mode 100644 .idea/ (2).gitignore
 create mode 100644 lib/hamcrest-core-1.3.jar
 create mode 100644 lib/junit-4.13.2.jar
 create mode 100644 out/production/lab1/TextToGraphTest.class
 delete mode 100644 random_walk.txt
 create mode 100644 src/TextToGraphTest.java
 create mode 100644 test.dot
 create mode 100644 test.png

mr@DESKTOP-RPKUGFQ MINGW64 /d/桌面/myself/STUDY/软件工程/实验/lab1/code (Lab3b)
$ git push origin Lab3b
Enumerating objects: 32, done.
Counting objects: 100% (32/32), done.
Delta compression using up to 16 threads
Compressing objects: 100% (19/19), done.
Writing objects: 100% (21/21), 416.83 KiB | 23.16 MiB/s, done.
Total 21 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
remote: This repository moved. Please use the new location:
remote:   https://github.com/rookiexiong7/Lab1-2021112845.git
remote:
remote: Create a pull request for 'Lab3b' on GitHub by visiting:
remote:      https://github.com/rookiexiong7/Lab1-2021112845/pull/new/Lab3b
remote:
To https://github.com/Rookiexiong7/Lab1-2021112845.git
 * [new branch]      Lab3b -> Lab3b
```

给出本地合并 Lab3b 分支到 master 分支，以及推送到 Github 上操作命令的截图。



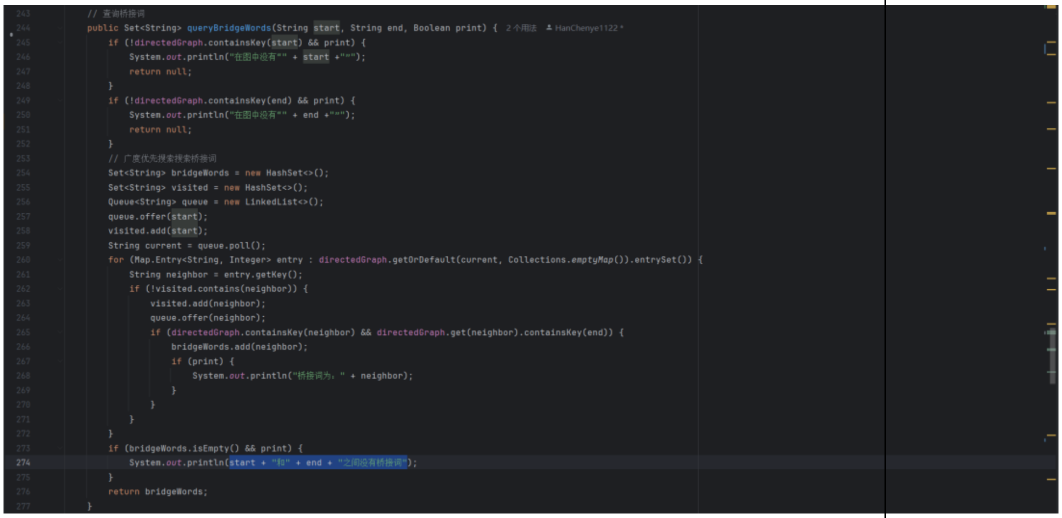
```
mr@DESKTOP-RPKUGFQ MINGW64 /d/桌面/myself/STUDY/软件工程/实验/lab1/code (master)
$ git merge Lab3b
Updating fbc5e29..0bb2838
Fast-forward
 .idea/ (2).gitignore                       |   8 +
 lab1.iml                                   |  32 ++
 lib/hamcrest-core-1.3.jar                  | Bin 0 -> 45024 bytes
 lib/junit-4.13.2.jar                       | Bin 0 -> 384581 bytes
 out/production/lab1/TextToGraph.class      | Bin 16132 -> 16550 bytes
 out/production/lab1/TextToGraphTest.class  | Bin 0 -> 5859 bytes
 random_walk.txt                            |   2 -
 src/TextToGraph.java                       | 853 +++++++++++++++---------------
 src/TextToGraphTest.java                   | 160 ++++++
 test.dot                                   |  14 +
 test.png                                   | Bin 0 -> 41555 bytes
 11 files changed, 643 insertions(+), 426 deletions(-)
 create mode 100644 .idea/ (2).gitignore
 create mode 100644 lib/hamcrest-core-1.3.jar
 create mode 100644 lib/junit-4.13.2.jar
 create mode 100644 out/production/lab1/TextToGraphTest.class
```



