

Lab 09 Assignment

班级: 202111

学号：19241027

姓名：胡峻诚

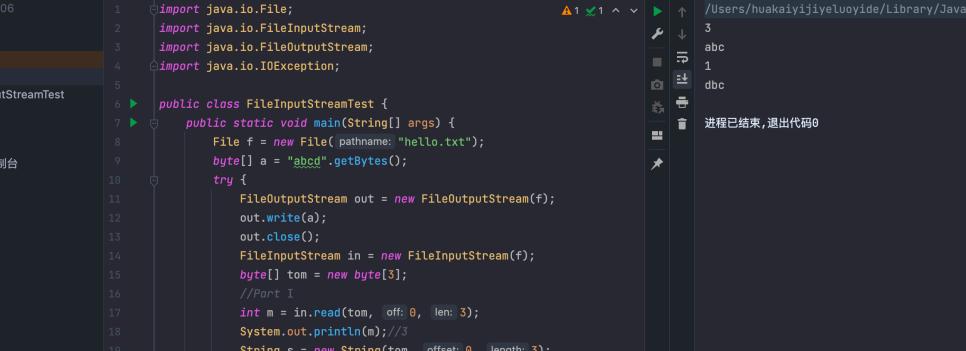
Question1 如果准备按字节读取一个文件的内容，应当使用 `FileInputStream` 流还是 `FileReader` 流，为什么？

应该选择 `FileInputStream`

- `FileInputStream` 流的 `read` 方法是按字节从文件中读取的
 - `FileReader` 的 `read` 是按字符读取的

Question2 简答

1. 请写出程序的输出



```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class FileInputStreamTest {
    public static void main(String[] args) {
        File f = new File("hello.txt");
        byte[] a = "abc".getBytes();
        try {
            FileOutputStream out = new FileOutputStream(f);
            out.write(a);
            out.close();
            FileInputStream in = new FileInputStream(f);
            byte[] tom = new byte[3];
            //Part I
            int m = in.read(tom, 0, 3);
            System.out.println(m); //3
            String s = new String(tom, 0, 3);
            System.out.println(s); //abc
            //Part II
            m = in.read(tom, 0, 3);
            System.out.println(m); //1
            s = new String(tom, 0, 3);
            System.out.println(s); //dbc
        } catch (IOException e) {
        }
    }
}
```

运行: FileInputStreamTest
/Users/huakaiyijielyuoyide/Library/Java/JavaVirtualMachine
3
abc
1
dbc
进程已结束,退出代码0

