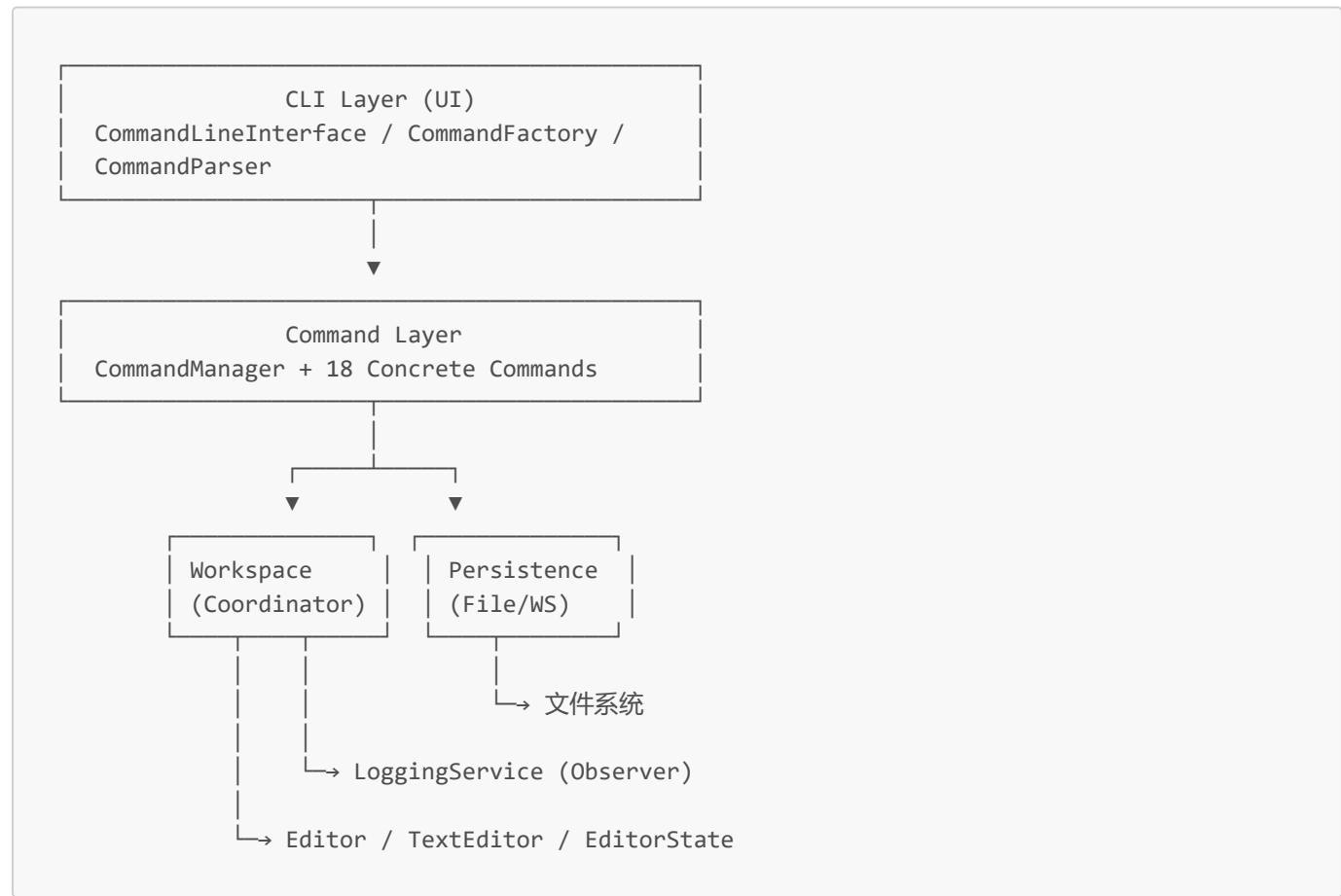


文本编辑器系统架构设计文档

2.1 系统架构

2.1.1 模块划分图



2.1.2 模块职责说明

模块	包路径	主要类	职责
工作区 (Workspace)	com.editor.core.workspace	Workspace EditorManager WorkspaceMemento	统一维护打开文件、活动文件、修改状态、日志开关，负责命令与编辑器的调度，并对外发布事件、生成状态快照
编辑器 (Editor)	com.editor.core.editor	Editor TextEditor EditorState	面向行的文本操作： append/insert/delete/replace/show， 独立维护 undo/redo 与 modified 标记，提供状态快照
命令 (Command)	com.editor.core.command	Command UndoableCommand CommandManager 及 18 个命令实现	将 CLI 的每条指令封装为对象，统一执行、压栈、撤销与重做，区分可撤销和展示类命令

