浙江大学 实验报告

专业：生物医学工程
姓名：倪旌哲
学号：3220100733
日期：2024.10.21

课程名称：高级程序设计　指导老师：耿晨歌　成绩：_____
实验名称：计算代码行数　实验类型：基础实验　同组学生姓名：无

一、实验目的和要求

写一个 Java 程序，统计一个项目中一共有多少行代码，多少行注释，多少个空白行.

通过命令行参数给定 目录

对目录下所有 java 文件，扫描其中有:

i. 多少行代码
ii. 多少行注释
iii. 多少空白行

二、实验内容和原理

i. 统计代码行数
    i. 如果我们将目光限制在一个文件上，要统计这个文件内的代码行，空行和注释行，只需要做简单的模式检查即可。
    ii. 空行很简单，如果 `line = ""`，那么这一行就是空行。
    iii. 注释行略微复杂，因为有单行注释和多行注释:
        i. 针对单行注释，我们只需要检查 `line.startsWith("//")`即可。
        ii. 针对多行注释，我们需要检查 `line.startsWith("/*")`和 `line.endsWith("*/")`，并且在多行注释中间的行都是注释行，可以用一个 `while` 循环来处理。
        iii. 针对代码行，我们可以通过排除空行和注释行来得到。
ii. Folder 递归搜索统计
    i. 要实现对一个目录下所有 java 文件的统计，我们需要递归地遍历这个目录，对每一个 java 文件进行统计。
    ii. 关键就是判断一个路径是否是一个 folder，我们可以通过 `Files.isDirectory()`来判断。
    iii. 对目录的递归处理可以通过 `Files.walk(path)`来实现，这个方法会返回一个 `Stream<Path>`，我们可以通过 `forEach()`来遍历这个 Stream。
iii. 命令行参数处理

i. 我们可以通过 `args[]`来获取命令行参数，这个参数是一个 `String[]`，我们可以通过 `args.length` 来获取参数的个数。

iv. 数据统计

i. 创建一个 `FileState` 类来统计文件/文件夹的代码行数，空行数和注释行数。

三、主要仪器设备

> 个人电脑

四、操作法方法与实现步骤

i. 创建 UniTest

详见第五部分

ii. 定义 `FileState`

```
public static class FileStats {
    public int codeLines = 0;
    public int commentLines = 0;
    public int blankLines = 0;

    public void add(FileStats other) {
        this.codeLines += other.codeLines;
        this.commentLines += other.commentLines;
        this.blankLines += other.blankLines;
    }

    @Override
    public String toString() {
        return String.format(
          "Code Lines: %d, Comment Lines: %d, Blank Lines: %d",
          codeLines, commentLines, blankLines
        );
    }
}
```

iii. 读取命令行参数

```
// Help menu
if (args.length != 1 || args[0].equals("-h") || args[0].equals("--help")) {
    System.out.println("Usage: java Cloc <file or directory path>");
    return;
}

// Get the path (dir or single file both ok)
Path path = Paths.get(args[0]);

// Make sure the file or directory exists
if (!Files.exists(path)) {
    System.out.println("File or directory does not exist.");
```

```
        return;
    }
```

iv. 根据 path 属性递归 travel 目录，并统计代码行数，最终打印出总计

```
    // If so, we will create a new FileStats object to store the total statistics
    FileStats totalStats = new FileStats();

    // Recursively process the directory or a single file
    try {
        if (Files.isDirectory(path)) {
            // 递归处理目录
            Files.walk(path).filter(Files::isRegularFile).forEach(file -> {

                // If this is a java code file
                if (!file.toString().endsWith(".java")) {
                    return;
                }

                try {
                    FileStats stats = processFile(file);
                    System.out.println(file + ": " + stats);
                    totalStats.add(stats);
                } catch (IOException e) {
                    System.out.println("Error reading file: " + file);
                }
            });
        } else {
            // 处理单个文件
            FileStats stats = processFile(path);
            System.out.println(path + ": " + stats);
            totalStats.add(stats);
        }

        // 输出总计
        System.out.println("\nTotal Stats: " + totalStats);

    } catch (IOException e) {
        e.printStackTrace();
    }
```

i. 其中， 对于 processFile 的实现如下:

```
    public static FileStats processFile(Path file) throws IOException {
        FileStats stats = new FileStats();
        boolean blockComment = false; // 标记块注释的状态

        List<String> lines = Files.readAllLines(file);
        for (String line : lines) {
            line = line.trim();

            if (line.isEmpty()) {
                stats.blankLines++;
            } else if (blockComment) {
                stats.commentLines++;
                if (line.endsWith("*/")) {
                    blockComment = false;
                }
            } else if (line.startsWith("//")) {
```