```
 delete mode 100644 random_walk.txt
 create mode 100644 src/TextToGraphTest.java
 create mode 100644 test.dot
 create mode 100644 test.png

mr@DESKTOP-RPKUGFQ MINGW64 /d/桌面/myself/STUDY/软件工程/实验/lab1/code (master)
$ git push origin master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: This repository moved. Please use the new location:
remote:   https://github.com/rookiexiong7/Lab1-2021112845.git
To https://github.com/Rookiexiong7/Lab1-2021112845.git
   e527bf2..0bb2838  master -> master
```

6　针对 Lab1 的白盒测试

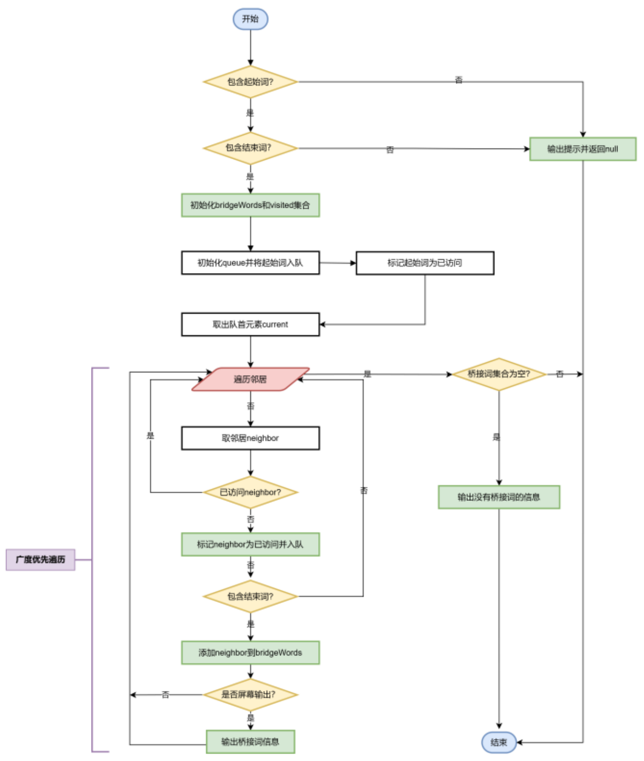
6.1 所选的被测函数

被测函数的名称	**queryBridgeWords**
功能描述	在生成有向图之后，用户输入任意两个英文单词 start、end，程序从图中查询它们的“桥接词”。start、end 的桥接词 word 满足图中存在两条边 start→word, word→end。 ➢ 输入的 start 或 end 如果不在图中出现，则输出提示“在图中没有"start"”或者“在图中没有"end"”。 ➢ 如果不存在桥接词，则输出“start 和 end 之间没有桥接词”。 ➢ 如果存在一个或多个桥接词，则依次输出“桥接词为：xxx”。
被测函数的代码	

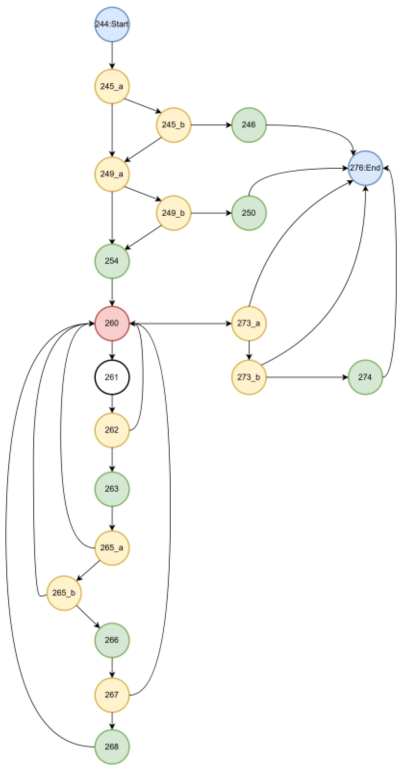
```java
    // 查询桥接词
    public Set<String> queryBridgeWords(String start, String end, Boolean print) {  2 个用法  HanChenye1122 *
        if (!directedGraph.containsKey(start) && print) {
            System.out.println("在图中没有“" + start +"”");
            return null;
        }
        if (!directedGraph.containsKey(end) && print) {
            System.out.println("在图中没有“" + end +"”");
            return null;
        }
        // 广度优先搜索搜索桥接词
        Set<String> bridgeWords = new HashSet<>();
        Set<String> visited = new HashSet<>();
        Queue<String> queue = new LinkedList<>();
        queue.offer(start);
        visited.add(start);
        String current = queue.poll();
        for (Map.Entry<String, Integer> entry : directedGraph.getOrDefault(current, Collections.emptyMap()).entrySet()) {
            String neighbor = entry.getKey();
            if (!visited.contains(neighbor)) {
                visited.add(neighbor);
                queue.offer(neighbor);
                if (directedGraph.containsKey(neighbor) && directedGraph.get(neighbor).containsKey(end)) {
                    bridgeWords.add(neighbor);
                    if (print) {
                        System.out.println("桥接词为：" + neighbor);
                    }
                }
            }
        }
        if (bridgeWords.isEmpty() && print) {
            System.out.println(start + "和" + end + "之间没有桥接词");
        }
        return bridgeWords;
    }
```

输入参数列表	参数名	含义	数据类型
	Start	第一个单词	字符串 String
	End	第二个单词	字符串 String
	print	是否屏幕输出桥接词	布尔变量 Boolean
输出参数	含义		数据类型
	若单词在图中不存在则为 null； 否则为查询到的桥接词集合（可以为空）；		字符串列表 Set<String>
代码总行数	34		
包含的循环数	1		
包含的判定数	6		

6.2 程序流程图



6.3 控制流图



6.4 圈复杂度计算与基本路径识别

圈复杂度为：

1. 流图 G 的圈复杂度 V(G)，定义为 V(G)=E-N+2，E 是流图中边的数量，N 是流图中结点的数量。流图的边的数量 E 为 30，结点的数量 N 为 21。所以 V(G)=30-21+2=11；
2. 流图 G 的圈复杂度 V(G)，定义为 V(G)=P+1，P 是流图 G 中判定结点的数量，流图中的判定结点的个数 P 为 10。所以 V(G)=10+1=11。

基本路径1： 244→245_a→245_b→246→276
基本路径2： 244→245_a→245_b→249_a→249_b→250→276
基本路径3： 244→245_a→249_a→249_b→250→276
基本路径4： 244→245_a→249_a→249_b→254→260→273_a→276
基本路径5： 244→245_a→249_a→254→260→261→262→260→273_a→273_b→276
基本路径6： 244→245_a→249_a→254→260→261→262→263→265_a→260→273_a→273_b→274→276
基本路径7： 244→245_a→249_a→254→260→261→262→263→265_a→265_b→260→273_a→273_b→274→276
基本路径8： 244→245_a→249_a→254→260→261→262→263→265_a→265_b→266→267→260→273_a→276
基本路径9： 244→245_a→249_a→254→260→261→262→263→265_a→265_b→266→267→268→260→273_a→273_b→274→276
基本路径10： 244→245_a→249_a→254→260→261→262→260→273_a→273_b→274→276
基本路径11： 244→245_a→249_a→254→260→261→262→263→265_a→265_b→266→267→268→260→273_a→276

6.5 测试用例设计

测试用例编号	输入数据	期望的输出	所覆盖的基本路径编号

1.	Start = aaa
	End = to
	print = true
		Null	1

2.	Start = aaa
	End = bbb
	print = false
		Null	2

3.	Start =i
	End = bbb
	print = true
		Null	3

4.	Start = i
	End = bbb
	print = false
		Null	4

5.	Start = i
	End = i
	print = false
		空列表[]	5

6.	Start =i
	End =to
	print = true
		Null	6

7.	Start =i
	End =play
	print = true
		空列表[]	7

8.	Start = i
	End = to
	print = false
		[‘like’]	8

9.	Start = i
	End = to
	print = true
		[‘like’]	9

10.	Start = i
	End = i
	print = true
		空列表[]	10

11.	Start = i
	End = to
	print = true
		[‘like’]	11

6.6 JUnit 测试代码

针对 6.5 中的每一个用例，把其测试代码粘贴如下，代码必须是完整的。

测试用例编号	jUnit 测试代码

1.	`@Test`
	`void test1() {`
	`    // word1 不在图中—路径 1`
	`    Set<String> bridgeWords = graphBuilder.queryBridgeWords("aaa", "to", true);`
	`    assertNull(bridgeWords);`
	`    System.out.println("测试用例 1 已通过");`
	`}`

2.	`@Test`
	`void test2() {`
	`    // word1 不存在，word2 不存在——路径 2`
	`    Set<String> bridgeWords = graphBuilder.queryBridgeWords("aaa", "bbb", true);`
	`    assertNull(bridgeWords);`
	`    System.out.println("测试用例 2 已通过");`
	`}`

3.