模块	包路径	主要类	职责
日志 (Logging)	com.editor.core.logging	LoggingService EditorEvent EventListener Session	订阅 Workspace 事件，写入 .filename.log，支持 log-on/off/show 以及 #log 自动启用，失败仅警告
持久化 (Persistence)	com.editor.core.persistence	FilePersistence WorkspacePersistence	FilePersistence 负责 UTF-8 文本读写；WorkspacePersistence 负责 .editor_workspace 状态序列化
CLI	com.editor.ui.cli	CommandLineInterface CommandFactory CommandParser	读取用户输入、解析参数、实例化命令、统一输出；CLI 本身不持有业务状态

2.1.3 模块依赖关系

- CLI → Command → Workspace → (Editor / Logging / Persistence)
- Logging 通过 Observer 监听 Workspace，不反向依赖命令层
- Persistence 不依赖任何业务模块，保证可重复利用
- 所有编辑行为都经由 Workspace 统一入口，确保状态一致性

2.2 核心设计

2.2.1 设计模式应用

模式	位置	作用
Command	Command/UndoableCommand、18 个命令、CommandManager	将每条 CLI 指令封装为对象，支持统一执行、撤销、重做与历史记录
Memento	Workspace.saveState/restoreState + WorkspaceMemento + WorkspacePersistence	持久化打开文件列表、活动文件、修改状态、日志开关等工作区快照，启动时恢复
Observer	Workspace (Subject) + LoggingService (Listener)	通过事件通知体系将命令执行、文件打开/关闭/保存等信息推送给日志模块
Decorator (预留扩展)	Editor 接口与潜在 EditorDecorator	当前实现只有 TextEditor，但通过接口定义为 Lab2/Lab3 预留只读、语法高亮等装饰器扩展点

2.2.2 其他关键设计说明

1. **文本存储**：List<String> 逐行存储，方便以行号/列号操作；保存时使用 \n 连接。
2. **Undo/Redo 策略**：每个 TextEditor 自身维护局部历史栈，CommandManager 维护全局命令栈，实现“命令视角撤销”与“单文件多步撤销”两者兼顾。
3. **文件访问顺序**：Workspace 内置 LinkedList 记录最近访问文件，实现 close 命令切回最近使用文件的需求。
4. **日志开关策略**：如果文件首行是 # log 自动启用，否则可通过 log-on/off 手动控制；日志落地路径固定为 .filename.log 并与源文件同目录。
5. **工作区文件格式**：.editor_workspace 采用分段文本格式（# openFiles 等），兼顾可读性与简单解析；加载时具备容错能力，可跳过未知段落或异常行。

lab1.pdf .editor_workspace X CommandFa

E:\daily\2025_autumn\softwareEngineering\LAB1\docs\lab1.

```
1 # openFiles
2 t1.txt
3 # endOpenFiles
4 # activeFile
5 t1.txt
6 # endActiveFile
7 # modifiedStatus
8 t1.txt=false
9 # endModifiedStatus
10 # loggingEnabled
11 t1.txt=true
12 # endLoggingEnabled
13
```

2.3 运行说明

2.3.1 开发环境

项	说明
操作系统	Windows 10 / macOS / Linux（任一终端环境均可）
JDK	Java SE 8（最低）
构建工具	Maven 3.8+（项目提供 <code>pom.xml</code> ，也可使用 IDE 直接运行）

2.3.2 安装依赖步骤

1. 安装 JDK 8，并将 `JAVA_HOME`/`PATH` 配置到 `java`、`mvn` 可执行。
2. 克隆或解压实验项目到任意目录，例如 `E:\daily\2025_autumn\softwareEngineering\LAB1`。
3. 进入项目根目录执行一次 `mvn dependency:resolve`（纯标准库项目，该步骤会非常快）。
4. （可选）在 VS Code / IntelliJ IDEA 中导入 Maven 项目以获得自动补全与单测运行按钮。

2.3.3 运行程序命令

```
mvn -q clean compile
mvn -q exec:java -Dexec.mainClass="com.editor.Main"
# 或者直接使用 VS Code/IDEA 启动配置: .vscode/launch.json 中的 "Run Main CLI"
```

运行后即可在命令行看到提示:

```
输入 'help' 查看帮助, 输入 'exit' 退出
>
```

随后可依照《测试命令.md》进行功能验证。

2.3.4 运行测试命令

```
# 运行全部单元 & 集成测试
mvn -q test

# 运行单个测试类示例
mvn -q -Dtest=TextEditorTest test
mvn -q -Dtest=WorkspaceRestoreIntegrationTest test
```