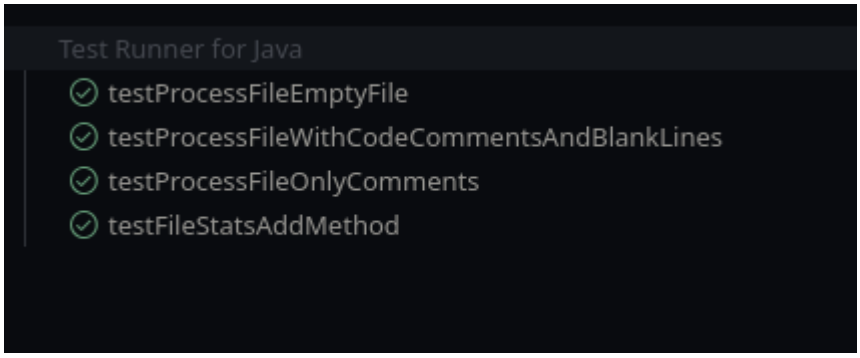
```
                stats.commentLines++;
            } else if (line.startsWith("/*")) {
                stats.commentLines++;
                if (!line.endsWith("*/")) {
                    blockComment = true;
                }
            } else {
                stats.codeLines++;
            }
        }

        return stats;
    }
```

五、实验结果与分析

i. 为了测试这个程序，我们在一开始规划了以下几种测试用例：
   i. 测试一个正常的文件，包含 5 行代码，3 行注释，1 个空行
   ii. 测试一个空文件
   iii. 测试一个只有注释的文件
   iv. 测试 `FileState` 的 `add()` 方法

ii. 没有文件夹用例的测试，因为以上几种情况能够完全组合出针对文件夹的测试用例

iii. 测试结果：



iv. 单元测试代码：

```
import org.junit.Test;

import java.nio.file.Files;
import java.nio.file.Path;
import java.io.IOException;

import static org.junit.Assert.*;

public class ClocTest {
```

```java
// Test case for processing files with code comments and blank lines
@Test
public void testProcessFileWithCodeCommentsAndBlankLines() throws IOException {
    // 准备测试文件内容
    // 5 lines of code
    // 3 lines of comments
    // 1 blank line
    String testContent = "public class Test {\n" +
            "    // This is a comment\n" +
            "    public void method() {\n" +
            "        System.out.println(\"Hello, world!\");\n" +
            "    }\n" +
            "    /* Block comment start\n" +
            "       Block comment end */\n" +
            "}\n" +
            "\n";

    // 创建临时文件
    Path testFile = Files.createTempFile("testFile", ".java");
    Files.write(testFile, testContent.getBytes());

    // 执行 processFile 方法
    Cloc.FileStats result = Cloc.processFile(testFile);

    // 断言代码行、注释行、空行的统计结果
    assertEquals(5, result.codeLines); // 3 行代码
    assertEquals(3, result.commentLines); // 3 行注释
    assertEquals(1, result.blankLines); // 1 行空行

    // 删除临时文件
    Files.delete(testFile);
}


// Test case for empty file
@Test
public void testProcessFileEmptyFile() throws IOException {
    // 创建空文件
    Path emptyFile = Files.createTempFile("emptyFile", ".java");

    // 执行 processFile 方法
    Cloc.FileStats result = Cloc.processFile(emptyFile);

    // 断言结果是 0 行
    assertEquals(0, result.codeLines);
    assertEquals(0, result.commentLines);
    assertEquals(0, result.blankLines);

    // 删除临时文件
    Files.delete(emptyFile);
}

// Test case for file contains only comments
@Test
public void testProcessFileOnlyComments() throws IOException {
    // 准备只有注释的文件内容
```

```
            String testContent = "// Single line comment\n" +
                    "/* Block comment start\n" +
                    "   Block comment end */\n" +
                    "\n";

            // 创建临时文件
            Path testFile = Files.createTempFile("testFileOnlyComments", ".java");
            Files.write(testFile, testContent.getBytes());

            // 执行 processFile 方法
            Cloc.FileStats result = Cloc.processFile(testFile);

            // 断言只有注释行
            assertEquals(0, result.codeLines);
            assertEquals(3, result.commentLines); // 3 行注释
            assertEquals(1, result.blankLines); // 1 行空行

            // 删除临时文件
            Files.delete(testFile);
        }

        // Test file for FileStats add method
        @Test
        public void testFileStatsAddMethod() {
            // 创建两个 FileStats 对象
            Cloc.FileStats stats1 = new Cloc.FileStats();
            stats1.codeLines = 5;
            stats1.commentLines = 3;
            stats1.blankLines = 2;

            Cloc.FileStats stats2 = new Cloc.FileStats();
            stats2.codeLines = 10;
            stats2.commentLines = 7;
            stats2.blankLines = 4;

            // 合并统计结果
            stats1.add(stats2);

            // 断言合并后的结果
            assertEquals(15, stats1.codeLines);
            assertEquals(10, stats1.commentLines);
            assertEquals(6, stats1.blankLines);
        }
    }
```