```java
    @Test
    void test3() {
        // word1 存在，word2 不在图中——路径 3
        Set<String> bridgeWords = graphBuilder.queryBridgeWords("i", "bbb", true);
        assertNull(bridgeWords);
        System.out.println("测试用例 3 已通过");
    }
```

4.

```java
    @Test
    void test4() {
            // word1 存在和 word2 不存在，print 为 false——路径 4
        Set<String> bridgeWords = graphBuilder.queryBridgeWords("i", "bbb", false);
        assertNotNull(bridgeWords);
        assertTrue(bridgeWords.isEmpty());
        System.out.println("测试用例 4 已通过");
    }
```

5.

```java
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Set;

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

class WhiteTest {
    private TextToGraph graphBuilder;

    @BeforeEach
    void setUp() {
        graphBuilder = new TextToGraph();

        // 手动添加一些节点和边以创建测试用的有向图
        graphBuilder.addNode("i");
        graphBuilder.addNode("i");
//          graphBuilder.addNode("like");
//          graphBuilder.addNode("to");
//          graphBuilder.addNode("play");
//          graphBuilder.addNode("games");

        graphBuilder.addEdge("i", "i");
//          graphBuilder.addEdge("like", "to");
//          graphBuilder.addEdge("to", "play");
//        graphBuilder.addEdge("play", "games");
    }

    @Test
    void test5() {
        // word1 和 word2 均存在，但是只有一个点—路径 5
        Set<String> bridgeWords = graphBuilder.queryBridgeWords("i", "i", true);
        Set<String> expected = new HashSet<>();
        assertEquals(expected, bridgeWords);
        System.out.println("测试用例 5 已通过");
    }
```

6.