1	3
2	abc
3	1
4	dbc

2. 解释 Part I 和 Part II 的输出为什么不同

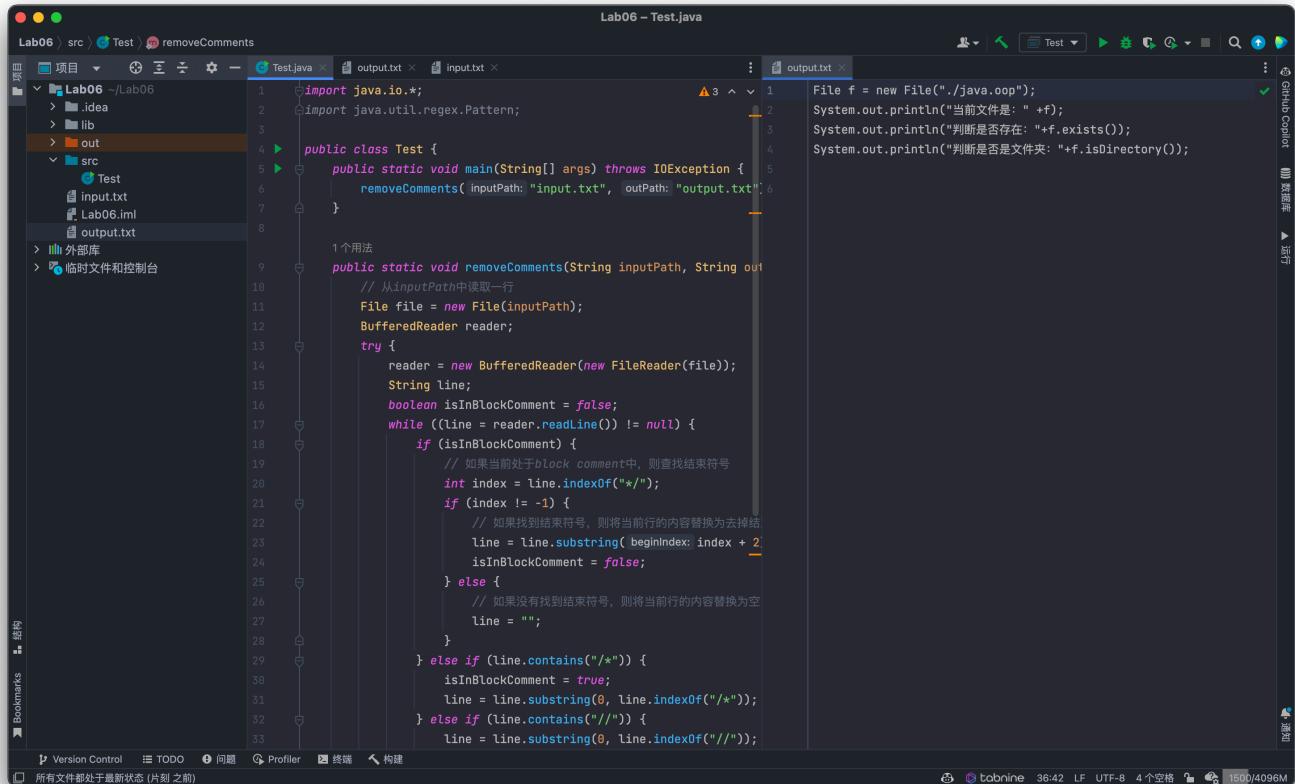
`m` 的值是 `read` 函数的返回值，也就是读取的长度。

因为文件一共只有 `abcd` 四个字符，之前读了三个，所以 `Part II` 的读入只读进来了 1 个字符，故 `m` 的输出是 1。

`tom` 之前的内容是 `"abc"`，读入一个 `"d"` 后自然变成了 `"dbc"`。

Question3 设计一个方法，用于移除文件中的注释

编程



```

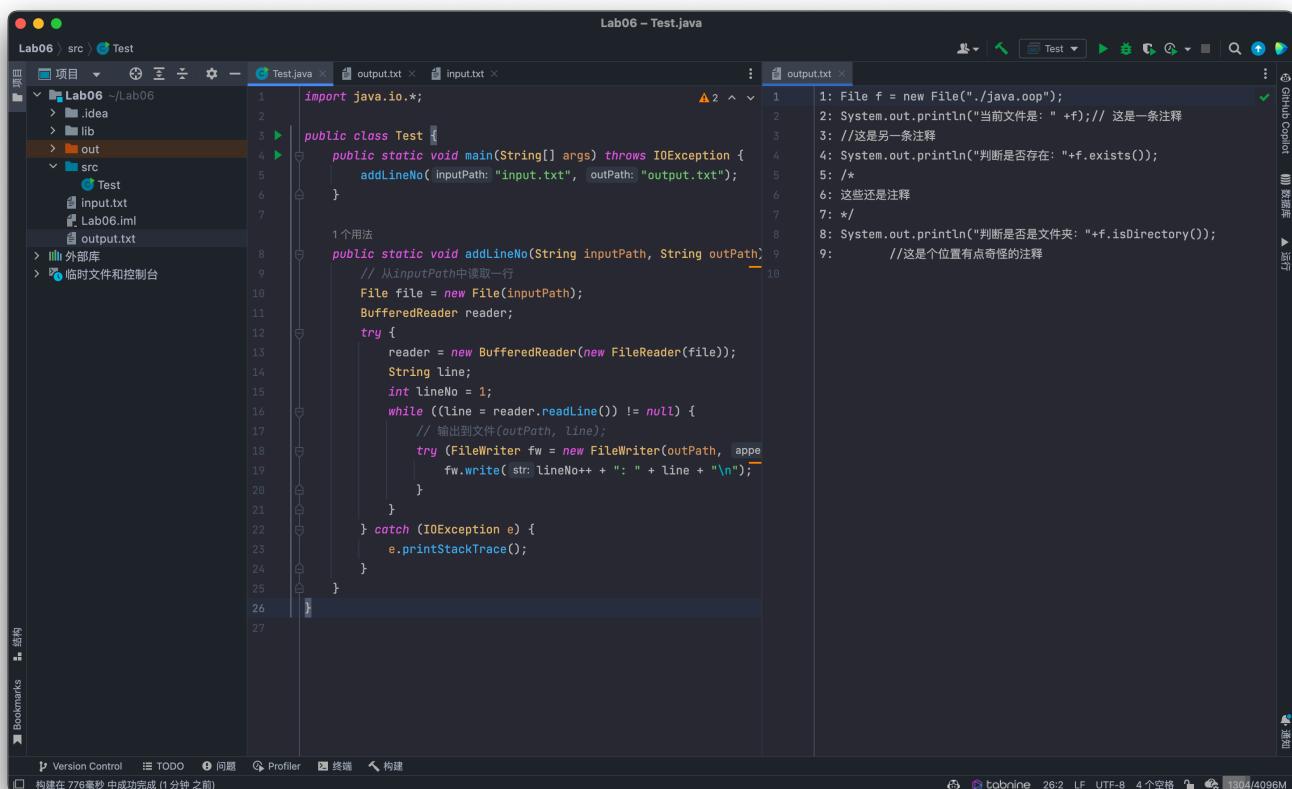
Lab06 - Test.java
Lab06 > src > Test > removeComments
Lab06 ~Lab06
  > .idea
  > lib
  > out
  > src
    > Test
      > input.txt
      > Lab06.iml
      > output.txt
  > 外部库
  > 临时文件和控制台

1个用法
public static void removeComments(String inputPath, String outPath) {
