Lab1：基于字符命令界面的文本编辑器

实验目标

本实验要求实现一个**基于命令行的文本编辑器**，支持同时打开多个文本文件，提供工作区管理、日志记录、状态持久化等功能。

实验重点考察:

- 面向对象建模能力
- 模块化设计与依赖管理
- 设计模式的合理应用
- 自动化测试能力

文件编码格式：统一使用UTF-8编码

一、功能需求

1. 工作区模块 (Workspace)

核心职责:

- 管理当前会话的全局状态，包括已打开文件列表、当前活动文件、文件修改状态、日志开关等
- 协调命令与具体编辑器之间的交互，例如 `load` / `save` / `close` 等操作
- 状态持久化：程序退出后能保存工作区状态到 `.editor_workspace` 文件，下次启动时恢复
- 发布事件供日志记录等模块订阅

设计要点:

- 工作区中可以有多个编辑中的文件(Editor)，有一个当前的活动文件(Active Editor)
- 每个Editor有独立的undo/redo状态
- ~~**持久化保存的内容**:~~
  - 打开的文件列表及其文件路径
  - 当前活动文件
  - 文件修改状态(modified标记)
  - 日志开关状态
- **不需要持久化的内容**：undo/redo历史记录、编辑时长统计

建议使用的设计模式:

- 备忘录模式 (Memento)：用于工作区状态的持久化和恢复
- 观察者模式 (Observer)：用于事件通知机制

2. 编辑器模块 (Editor)

本实验仅需实现文本编辑器，XML编辑器将在Lab2中引入。

文本编辑器 (TextEditor)

功能职责:

- 支持基本文本编辑操作：追加(append)、插入(insert)、删除(delete)、替换(replace)
- 支持显示操作：指定行号范围显示文本内容(show)
- 所有编辑操作后自动标记文件为已修改
- 提供必要的错误反馈(如范围越界)

数据结构要求:

- 使用**行数组**(`List<String>`)存储文本，每个元素是一行
- 保存时用换行符(`\n`)连接各行
- 这样可以方便地通过行号定位和操作

建议使用的设计模式:

- 命令模式 (Command)：实现undo/redo功能
- 装饰器模式 (Decorator)

文本文件示例:

```
The quick brown fox
jumps over the lazy dog.
This line contains     extra spaces.
```

3. 日志模块 (Logging)

核心功能：记录每一次命令执行，包括执行的时间戳，并持久化到日志文件中。

功能职责:

- 若文件第一行是 `# log`，则打开该文件时**自动启用日志记录**
- 记录每一条命令的执行内容与时间戳
- 每次程序启动视为一次新的会话(Session)，日志以时间段划分
- 支持日志开关，可通过命令手动启用/关闭(`log-on` / `log-off`)
- 支持查看日志记录(`log-show`)
- 日志写入与源文件同目录的 `.filename.log` 文件，永久保存
- 若日志记录失败仅提示警告，不中断程序正常运行

建议使用的设计模式:

- 观察者模式 (Observer)：日志模块作为观察者监听命令执行事件

日志文件格式:

```
session start at 20251024 09:41:33
20251024 09:41:40 load lab.txt
20251024 09:42:05 append "test line"
20251024 09:44:27 save
20251024 09:44:50 close
```

格式说明:

- 每条命令一行，格式为：`时间戳 命令参数`
- 会话开始用 `session start at` 标识
- 命令参数与用户交互时保持一致

示例场景:

文件 `lab.txt` 内容：

```
# log
今天是个写代码的好日子
记得把实验报告补完
```

打开该文件后，自动在 `.lab.txt.log` 中记录操作。

二、命令设计

命令约定

- 命令默认对**当前活动文件**生效
- `<file>` 表示文件路径
- `line:col` 表示行号和列号(**从1开始计数**)
- 带空格的文本参数使用双引号包裹，文本内容本身不包含双引号
- 所有命令参数区分大小写

命令速查表

工作区命令

命令	功能	必需参数	可选参数
`load <file>`	加载文件	文件路径	-
`save [file|all]`	保存文件	-	file/all
`init <file> [with-log]`	创建新缓冲区	文件	with-log
`close [file]`	关闭文件	-	file
`edit <file>`	切换活动文件	文件	-
`editor-list`	显示文件列表	-	-
`dir-tree [path]`	显示目录树	-	path
`undo`	撤销	-	-
`redo`	重做	-	-
`exit`	退出程序	-	-

说明：`save [file|all]` 中的 `|` 表示"或"，即可以指定文件名或使用 `all` 关键字。

文本编辑命令

命令	功能	适用文件
`append "text"`	追加文本	.txt
`insert <line:col> "text"`	插入文本	.txt
`delete <line:col> <len>`	删除字符	.txt
`replace <line:col> <len> "text"`	替换文本	.txt
`show [start:end]`	显示内容	.txt

日志命令

命令	功能	说明
`log-on [file]`	启用日志	file参数可选
`log-off [file]`	关闭日志	file参数可选
`log-show [file]`	显示日志	file参数可选

2.1 工作区命令

1. `load` - 加载文件

```
load <file>
```

功能：加载文件。

行为:

- 文件已存在：读取并解析内容
- 文件不存在：创建新文件，标记为已修改
- 文件成为当前活动文件
- 如果文件已在工作区打开，则切换为活动文件

特殊处理:

- 若文件首行为 `# log`，自动启用日志记录

---

2. `save` - 保存文件

```
save [file|all]
```

功能：保存文件内容到磁盘。

参数说明:

- 不指定参数：保存当前活动文件
- `file`：保存指定文件
- `all`：保存所有已打开的文件

行为:

- 保存成功后清除已修改标记
- 若路径无法写入，提示错误信息

---

3. `init` - 创建新缓冲区

```
init <file> [with-log]
```

功能：创建一个未保存的新缓冲文件，并初始化基础结构。

参数说明:

- `file`：文件路径，如 `test.txt`
- `with-log`(可选)：是否在第一行添加 `# log` 以启用日志

初始化内容:

创建文本文件(`init test.txt with-log`):

```
# log
```

不带 `with-log` 则创建空文件。

说明:

- 新缓冲区标记为已修改，需要使用 `save` 命令指定路径保存
- 创建后自动成为当前活动文件
- 若文件已存在，提示错误

---

4. ~~`close` - 关闭文件~~

```
close [file]
```

功能：关闭当前活动文件或指定文件。

行为:

- 文件已修改且未保存：提示"文件已修改，是否保存? (y/n)"
  - 用户输入 `y`：保存文件后关闭
  - 用户输入 `n`：直接关闭不保存
- 关闭后，如果还有其他打开的文件，切换到最近使用的文件

"最近使用"定义：最后一次通过 `load` 或 `edit` 命令切换到的文件

---

5. ~~`edit` - 切换活动文件~~

```
edit <file>
```

功能：切换当前活动文件。

行为:

- 文件必须已在工作区中打开
- 切换失败提示："文件未打开: [file]"

---

6. ~~`editor-list` - 显示文件列表~~

```
editor-list
```

功能：显示工作区中所有打开的文件及其状态。

显示格式(可选以下任一种):

格式1:

```
* file1.txt [modified]
  file2.txt
```

格式2:

```
> file1.txt*
  file2.txt
```

说明:

- **当前活动文件标记**：`*` (格式1)或 `>` (格式2)
- **已修改未保存标记**：`[modified]` (格式1)或后缀 `*` (格式2)

~~7. `dir-tree` - 显示工作目录文件树~~

```
dir-tree [path]
```

功能：以树形结构显示当前工作目录(或指定目录)的文件和文件夹。

参数说明:

- 不指定参数：显示当前工作目录
- `path`：指定要显示的目录路径

显示格式:

```
├── visitor
│   ├── visitor.worksheet.sc
│   ├── visitor.scala
│   ├── README.md
│   ├── diagram.md
│   └── livedemo
│       ├── visitor.scala
│       └── livedemo.worksheet.sc
└── strategy
    ├── README.md
    ├── livedemo
    │   ├── strategy.scala
    │   └── strategy.worksheet.sc
    ├── strategy.scala
    └── strategy.worksheet.sc
```

说明:

- 使用 `├──`、`└──` 和 `│` 字符绘制树形结构
- 显示目录和文件的层级关系

8. `undo` - 撤销

```
undo
```

功能：撤销上一次编辑操作。

说明:

- 只撤销会改变文件状态的命令
- 显示类命令(show、dir-tree、editor-list等)不进入撤销栈

---

9. `redo` - 重做

```
redo
```

功能：重做上一次撤销的操作。

~~10. `exit` - 退出程序~~

```
exit
```

功能：退出编辑器程序。

行为:

- 保存工作区状态到配置文件
- 若有未保存的文件，逐一提示是否保存

2.2 文本编辑命令

~~1. `append` - 追加文本~~

```
append "text"
```

功能：在文件末尾追加一行文本。

示例:

```
原文：
Hello world

执行：append "New line"

结果：
Hello world
New line
```

---

2. ~~`insert` - 插入文本~~

```
insert <line:col> "text"
```

功能：在指定位置插入文本。

参数说明:

- `line`：行号(从1开始)
- `col`：列号(从1开始)，指定插入位置，插入后新文本将出现在该位置，原有字符向后移动
- `"text"`：要插入的文本内容

行为说明:

- 文本中可以包含换行符(`\n`)，将被解析并自动拆分为多行

异常处理:

- 行号或列号越界：提示"行号或列号越界"
- 空文件插入非1:1位置：提示"空文件只能在1:1位置插入"

示例:

```
原文：
abcdef

执行：insert 1:4 "XYZ"
(列号4表示插入到第4个位置，即原来'd'字符的位置)

结果：
abcXYZdef
```

---

3. ~~`delete` - 删除字符~~

```
delete <line:col> <len>
```

功能：从指定位置的字符开始(包含该字符)删除连续 `len` 个字符。

行为说明:

- 删除范围**不可跨行**
- 删除长度不可超过该行剩余字符数

异常处理:

- 删除长度超出该行剩余字符：提示"删除长度超出行尾"
- 行号或列号越界：提示相应的范围错误

示例:

```
原文:
Hello world

执行: delete 1:7 5
(从第7个字符'w'开始删除5个字符'world')

结果:
Hello
```

---

4. ~~`replace` - 替换字符~~

```
replace <line:col> <len> "text"
```

功能：删除从指定位置起 `len` 个字符，并插入 `"text"`。

行为说明:

- 等效于先执行 `delete`，再执行 `insert`
- 替换文本可为空字符串，效果等同于删除

示例:

```
原文:
fast fox

执行: replace 1:1 4 "slow"

结果:
slow fox
```

---

5. ~~`show` - 显示文本内容~~

```
show [startLine:endLine]
```

功能：显示文本编辑器中指定范围的内容(按行)。

参数说明:

- 不指定参数：显示全文
- `startLine`：起始行号(从1开始)
- `endLine`：结束行号(包含)

适用对象：仅适用于文本编辑器(`.txt` 文件)

示例:

```
> show
1: Hello world
2: This is line 2
3: This is line 3

> show 1:2
1: Hello world
2: This is line 2
```

说明：显示类命令不改变文件状态，不进入撤销栈

2.3 日志命令

1. `log-on` - 启用日志

```
log-on [file]
```

功能：为指定文件(或当前活动文件)启用日志记录。

参数说明:

- 不指定参数：为当前活动文件启用日志
- `file`：为指定文件启用日志

行为说明:

- 后续该文件的所有编辑操作、保存行为将被记录到 `.filename.log` 文件中
- 如果文件首行已是 `# log`，可自动启用

---

2. `log-off` - 关闭日志

```
log-off [file]
```

功能：关闭对指定文件(或当前活动文件)的日志记录。

参数说明:

- 不指定参数：关闭当前活动文件的日志
- `file`：关闭指定文件的日志

行为说明:

- 停止监听该文件的日志事件，但不删除已有日志

---

3. `log-show` - 显示日志

```
log-show [file]
```

功能：显示指定文件(或当前活动文件)的日志记录。

参数说明:

- 不指定参数：显示当前活动文件的日志
- `file`：显示指定文件的日志

输出格式：显示 `.filename.log` 文件的内容。

三、评分指南

评分标准

总分：100分

评分项	分值	说明
架构设计	15分	模块划分、接口设计、依赖管理
自动化测试	15分	要求代码分层，每层都有单独的测试，对测试的覆盖率不做要求
命令实现	60分	各命令功能实现的正确性
代码质量	10分	代码结构、可读性、规范性

1. 架构设计 (15分)

需要提供一个简单的文档，给出各个模块的描述以及依赖关系。可以补充一些关键的设计决策的说明。

评分细则:

评分项	分值	要求
模块划分	4分	模块职责清晰，高内聚低耦合
接口设计	4分	接口抽象合理，符合依赖倒置原则
设计模式应用	4分	正确使用建议的设计模式
依赖管理	3分	第三方库依赖隔离良好

2. 自动化测试 (15分)

针对每层完成自动测试代码。

3. 命令实现 (60分)

命令功能评分 (共18个命令，60分)：

评分规则：

- **基础分**：60分
- **扣分规则**：每缺少一个命令扣5分
- **命令清单**：共18个命令需要实现

工作区命令 (10个)

- load、save、init、close、edit
- editor-list、dir-tree、undo、redo、exit

文本编辑命令 (5个)

- append、insert、delete、replace、show

日志命令 (3个)

- log-on、log-off、log-show

评分要点：

- 命令功能实现正确
- 异常处理完善
- 边界条件处理正确
- 输出格式符合要求

示例：

- 实现全部18个命令：60分
- 缺少2个命令：60 - 2×5 = 50分
- 缺少12个命令及以上将记为0分

4. 代码质量 (10分)

评分细则：

评分项	分值	要求
代码结构	4分	文件组织清晰，模块化良好
命名规范	2分	变量、函数、类命名符合规范
代码可读性	2分	代码逻辑清晰，有必要的注释
编码风格	2分	遵循语言编码规范

四、提交要求

提交时间

实验布置后 **4周** 提交。ddl：2025年11月24日23:59。

提交方式

每人单独完成，提交内容为一份压缩包，以 `学号_姓名.zip` 格式命名。

提交内容

压缩包应包含以下内容：

1. 源代码

- 所有自己编写的代码文件
- 代码应有清晰的目录结构
- **不要提交**第三方库的源代码

2. 架构设计文档

文档应包含以下章节：

2.1 系统架构

- 模块划分图
- 模块职责说明
- 模块依赖关系

2.2 核心设计

- 设计模式应用说明
- 其他设计相关说明

2.3 运行说明

- 使用的编程语言及版本

- 安装依赖的步骤
- 运行程序的命令
- 运行测试的命令

2.4 测试文档

- 测试用例列表
- 测试执行结果

注意事项

1. **代码必须能正常运行**，确保在提交前测试过
2. **文档必须完整**，缺少必要文档将影响评分
3. **遵守学术诚信**，独立完成，不得抄袭，一旦发现抄袭，两份作业都计0分。
4. **按时提交**，逾期提交将扣分，每逾期一天扣除10%。