2.4 测试文档

2.4.1 测试用例列表

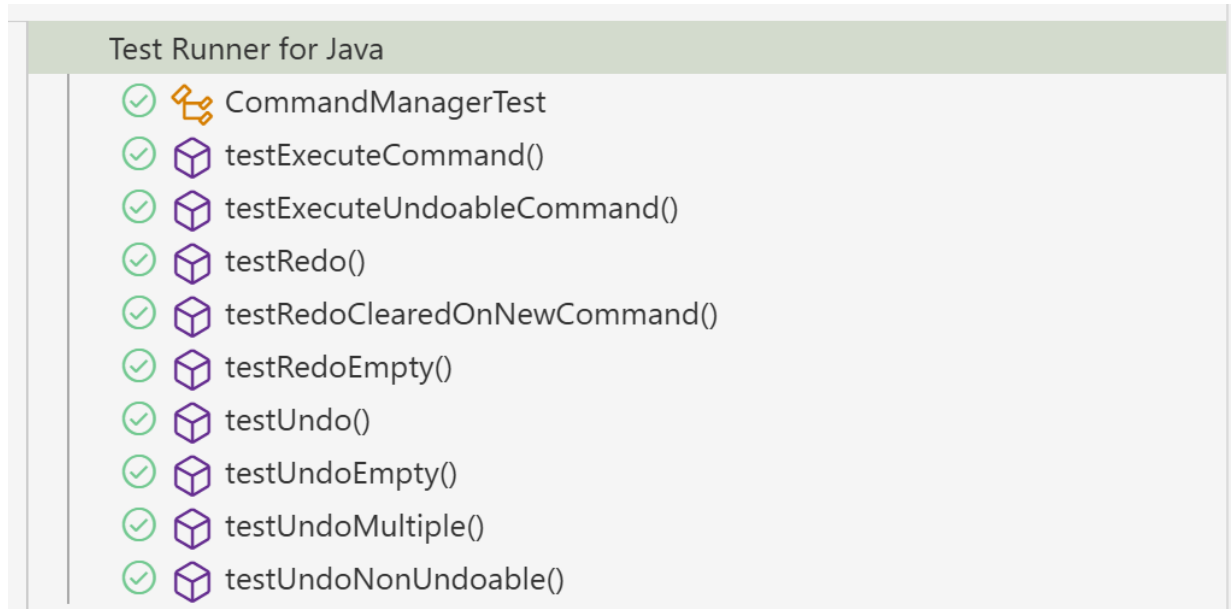
模块	用例/类	场景概要
Editor	TextEditorTest	覆盖 append/insert/delete/replace/show、跨行插入、范围越界、undo/redo 等 20+ 个断言
EditorState	EditorStateTest	验证状态快照的深拷贝、不可变性、空列表/null 安全
Workspace	WorkspaceTest	打开/关闭/切换文件、最近使用列表、日志启停、状态保存与恢复、事件监听
Workspace Manager	EditorManagerTest	多编辑器增删查、集合副本返回等
Workspace Memento	WorkspaceMementoTest	快照构造、防御性拷贝、空状态与 null 输入容错
Persistence	WorkspacePersistenceTest	.editor_workspace 保存/加载、缺省/空文件、段落结构、异常内容容忍
Logging	LoggingServiceTest	日志开关、事件监听、不同事件类型处理、读取日志文件
Command	CommandManagerTest	命令执行、可撤销与不可撤销栈、redo 清空规则、异常分支
CLI	CommandParserTest	命令与参数解析、引号/空格/范围/整数等输入组合
Integration	WorkspaceRestoreIntegrationTest	全量恢复、未保存文件、缺失文件、活动文件恢复、日志状态恢复