```java
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Set;

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

class WhiteTest {
    private TextToGraph graphBuilder;

    @BeforeEach
    void setUp() {
        graphBuilder = new TextToGraph();

        // 手动添加一些节点和边以创建测试用的有向图
        graphBuilder.addNode("i");
        graphBuilder.addNode("like");
        graphBuilder.addNode("to");
        graphBuilder.addNode("play");
        graphBuilder.addNode("games");

        graphBuilder.addEdge("i", "now");
        graphBuilder.addEdge("like", "to");
        graphBuilder.addEdge("to", "play");
        graphBuilder.addEdge("play", "games");
    }
    @Test
    void test6() {
        // word1 的邻居不在图中，且 print 为 false， word1 无邻居，bridgeWords 为空—路径 6
        Set<String> bridgeWords = graphBuilder.queryBridgeWords("i", "to", true);
        assertNull(bridgeWords);
        System.out.println("测试用例 6 已通过");
    }
```

7.

```java
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Set;

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

class WhiteTest {
    private TextToGraph graphBuilder;

    @BeforeEach
    void setUp() {
        graphBuilder = new TextToGraph();

        // 手动添加一些节点和边以创建测试用的有向图
        graphBuilder.addNode("i");
        graphBuilder.addNode("like");
        graphBuilder.addNode("to");
        graphBuilder.addNode("play");
        graphBuilder.addNode("games");

        graphBuilder.addEdge("i", "like");
        graphBuilder.addEdge("like", "to");
        graphBuilder.addEdge("to", "play");
        graphBuilder.addEdge("play", "games");
    }
    @Test
    void test7() {
        // word1 的邻居的邻居不是 word2——路径 7
        Set<String> bridgeWords = graphBuilder.queryBridgeWords("i", "play", true);
        assertNull(bridgeWords);
        System.out.println("测试用例 7 已通过");

    }
```

8.

```java
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Set;

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

class WhiteTest {
    private TextToGraph graphBuilder;

    @BeforeEach
    void setUp() {
        graphBuilder = new TextToGraph();

        // 手动添加一些节点和边以创建测试用的有向图
        graphBuilder.addNode("i");
        graphBuilder.addNode("like");
        graphBuilder.addNode("to");
        graphBuilder.addNode("play");
        graphBuilder.addNode("games");

        graphBuilder.addEdge("i", "like");
        graphBuilder.addEdge("like", "to");
        graphBuilder.addEdge("to", "play");
        graphBuilder.addEdge("play", "games");
    }
    @Test
    void test8() {
        // word1 和 word2 存在桥接词但是 print 为 false——路径 9
        Set<String> bridgeWords = graphBuilder.queryBridgeWords("i", "to", false);
        assertNotNull(bridgeWords);
        assertEquals(1, bridgeWords.size());
        assertTrue(bridgeWords.contains("like"));
        System.out.println("测试用例 8 已通过");
    }
```

9.

```java
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Set;

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

```java

class WhiteTest {
    private TextToGraph graphBuilder;

    @BeforeEach
    void setUp() {
        graphBuilder = new TextToGraph();

        // 手动添加一些节点和边以创建测试用的有向图
        graphBuilder.addNode("i");
        graphBuilder.addNode("like");
        graphBuilder.addNode("to");
        graphBuilder.addNode("play");
        graphBuilder.addNode("games");

        graphBuilder.addEdge("i", "like");
        graphBuilder.addEdge("like", "to");
        graphBuilder.addEdge("to", "play");
        graphBuilder.addEdge("play", "games");
    }
    @Test
    void test9() {
        // word1 和 word2 存在桥接词但是 print 为 true——路径 9
        Set<String> bridgeWords = graphBuilder.queryBridgeWords("i", "to", true);
        assertNotNull(bridgeWords);
        assertEquals(1, bridgeWords.size());
        assertTrue(bridgeWords.contains("like"));
        System.out.println("测试用例 9 已通过");
    }
```

10.