    // 从inputPath中读取一行
    File file = new File(inputPath);
    BufferedReader reader;
    try {
        reader = new BufferedReader(new FileReader(file));
        String line;
        boolean isInBlockComment = false;
        while ((line = reader.readLine()) != null) {
            if (isInBlockComment) {
                // 如果当前处于block comment中，则查找结束符号
                int index = line.indexOf("*/");
                if (index != -1) {
                    // 如果找到结束符号，则将当前行的内容替换为去掉结束符号
                    line = line.substring(0, index + 2);
                    isInBlockComment = false;
                } else {
                    // 如果没有找到结束符号，则将当前行的内容替换为空
                    line = "";
                }
            } else if (line.contains("//")) {
                isInBlockComment = true;
                line = line.substring(0, line.indexOf("//"));
            } else if (line.contains("/*")) {
                line = line.substring(0, line.indexOf("/*"));
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

Question4 设计一个方法，使用 Java 的输入、输出流将一个文本文件的内容按行读出，每读出一行就顺序添加行号，并写入到另一个文件中

编程

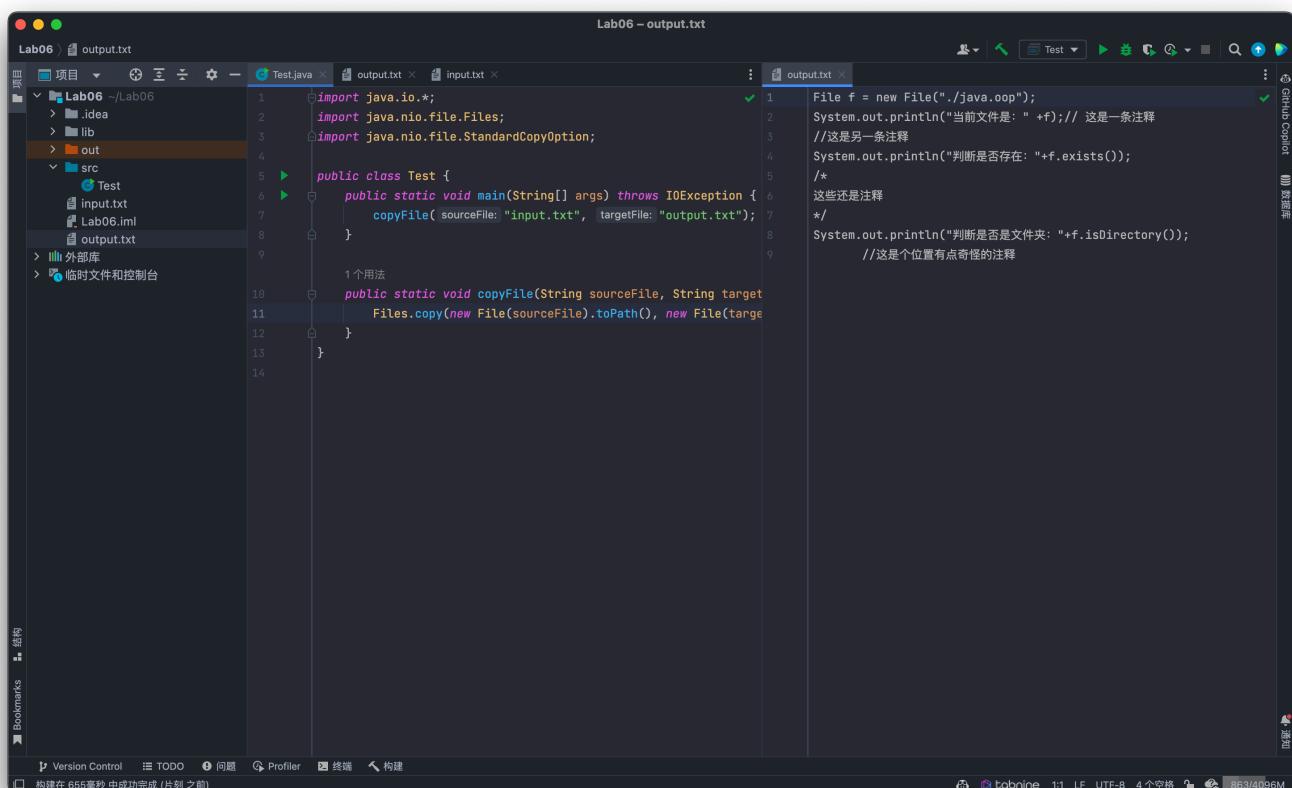