若需手工测试，可参考 [docs/测试命令.md](#)，该文档按照命令类别列出了所有交互脚本。

2.4.2 测试执行结果

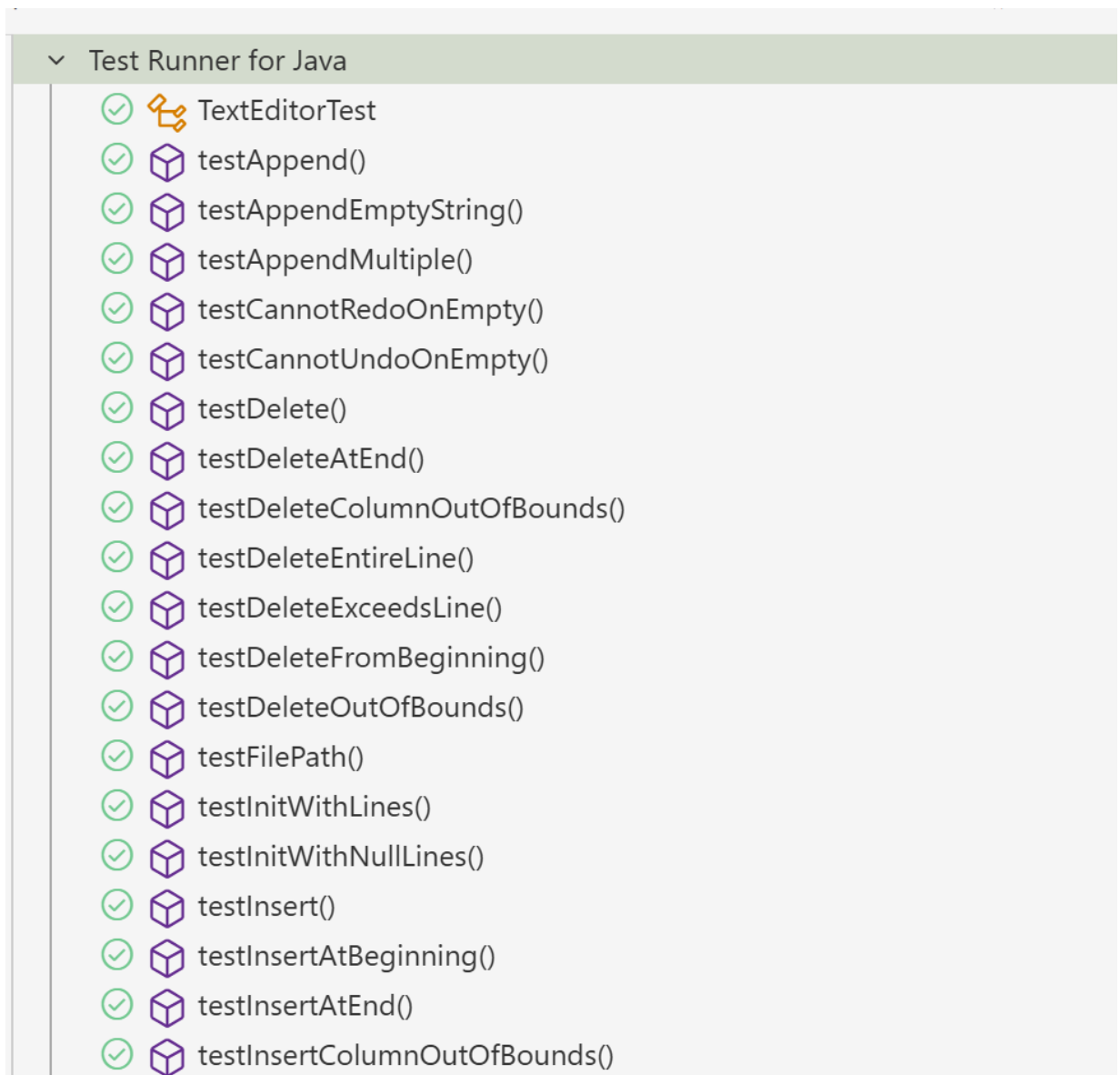
自动化测试

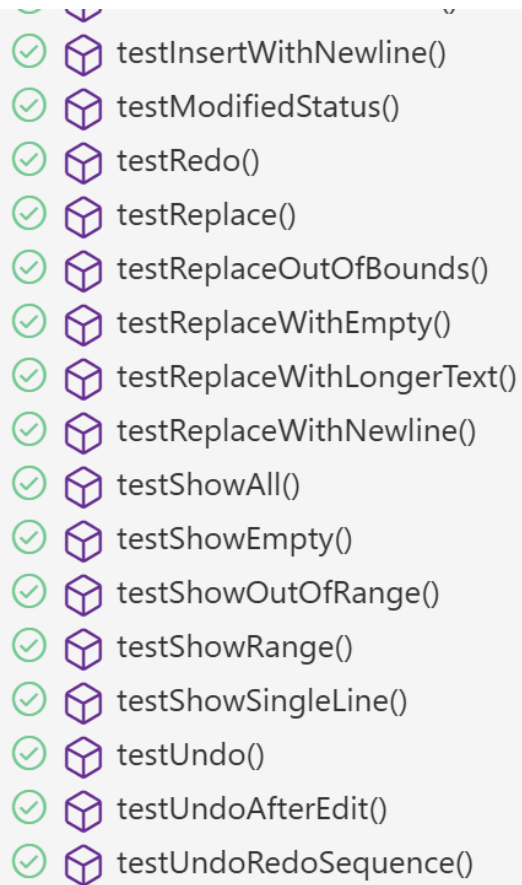
- Command测试
 - CommandManagerTest



- Edito测试

◦ TextEditorTest

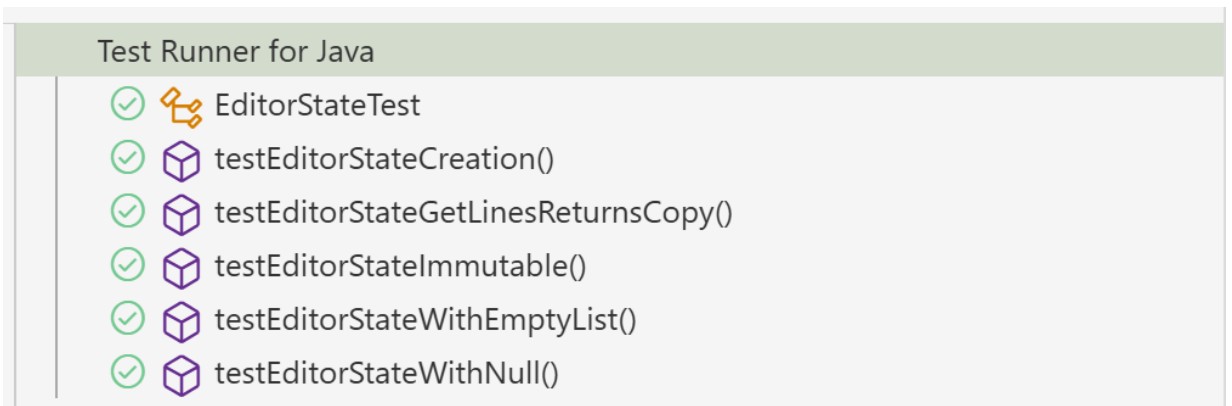




A screenshot of a test runner interface showing a list of 18 test cases, all of which passed. Each test case is preceded by a green checkmark icon and a purple cube icon. The test cases are:

- testInsertWithNewline()
- testModifiedStatus()
- testRedo()
- testReplace()
- testReplaceOutOfBounds()
- testReplaceWithEmpty()
- testReplaceWithLongerText()
- testReplaceWithNewline()
- testShowAll()
- testShowEmpty()
- testShowOutOfRange()
- testShowRange()
- testShowSingleLine()
- testUndo()
- testUndoAfterEdit()
- testUndoRedoSequence()

◦ EditorStateTest



A screenshot of a test runner interface showing a list of 6 test cases, all of which passed. Each test case is preceded by a green checkmark icon and a purple cube icon. The test cases are:













- EditorStateTest
- testEditorStateCreation()
- testEditorStateGetLinesReturnsCopy()
- testEditorStateImmutable()
- testEditorStateWithEmptyList()
- testEditorStateWithNull()

• Workspace测试





















◦ WorkspaceTest

Test Runner for Java	
	 WorkspaceTest
	 testAddListener()
	 testCloseActiveFile()
	 testCloseFile()
	 testCloseLastFile()
	 testCloseNonExistentFile()
	 testDisableLogging()
	 testEnableLogging()
	 testEnableLoggingNotOpen()
	 testGetActiveEditor()
	 testGetActiveEditorNoActiveFile()
	 testGetEditor()
	 testGetEditorNotOpen()
	 testGetUnsavedFiles()
	 testHasUnsavedFiles()
	 testModifiedStatus()
	 testOpenFile()
	 testOpenMultipleFiles()
	 testRemoveListener()
	 testRestoreState()
	 testRestoreStateWithMultipleFiles()
	 testSaveState()
	 testSetActiveFile()
	 testSetActiveFileNotOpen()

◦ WorkspaceMementoTest








Test Runner for Java	
	 WorkspaceMementoTest
	 testWorkspaceMementoCreation()
	 testWorkspaceMementoEmpty()
	 testWorkspaceMementoImmutable()
	 testWorkspaceMementoReturnsCopies()
	 testWorkspaceMementoWithNull()

◦ EditorManagerTest

Test Runner for Java	
	 EditorManagerTest
	 testAddEditor()
	 testGetAllEditors()
	 testGetAllEditorsReturnsCopy()
	 testGetAllFilePaths()
	 testGetEditor()
	 testGetEditorNotExists()
	 testHasEditor()
	 testMultipleEditors()
	 testRemoveEditor()









• Persistence测试

- WorkspacePersistenceTest

Test Runner for Java	
✓ 	WorkspacePersistenceTest
✓ 	testLoadNonExistentWorkspace()
✓ 	testSaveAndLoadEmptyWorkspace()
✓ 	testSaveAndLoadWithNullActiveFile()
✓ 	testSaveAndLoadWorkspaceMemento()
✓ 	testLoadIgnoresUnknownLines()
✓ 	testSaveWritesExpectedSections()

- Logging测试























- LoggingServiceTest

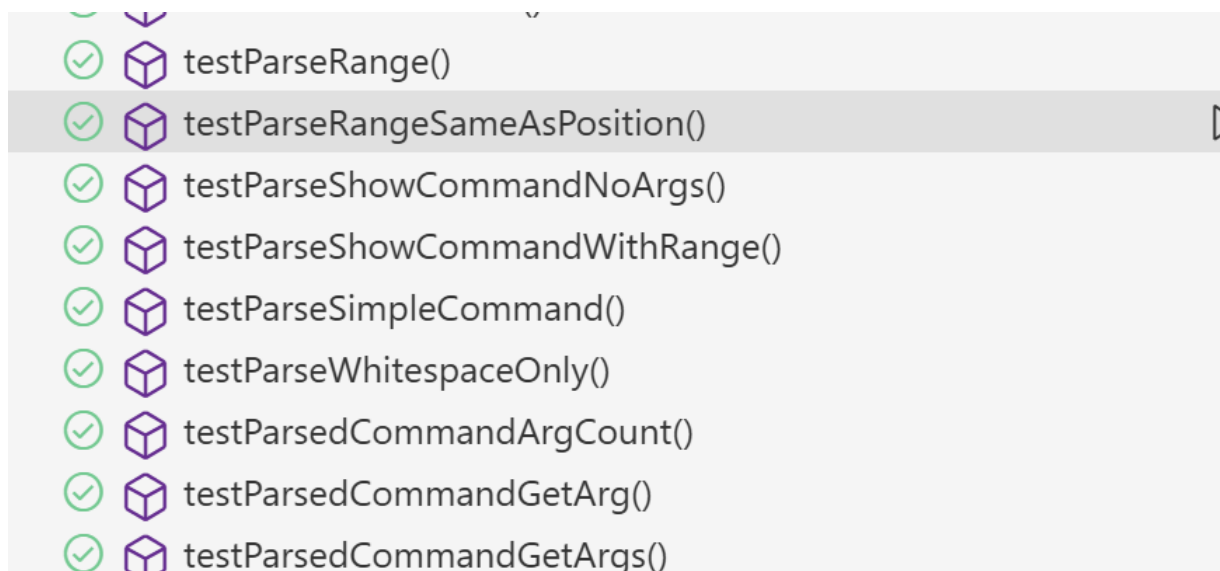
Test Runner for Java	
✓ 	LoggingServiceTest
✓ 	testDifferentEventTypes()
✓ 	testDisableLogging()
✓ 	testEnableLogging()
✓ 	testLoggingDisabledByDefault()
✓ 	testOnEventWithLoggingDisabled()
✓ 	testOnEventWithLoggingEnabled()
✓ 	testOnEventWithNullFilePath()
✓ 	testReadLogFileNotExists()