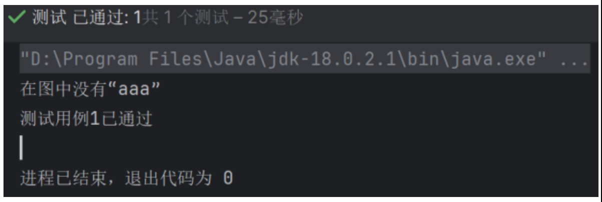
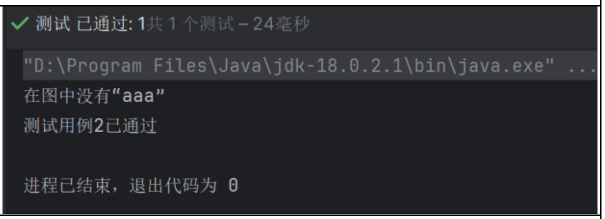
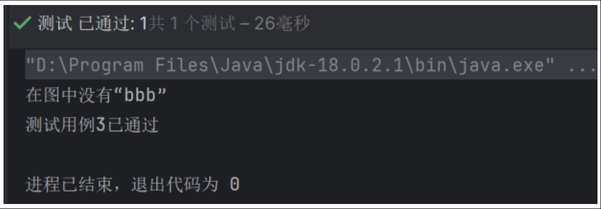
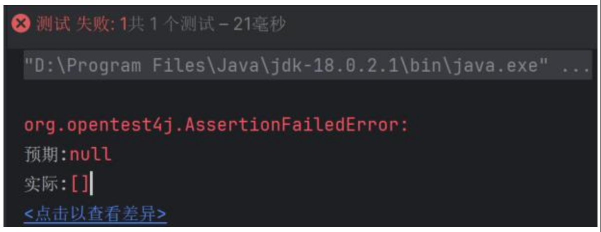
```java
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Set;

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

class WhiteTest {
    private TextToGraph graphBuilder;

    @BeforeEach
    void setUp() {
        graphBuilder = new TextToGraph();

        // 手动添加一些节点和边以创建测试用的有向图
        graphBuilder.addNode("i");
        graphBuilder.addNode("like");
        graphBuilder.addNode("to");
        graphBuilder.addNode("play");
        graphBuilder.addNode("games");

        graphBuilder.addEdge("i", "like");
        graphBuilder.addEdge("like", "to");
        graphBuilder.addEdge("to", "play");
        graphBuilder.addEdge("play", "games");
    }
    @Test
    void test10() {
        // word1 和 word2 均存在，但是只有一个点,print 为true——路径5
        Set<String> bridgeWords = graphBuilder.queryBridgeWords("i", "i", true);
        assertNull(bridgeWords);
        System.out.println("测试用例 10 已通过");
    }
```

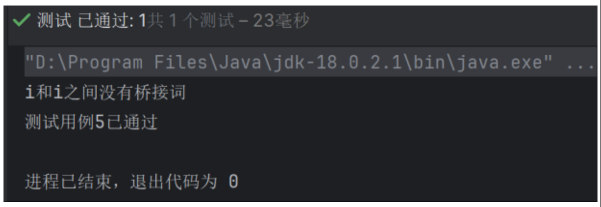
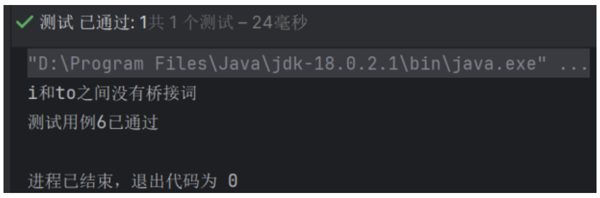
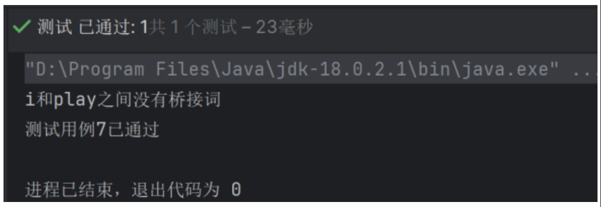
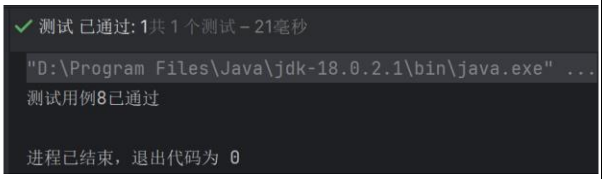
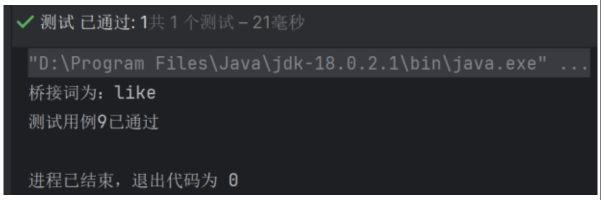
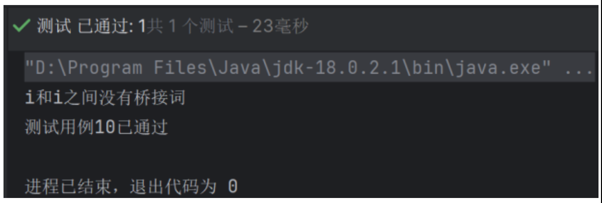
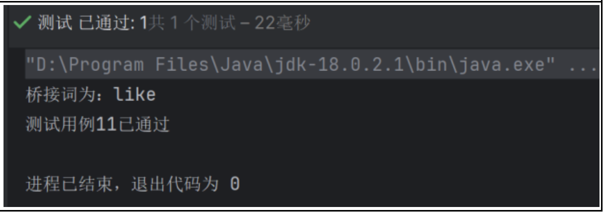
11.

```java
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Set;

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

class WhiteTest {
    private TextToGraph graphBuilder;

    @BeforeEach
    void setUp() {
        graphBuilder = new TextToGraph();

        // 手动添加一些节点和边以创建测试用的有向图
        graphBuilder.addNode("i");
        graphBuilder.addNode("like");
        graphBuilder.addNode("to");
        graphBuilder.addNode("play");
        graphBuilder.addNode("games");
```