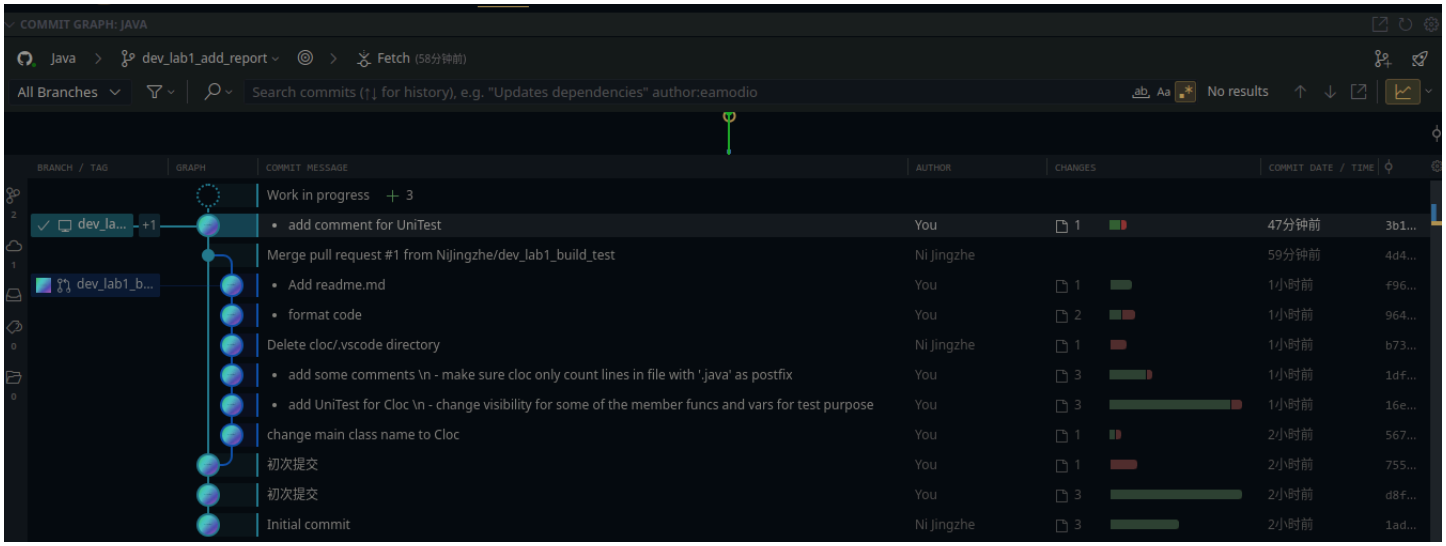
六、讨论、心得

i. 熟悉了敏捷开发的流程，通过 TDD 的方式编写代码，提高了代码的质量和可维护性

ii. 通过这个实验，我学会了如何处理命令行参数，如何递归地遍历一个目录，如何统计代码行数，空行数和注释行数

iii. 通过这个实验，我学会了如何编写单元测试，如何使用 JUnit 来进行单元测试

iv. 熟悉了使用 git 进行版本管理的流程，学会了如何使用 GitHub 来托管代码


COMMIT GRAPH: JAVA
Java
dev_lab1_add_report
Fetch (58分钟前)
All Branches
Search commits (↑↓ for history), e.g. "Updates dependencies" author:eamodio
No results
BRANCH / TAG
GRAPH
COMMIT MESSAGE
AUTHOR
CHANGES
COMMIT DATE / TIME
Work in progress + 3
dev_la... +1
• add comment for UniTest
You
1
47分钟前
3b1...
Merge pull request #1 from NiJingzhe/dev_lab1_build_test
Ni Jingzhe
59分钟前
4d4...
dev_lab1_b...
• Add readme.md
You
1
1小时前
f96...
• format code
You
2
1小时前
964...
Delete cloc/.vscode directory
Ni Jingzhe
1
1小时前
b73...
• add some comments \n - make sure cloc only count lines in file with '.java' as postfix
You
3
1小时前
1df...
• add UniTest for Cloc \n - change visibility for some of the member funcs and vars for test purpose
You
3
1小时前
16e...
change main class name to Cloc
You
1
2小时前
567...
初次提交
You
1
2小时前
755...
初次提交
You
3
2小时前
d8f...
Initial commit
Ni Jingzhe
3
2小时前
1ad...