- CLI测试

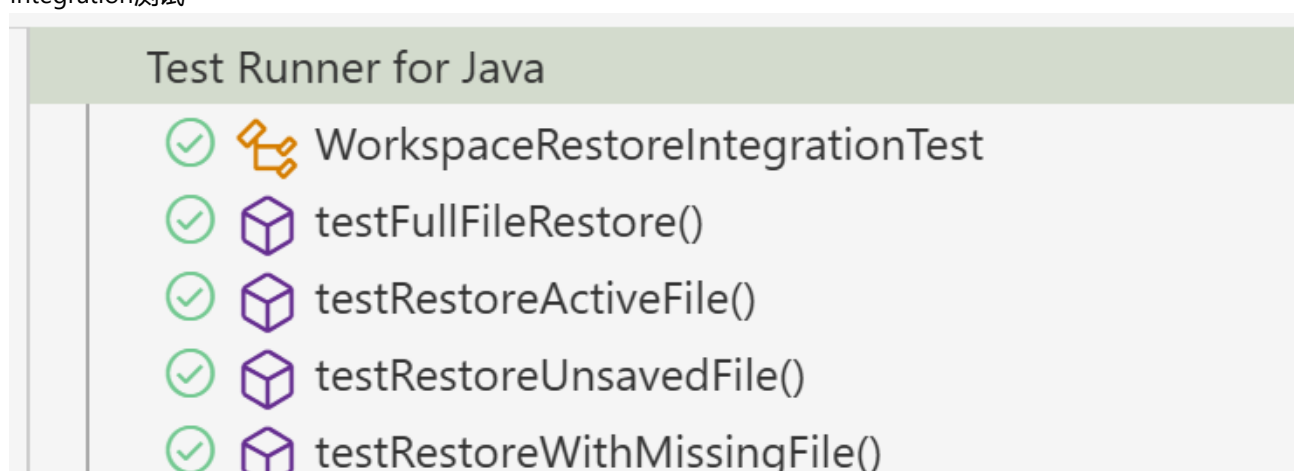
◦ CommandParserTest

▼ Test Runner for Java

- ✓  CommandParserTest
- ✓  testParseCommandWithMixedQuotes()
- ✓  testParseCommandWithMultipleArgs()
- ✓  testParseCommandWithOneArg()
- ✓  testParseCommandWithQuotedArg()
- ✓  testParseCommandWithQuotedArgContainingSpaces()
- ✓  testParseComplexCommand()
- ✓  testParseEmptyInput()
- ✓  testParseInteger()
- ✓  testParseIntegerEmpty()
- ✓  testParseIntegerInvalid()
- ✓  testParseIntegerNegative()
- ✓  testParseIntegerNull()
- ✓  testParseIntegerZero()
- ✓  testParseNullInput()
- ✓  testParsePosition()
- ✓  testParsePositionInvalidFormat()
- ✓  testParsePositionInvalidNumber()
- ✓  testParsePositionLargeNumbers()
- ✓  testParsePositionMultipleColons()
- ✓  testParsePositionNoColon()
- ✓  testParsePositionNull()



- Integration测试



所有用例均通过

手动测试

工作区命令

- 启动编辑器

文本编辑器 v1.0
输入 'help' 查看帮助，输入 'exit' 退出

>

- load file 加载已经存在的文件

```
> editor-list
test1.txt
* test2.txt
> close test1.txt
> editor-list
* test2.txt
> load test1.txt
> editor-list
* test1.txt
test2.txt
```