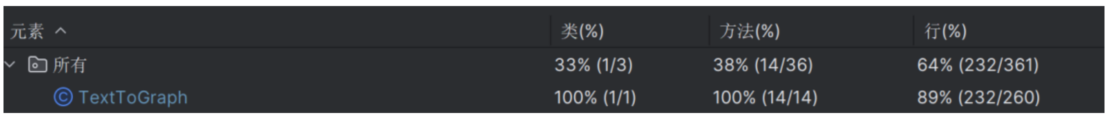
```java

            graphBuilder.addEdge("i", "like");
            graphBuilder.addEdge("like", "to");
            graphBuilder.addEdge("to", "play");
            graphBuilder.addEdge("play", "games");
        }
        @Test
        void test11() {
            // word1 和 word2 存在桥接词,bridgeWords.isEmpty()为false——路径11
            Set<String> bridgeWords = graphBuilder.queryBridgeWords("i", "to", true);
            assertNotNull(bridgeWords);
            assertEquals(1, bridgeWords.size());
            assertTrue(bridgeWords.contains("like"));
            System.out.println("测试用例 11 已通过");
        }
```

6.7 JUnit 单元测试结果

测试用例编号	期望输出	实际输出	是否通过测试，请给出屏幕截图
1.	Null	Null	 ✓ 测试 已通过: 1共 1 个测试 – 25毫秒 "D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ... 在图中没有“aaa” 测试用例1已通过 进程已结束，退出代码为 0
2.	Null	Null	 ✓ 测试 已通过: 1共 1 个测试 – 24毫秒 "D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ... 在图中没有“aaa” 测试用例2已通过 进程已结束，退出代码为 0
3.	Null	Null	 ✓ 测试 已通过: 1共 1 个测试 – 26毫秒 "D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ... 在图中没有“bbb” 测试用例3已通过 进程已结束，退出代码为 0
4.	Null	[]	 ✗ 测试 失败: 1共 1 个测试 – 21毫秒 "D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ... org.opentest4j.AssertionFailedError: 预期:null 实际:[] <点击以查看差异>

5.	[]	[]	 ✓ 测试 已通过: 1共 1 个测试 – 23毫秒 "D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ... i和i之间没有桥接词 测试用例5已通过 进程已结束，退出代码为 0
6.	Null	Null	 ✓ 测试 已通过: 1共 1 个测试 – 24毫秒 "D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ... i和to之间没有桥接词 测试用例6已通过 进程已结束，退出代码为 0
7.	[]	[]	 ✓ 测试 已通过: 1共 1 个测试 – 23毫秒 "D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ... i和play之间没有桥接词 测试用例7已通过 进程已结束，退出代码为 0
8.	[‘like’]	[‘like’]	 ✓ 测试 已通过: 1共 1 个测试 – 21毫秒 "D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ... 测试用例8已通过 进程已结束，退出代码为 0
9.	[‘like’]	[‘like’]	 ✓ 测试 已通过: 1共 1 个测试 – 21毫秒 "D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ... 桥接词为：like 测试用例9已通过 进程已结束，退出代码为 0
10.	[]	[]	 ✓ 测试 已通过: 1共 1 个测试 – 23毫秒 "D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ... i和i之间没有桥接词 测试用例10已通过 进程已结束，退出代码为 0
11.	[‘like’]	[‘like’]	 ✓ 测试 已通过: 1共 1 个测试 – 22毫秒 "D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ... 桥接词为：like 测试用例11已通过 进程已结束，退出代码为 0

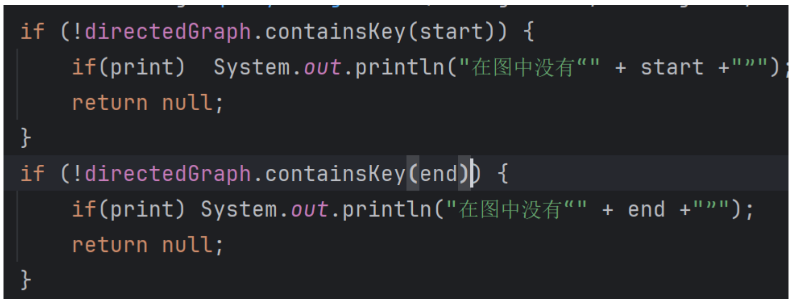
6.8 代码覆盖度分析



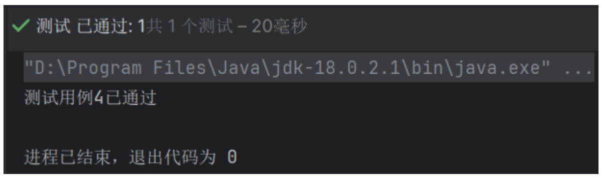
元素	类(%)	方法(%)	行(%)
所有	33% (1/3)	38% (14/36)	64% (232/361)
TextToGraph	100% (1/1)	100% (14/14)	89% (232/260)