```

Lab06 - Test.java
import java.io.*;
import java.nio.file.*;
import java.nio.file.StandardCopyOption;
public class Test {
    public static void main(String[] args) throws IOException {
        copyLine("input.txt", "output.txt");
    }
}
1 1: File f = new File("./javaoop");
2 2: System.out.println("当前文件是: " + f); // 这是一条注释
3 3: //这是另一条注释
4 4: System.out.println("判断是否存在: " + f.exists());
5 5: /*
6 6: 这些还是注释
7 7: */
8 8: System.out.println("判断是否是文件夹: " + f.isDirectory());
9 9: //这是个位置有点奇怪的注释
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
1 1个用法
public static void copyLine(String inputPath, String outputPath) {
    // 从inputPath中读取一行
    File file = new File(inputPath);
    BufferedReader reader;
    try {
        reader = new BufferedReader(new FileReader(file));
        String line;
        int lineNo = 1;
        while ((line = reader.readLine()) != null) {
            // 输出到文件(outputPath, line)
            try (FileWriter fw = new FileWriter(outputPath, true)) {
                fw.write(str: lineNo++ + ":" + line + "\n");
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

Question5 复制文件是常见的IO操作. 设计如下方法. 实现复制源文件 `sourceFile` 到目标文件 `targetFile` 编程

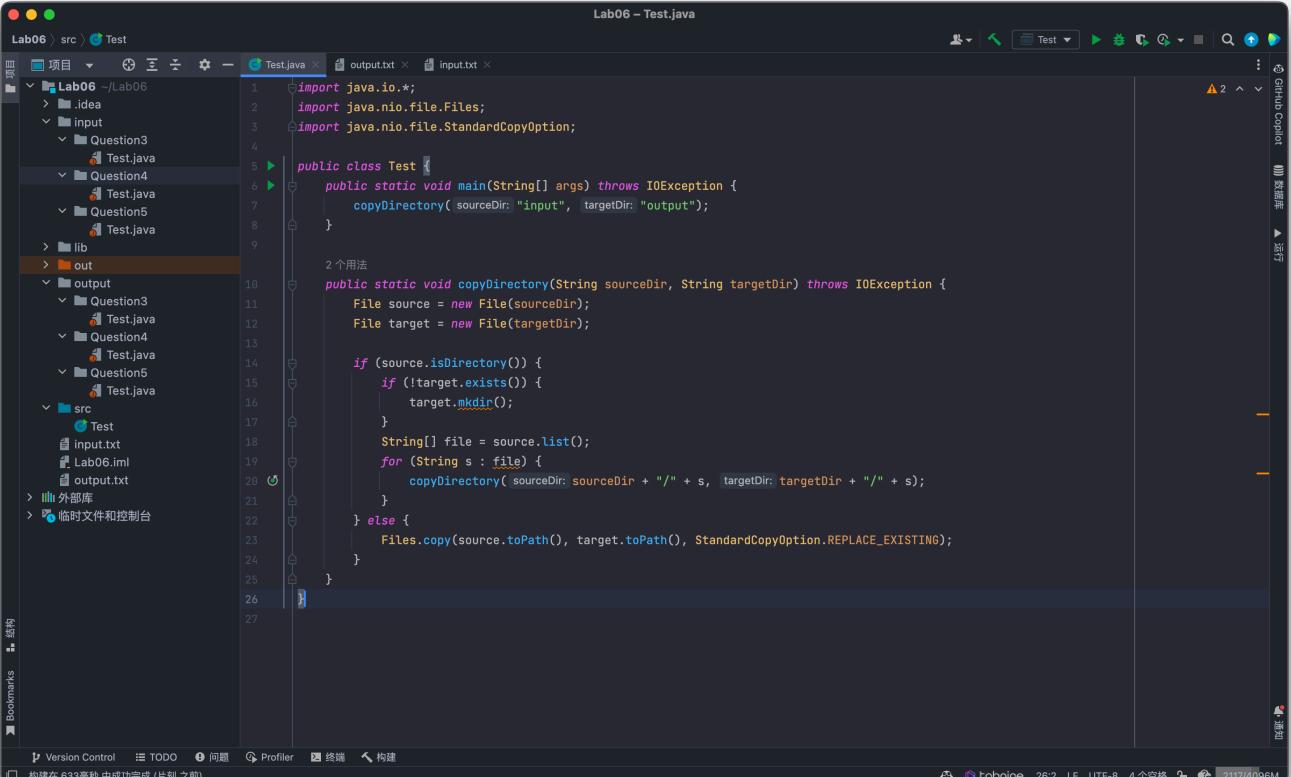


```

Lab06 - output.txt
import java.io.*;
import java.nio.file.*;
import java.nio.file.StandardCopyOption;
public class Test {
    public static void main(String[] args) throws IOException {
        copyFile("input.txt", "output.txt");
    }
}
1 1: File f = new File("./javaoop");
2 2: System.out.println("当前文件是: " + f); // 这是一条注释
3 3: //这是另一条注释
4 4: System.out.println("判断是否存在: " + f.exists());
5 5: /*
6 6: 这些还是注释
7 7: */
8 8: System.out.println("判断是否是文件夹: " + f.isDirectory());
9 9: //这是个位置有点奇怪的注释
10
11
12
13
14
1 1个用法
public static void copyFile(String sourceFile, String targetFile) {
    Files.copy(new File(sourceFile).toPath(), new File(targetFile).toPath(), StandardCopyOption.REPLACE_EXISTING);
}

```

Question6 复制一个文件夹下面所有文件和子文件夹内容到另一个文件夹 编程



The screenshot shows a Java IDE interface with the following details:

- Project Structure:** The project is named "Lab06". It contains a "src" directory with sub-directories "input", "output", and "Question3", "Question4", "Question5", "Question6". Each of these question directories contains a "Test.java" file. There is also a "lib" directory and an "out" directory.
- Code Editor:** The main editor window displays the "Test.java" file for "Question6". The code implements a "copyDirectory" method that recursively copies files and directories from a source directory to a target directory using Java's NIO API.
- Code Snippet:** The code is as follows:

```
1 import java.io.*;
2 import java.nio.file.*;
3 import java.nio.file.StandardCopyOption;
4
5 public class Test {
6     public static void main(String[] args) throws IOException {
7         copyDirectory("input", "output");
8     }
9
10    2 个用法
11    public static void copyDirectory(String sourceDir, String targetDir) throws IOException {
12        File source = new File(sourceDir);
13        File target = new File(targetDir);
14
15        if (source.isDirectory()) {
16            if (!target.exists()) {
17                target.mkdir();
18            }
19            String[] file = source.list();
20            for (String s : file) {
21                copyDirectory(sourceDir + "/" + s, targetDir + "/" + s);
22            }
23        } else {
24            Files.copy(source.toPath(), target.toPath(), StandardCopyOption.REPLACE_EXISTING);
25        }
26    }
27}
```

- Toolbars and Status Bar:** The status bar at the bottom shows "构建在 633毫秒 中成功完成 (片刻之前)". The bottom right corner shows "tabnine 26:2 LF UTF-8 4个空格 2117/4096M".