- 对于首行为 # log, 自动启用日志

```
> load t3.txt
> show
1: # log
> log-show

=== Session started at 2025-11-17 18:36:33 ===
[2025-11-17 18:36:33] FILE_MODIFIED
[2025-11-17 18:36:33] init t3.txt
[2025-11-17 18:36:35] show
[2025-11-17 18:38:39] FILE_SAVED
[2025-11-17 18:38:39] FILE_CLOSED
[2025-11-17 18:38:39] close t3.txt
[2025-11-17 19:02:48] FILE_OPENED
[2025-11-17 19:02:48] load t3.txt
[2025-11-17 19:02:50] show
>
```

- save [file | all] 保存文件

```
> editor-list
t1.txt
* test2.txt [已修改]
> save
> editor-list
t1.txt
* test2.txt
```

- 保存当前文件

```
> editor-list
* t1.txt [已修改]
test2.txt
> save t1.txt
> editor-list
* t1.txt
test2.txt
```

- 保存指定文件

```
> editor-list
t1.txt [已修改]
* test2.txt [已修改]
> save all
> editor-list
t1.txt
* test2.txt
```

- 保存所有文件

- init [file | with-log] 初始化文件

```
> init t1.txt
> show
> init t2.txt
```

- 不带 with-log 参数, 初始化空文件
- 带 with-log 参数, 初始化带日志的空文件

```
> init t3.txt with-log
> show
1: # log
>
```

```
> editor-list
t1.txt
* t3.txt
test2.txt [已修改]
> init t1.txt
错误：文件已存在： t1.txt
```

- 文件已存在，提示错位

```
> init test1.txt
> editor-list
* test1.txt [已修改]
> init test2.txt with-log
> editor-list
test1.txt [已修改]
* test2.txt [已修改]
> show
1: # log
>
```

- close file 关闭文件

- 如当前文件已修改未保存，提示是否保存

```
> editor-list
test1.txt
t1.txt [已修改]
* t3.txt [已修改]
t2.txt [已修改]
test2.txt
> close
文件已修改，是否保存？ (y/n): y
> editor-list
test1.txt
t1.txt [已修改]
* t2.txt [已修改]
test2.txt
□
```

- 当前文件已经保存，则直接关闭

```
> editor-list
test1.txt
t1.txt [已修改]
* t2.txt [已修改]
test2.txt
> save
> close
> editor-list
test1.txt
* t1.txt [已修改]
test2.txt
■
```

```
> editor-list
test1.txt
* t1.txt [已修改]
test2.txt
> close test1.txt
> editor-list
* t1.txt [已修改]
test2.txt
■
```

- 指定文件关闭

```
> editor-list
* t1.txt
test2.txt
> edit test2.txt
> editor-list
t1.txt
* test2.txt
■
```

- edit file 切换活动文件

- 切换错位, 提示文件未打开

```
> editor-list
t1.txt
* t3.txt
test2.txt [已修改]
> init t1.txt
错误: 文件已存在: t1.txt
> edit 3.txt
错误: 文件未打开: 3.txt
```

```
> init test1.txt
> editor-list
* test1.txt [已修改]
> init test2.txt with-log
> editor-list
test1.txt [已修改]
* test2.txt [已修改]
> show
1: # log
> save
> editor-list
test1.txt [已修改]
* test2.txt
> save all
> editor-list
test1.txt
* test2.txt
```

- editor-list 查看编辑器
- dir-tree [path]

- 不带path参数, 查看当前目录

```
test2.txt
> dir-tree
.
├── .vscode
│   ├── launch.json
│   └── settings.json
└── docs
    ├── image.png
    ├── lab1.pdf
    ├── 屏幕截图 2025-11-17 171406.png
    ├── 屏幕截图 2025-11-17 171742-1.png
    ├── 屏幕截图 2025-11-17 171742.png
    ├── 屏幕截图 2025-11-17 171820.png
    ├── 屏幕截图 2025-11-17 171830.png
    └── 屏幕截图 2025-11-17 171932.png
```

- 指定path

```
> dir-tree .vscode
.vscode
├── launch.json
└── settings.json
> █
```

- undo 撤销

```
> append "1"
> show
1: # log
2: Hello
3: world
4: 2
5: 1
6: 1
> undo
> show
1: # log
2: Hello
3: world
4: 2
5: 1
> █
```

- redo 重做

```
> append "1"
> show
1: # log
2: Hello
3: world
4: 2
5: 1
6: 1
> undo
> show
1: # log
2: Hello
3: world
4: 2
5: 1
> redo
> show
1: # log
2: Hello
3: world
4: 2
5: 1
6: 1
```

- exit 退出编辑器

```
=== Session started at 2025-11-17 20:04:07 ===
[2025-11-17 20:04:07] log-on t1.txt
> exit
文件 t1.txt 已修改，是否保存？(y/n): y
PS E:\daily\2025_autumn\softwareEngineering\LAB1>
```

文本编辑命令

```
> editor-list
* test1.txt
  test2.txt
> append "Hello World"
> append "This is line 2"
> append "This is line 3"
> show
1: Hello World
2: This is line 2
3: This is line 3
> editor-list
* test1.txt [已修改]
  test2.txt
```