6.9 未通过测试的原因分析及代码修改

未通过测试用例 4 是因为当 print 设置为 false 时，尽管不存在两个词中的某一个也会跳出判断条件，从而直接进入下面的桥接词查询环节。只需要将外部的对 print 的判断移入到内部即可，修改后代码如下：

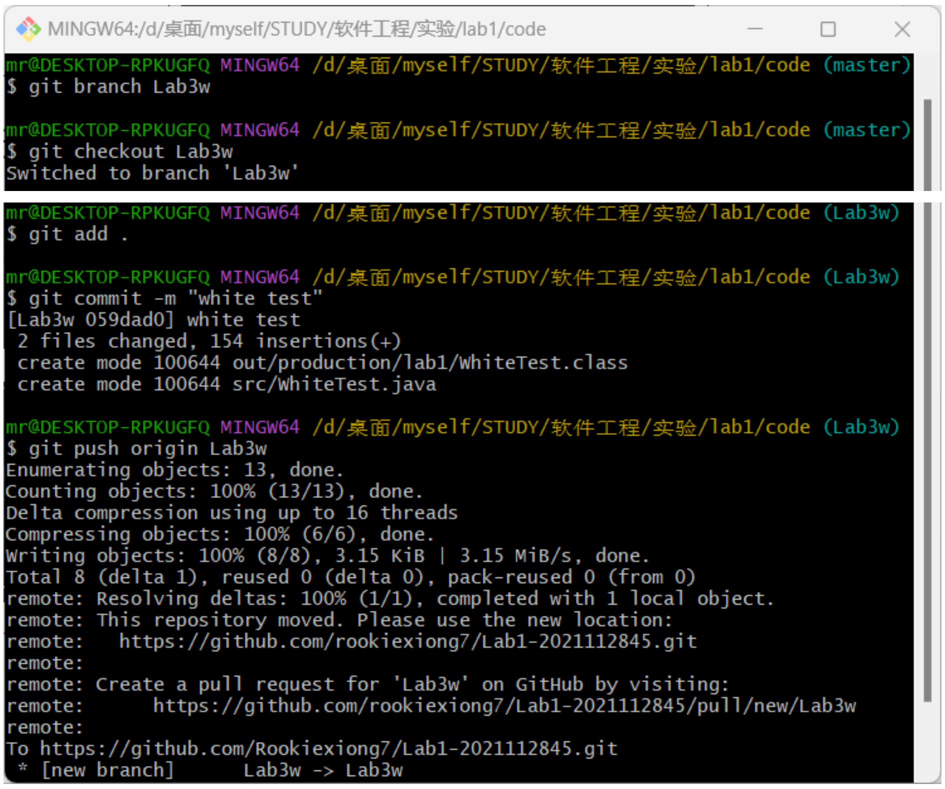


```
if (!directedGraph.containsKey(start)) {
    if(print)  System.out.println("在图中没有“" + start +"”");
    return null;
}
if (!directedGraph.containsKey(end)) {
    if(print) System.out.println("在图中没有“" + end +"”");
    return null;
}
```

测试用例编号	期望输出	实际输出	是否通过测试，请给出屏幕截图
4.	Null	Null	 ✓ 测试 已通过: 1共 1 个测试 – 20毫秒 "D:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ... 测试用例4已通过 进程已结束，退出代码为 0

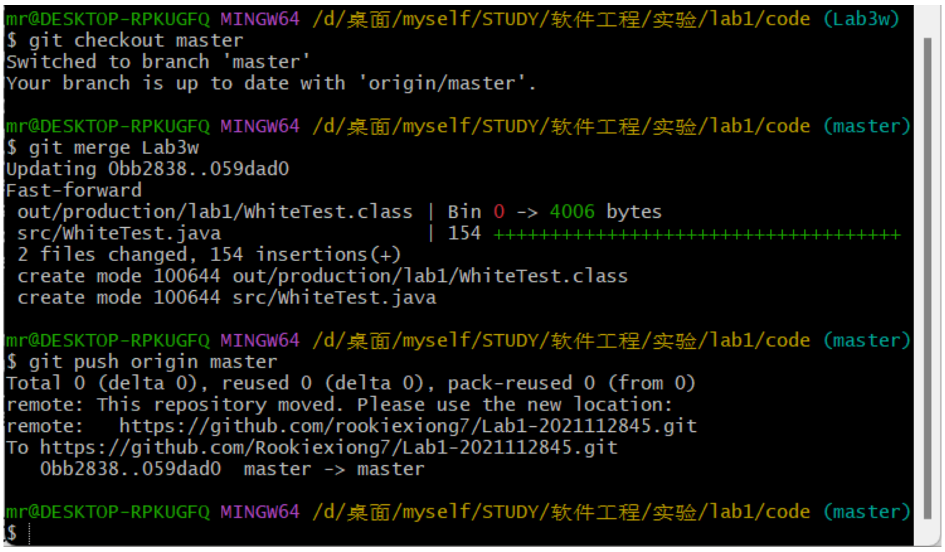
6.10 Git 操作记录

给出本地创建 Lab3w 分支，以及推送到 Github 上操作命令的截图；



```
mr@DESKTOP-RPKUGFQ MINGW64 /d/桌面/myself/STUDY/软件工程/实验/lab1/code (master)
$ git branch Lab3w

mr@DESKTOP-RPKUGFQ MINGW64 /d/桌面/myself/STUDY/软件工程/实验/lab1/code (master)
$ git checkout Lab3w
Switched to branch 'Lab3w'

mr@DESKTOP-RPKUGFQ MINGW64 /d/桌面/myself/STUDY/软件工程/实验/lab1/code (Lab3w)
$ git add .

mr@DESKTOP-RPKUGFQ MINGW64 /d/桌面/myself/STUDY/软件工程/实验/lab1/code (Lab3w)
$ git commit -m "white test"
[Lab3w 059dad0] white test
 2 files changed, 154 insertions(+)
 create mode 100644 out/production/lab1/WhiteTest.class
 create mode 100644 src/WhiteTest.java

mr@DESKTOP-RPKUGFQ MINGW64 /d/桌面/myself/STUDY/软件工程/实验/lab1/code (Lab3w)
$ git push origin Lab3w
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 3.15 KiB | 3.15 MiB/s, done.
Total 8 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote: This repository moved. Please use the new location:
remote:   https://github.com/rookiexiong7/Lab1-2021112845.git
remote:
remote: Create a pull request for 'Lab3w' on GitHub by visiting:
remote:      https://github.com/rookiexiong7/Lab1-2021112845/pull/new/Lab3w
remote:
To https://github.com/Rookiexiong7/Lab1-2021112845.git
 * [new branch]      Lab3w -> Lab3w
```

给出本地合并 Lab3w 分支到 master 分支，以及推送到 Github 上操作命令的截图。



```
mr@DESKTOP-RPKUGFQ MINGW64 /d/桌面/myself/STUDY/软件工程/实验/lab1/code (Lab3w)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

mr@DESKTOP-RPKUGFQ MINGW64 /d/桌面/myself/STUDY/软件工程/实验/lab1/code (master)
$ git merge Lab3w
Updating 0bb2838..059dad0
Fast-forward
 out/production/lab1/WhiteTest.class | Bin 0 -> 4006 bytes
 src/WhiteTest.java                  | 154 ++++++++++++++++++++++++++++++++++
 2 files changed, 154 insertions(+)
 create mode 100644 out/production/lab1/WhiteTest.class
 create mode 100644 src/WhiteTest.java

mr@DESKTOP-RPKUGFQ MINGW64 /d/桌面/myself/STUDY/软件工程/实验/lab1/code (master)
$ git push origin master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: This repository moved. Please use the new location:
remote:   https://github.com/rookiexiong7/Lab1-2021112845.git
To https://github.com/Rookiexiong7/Lab1-2021112845.git
   0bb2838..059dad0  master -> master

mr@DESKTOP-RPKUGFQ MINGW64 /d/桌面/myself/STUDY/软件工程/实验/lab1/code (master)
$
```

7 计划与实际进度

任务名称	计划时间长度(分钟)	实际耗费时间(分钟)	提前或延期的原因分析
配置 Checkstyle	30	40	安装和配置过程中遇到网络问题，导致时间延长。
配置 SpotBugs	20	25	部分规则集下载较慢，导致时间稍有延长。
配置 JUnit	20	15	配置过程顺利，所需时间较短。
Checkstyle 代码审查	40	50	代码中存在较多问题，分析和修改花费较多时间。
SpotBugs 代码审查	40	45	发现的部分问题较为复杂，解决时耗费了较多时间。
黑盒测试用例设计	60	70	设计过程中需要反复确认需求，导致时间延长。
黑盒测试代码编写	80	90	部分测试用例编写较为复杂，花费了更多时间。
白盒测试用例设计	60	65	设计过程中需要详细分析代码结构，导致时间延长。
白盒测试代码编写	80	85	测试过程中发现一些边界情况，处理花费较多时间。

8 小结

在本次实验中，我们对代码进行了全面的审查和测试，主要涉及 Checkstyle、SpotBugs、EclEmma 和 JUnit 四个工具的配置与使用。通过 Checkstyle 和 SpotBugs 工具，我们发现并修复了代码中的多个问题，提高了代码的规范性和正确性。在单元测试部分，我们设计了黑盒和白盒测试用例，编写并执行了相应的测试代码，并分析了测试的覆盖度。

实验过程中我们遇到了一些问题，但通过团队的合作和努力，都得到了有效解决。本次实验不仅提高了我们的代码质量，还增强了我们使用代码审查和测试工具的能力，为后续的开发工作打下了良好的基础