- append "text" 追加内容

```
> show
1:
2: 1
3: 2
> insert 2:1 good
> show
1:
2: good1
3: 2
> █
```

- insert [line:col] "text" 插入内容

```
> show
1:
2: good1
3: 2
> insert 2:5 "\n3\n4"
> show
1:
2: good
3: 3
4: 41
5: 2
>
```

- 文本中包含换行符

- delete [line:col] 删除内容

```
> show
1:
2: good
3: 3
4: 41
5: 2
> delete 2:2 2
> show
1:
2: gd
3: 3
4: 41
5: 2
>
```

- 正常

```
> show
1:
2: gd
3: 3
4: 41
5: 2
> delete 2:2 3
错误：删除长度超出行尾
```

- 删除长度超过行尾

```
> show
1:
2: gd
3: 3
4: 41
5: 2
> delete 2:2 3
错误：删除长度超出行尾
> delete 6:1 1
错误：行号越界
> delete 2:3 3
错误：列号越界
```

- 行号或者列号超出范围

- `replace [line:col] [len] "text"` 替换内容

```
> show
1:
2: gd
3: 3
4: 41
5: 2
> delete 2:2 3
错误：删除长度超出行尾
> delete 6:1 1
错误：行号越界
> delete 2:3 3
错误：列号越界
> replace 2:1 "\nwood"
错误：replace命令需要位置、长度和文本参数：replace <line:col> <len> "text"
> replace 2:1 2 "\nwd"
> show
1:
2:
3: wd
4: 3
5: 41
6: 2
> █
```

- `show [startLine:endLine]` 显示内容

```
> show
1:
2:
3: wd
4: 3
5: 41
6: 2
```

- 不指定范围，默认显示整个文件

```
> show
1:
2:
3: wd
4: 3
5: 41
6: 2
> show 1:3
1:
2:
3: wd
>
```

- 指定范围

日志命令

- log-on 打开日志
 - 首行是 `# log` 自动启用

```
> load t1.txt
> show
1: # log
> log-show

=== Session started at 2025-11-17 19:40:53 ===
[2025-11-17 19:40:53] FILE_MODIFIED
[2025-11-17 19:40:53] init t1.txt
[2025-11-17 19:40:56] show
[2025-11-17 19:41:06] FILE_SAVED
[2025-11-17 19:41:06] FILE_CLOSED
[2025-11-17 19:41:06] close t1.txt
[2025-11-17 19:41:22] FILE_OPENED
[2025-11-17 19:41:22] load t1.txt
[2025-11-17 19:41:25] show
>
```

◦ 手动启用

```
> edit t7.txt
> show
1: \nq
2: line2
3: lineq
> log-on
> append "2"
> log-show

=== Session started at 2025-11-17 19:44:45 ===
[2025-11-17 19:44:45] log-on t7.txt
[2025-11-17 19:44:54] FILE_MODIFIED
[2025-11-17 19:44:54] append "2"
> █
```

- log-off 关闭日志

```
> log-show

=== Session started at 2025-11-17 19:44:45 ===
[2025-11-17 19:44:45] log-on t7.txt
[2025-11-17 19:44:54] FILE_MODIFIED
[2025-11-17 19:44:54] append "2"
> log-off
> append "3"
[2025-11-17 19:44:54] FILE_MODIFIED
[2025-11-17 19:44:54] append "2"
[2025-11-17 19:44:54] FILE_MODIFIED
[2025-11-17 19:44:54] FILE_MODIFIED
[2025-11-17 19:44:54] FILE_MODIFIED
[2025-11-17 19:44:54] append "2"
> log-off
> append "3"
> log-show

=== Session started at 2025-11-17 19:44:45 ===
[2025-11-17 19:44:45] log-on t7.txt
[2025-11-17 19:44:54] FILE_MODIFIED
[2025-11-17 19:44:54] append "2"
[2025-11-17 19:44:58] log-show t7.txt
```

- log-show 显示日志

```
> show 1:3
1:
2:
3: wd
> log-on
> log-show

=== Session started at 2025-11-17 20:04:07 ===
[2025-11-17 20:04:07] log-on t1.txt
> █
```

Ctrl+K to generate command

```
LAB1
├── src
│   ├── main/java/com/editor
│   │   ├── Main.java
│   │   ├── core
│   │   │   ├── editor/...(Editor, TextEditor...)
│   │   │   ├── workspace/...(Workspace, EditorManager...)
│   │   │   ├── command/
│   │   │   │   ├── Command*.java
│   │   │   │   ├── workspace/*.java    # 10 个工作区命令
│   │   │   │   ├── editor/*.java      # 5 个文本命令
│   │   │   │   └── logging/*.java      # 3 个日志命令
│   │   │   ├── logging/*.java
│   │   │   └── persistence/*.java
│   │   └── ui/cli/*.java
│   ├── test/java/com/editor/...        # 与主代码同结构的测试
│   ├── docs/架构设计.md (本文)
│   ├── docs/测试命令.md
│   ├── pom.xml
│   └── .vscode/launch.json + settings.json
```