**虚拟文件系统管理器设计文档**

**一、引言**

**1.1 项目目的**

本项目旨在通过模拟真实文件系统的基本操作，深入理解文件系统的核心原理与实现过程。具体目标如下：

1. 掌握文件系统中的数据结构设计（如树形目录结构、文件节点）和基本算法。
2. 理解文件系统中的目录管理、文件操作、路径解析及状态管理机制。
3. 通过代码实现提升对文件系统设计的编程实践能力。
4. 学习 GUI 界面设计和用户交互的实现方法。

**1.2 项目背景**

本设计文档为基于Java Swing 实现的虚拟文件系统管理器的设计与实现方案。该系统模拟了真实文件系统的基本功能，包括文件和目录的创建、删除、导航等操作，并提供了直观的图形用户界面支持可视化操作和状态监控。

**二、系统设计**

**2.1 整体架构**

系统采用分层架构设计，分为三个主要层次：数据模型层、业务逻辑层和用户界面层。架构如下：

用户界面层 (FileSystemManager)

↓

业务逻辑层 (VirtualFileSystem)

↓

数据模型层 (FileNode)

**2.2 模块划分**

| **模块名称** | **功能描述** | **核心类** |
| --- | --- | --- |
| 数据模型层 | 定义文件和目录的数据结构，管理节点关系 | FileNode |
| 文件系统核心 | 实现文件系统的核心逻辑，提供文件和目录操作 API | VirtualFileSystem |
| 图形用户界面 | 提供用户交互界面，实现操作可视化和状态显示 | FileSystemManager |

**三、详细设计**

**3.1 数据模型层设计**

**3.1.1 FileNode 类：表示文件系统节点**

**属性：**

1. name：节点名称
2. isDirectory：是否为目录
3. content：文件内容（仅文件有效）
4. parent：父目录节点
5. children：子节点映射表
6. createTime：创建时间
7. modifyTime：修改时间
8. size：文件大小

**核心方法：**

1. addChild()：添加子节点，维护父子关系
2. removeChild()：删除子节点
3. getChild()：获取指定名称的子节点
4. getFullPath()：计算并返回完整路径
5. setContent()：设置文件内容并更新修改时间和大小

**3.2 文件系统核心层设计**

**3.2.1 VirtualFileSystem 类：文件系统管理器**

**属性：**

1. root：根目录节点
2. currentDir：当前工作目录
3. openFiles：当前打开的文件集合
4. isFormatted：文件系统格式化状态

**核心方法：**

1. format()：初始化文件系统，创建根目录
2. createDirectory()：在当前目录下创建子目录，包含名称合法性检查和重复性检查
3. removeDirectory()：删除空目录，确保目录为空才能删除
4. changeDirectory()：切换工作目录，支持绝对路径和相对路径（"."、".."、"/"）
5. createFile()：创建文件，检查名称合法性和唯一性
6. openFile()/closeFile()：管理文件打开状态
7. readFile()/writeFile()：读写文件内容，要求文件必须处于打开状态
8. deleteFile()：删除文件，自动关闭已打开的文件
9. listDirectory()：列出当前目录内容，包括"."和".."特殊目录
10. getFileInfo()：获取文件或目录的详细信息

**3.3 用户界面层设计**

**3.3.1 FileSystemManager 类：主界面管理器**

**界面组件：**

1. **顶部状态面板**：
   * 系统状态标签（已格式化/未格式化）
   * 当前路径显示标签
2. **左侧操作面板**：
   * 系统操作：格式化按钮
   * 目录操作：创建目录、删除目录、切换目录按钮
   * 文件操作：创建文件、打开文件、关闭文件、读取文件、写入文件、删除文件、显示信息按钮
3. **右侧显示面板**：
   * **目录浏览选项卡**：显示当前目录内容，支持双击目录快速切换
   * **文件内容选项卡**：显示和编辑文件内容
4. **底部日志面板**：记录操作历史和系统消息

**交互特性：**

1. 双击目录列表中的目录项可快速切换目录
2. 实时更新界面状态和目录内容
3. 操作日志以时间戳形式记录所有用户操作
4. 按钮状态管理：未格式化时禁用大部分操作按钮

图形用户界面

AI 生成的内容可能不正确。

**四、代码结构**

**4.1 核心类列表**

| **类名** | **职责** | **依赖关系** |
| --- | --- | --- |
| FileNode | 文件系统节点数据模型 | - |
| VirtualFileSystem | 文件系统核心逻辑管理 | FileNode |
| FileSystemManager | GUI 界面和用户交互 | VirtualFileSystem、FileNode |

**4.2 关键交互流程**

1. **文件系统初始化流程：** 用户点击格式化按钮 → FileSystemManager 调用 VirtualFileSystem.format() → 创建根目录 FileNode → 更新界面状态
2. **目录操作流程：** 用户输入目录操作参数 → FileSystemManager 触发相应按钮 → 调用 VirtualFileSystem 的目录管理方法 → 更新 FileNode 树结构 → 刷新目录列表和日志
3. **文件操作流程：** 用户选择文件操作 → FileSystemManager 验证输入 → 调用 VirtualFileSystem 的文件管理方法 → 更新 FileNode 状态和 openFiles 集合 → 更新文件内容显示和操作日志

**五、测试方案**

**5.1 文件系统核心功能测试用例**

| **测试场景** | **输入参数** | **预期结果** |
| --- | --- | --- |
| 文件系统格式化 | 无 | 创建根目录"/"，当前路径为"/"，状态变为"已格式化" |
| 创建目录 | 目录名"documents" | 在当前目录下创建 documents 目录，目录列表更新 |
| 切换目录 | 路径"documents" | 当前路径变为"/documents"，目录内容更新 |
| 返回上级目录 | 路径".." | 当前路径变为"/"，返回根目录 |
| 创建文件 | 文件名"test.txt" | 在当前目录下创建 test.txt 文件 |
| 打开并写入文件 | 文件名"test.txt"，内容"Hello World" | 文件状态变为已打开，内容写入成功 |
| 读取文件 | 文件名"test.txt" | 返回文件内容"Hello World" |
| 删除非空目录 | 目录名"documents"（包含文件） | 操作失败，提示"目录不为空" |
| 删除文件 | 文件名"test.txt" | 文件删除成功，自动从打开文件列表移除 |

**图形用户界面

AI 生成的内容可能不正确。**

**图形用户界面, 文本

AI 生成的内容可能不正确。**

**5.2 用户界面功能测试用例**

| **测试场景** | **操作步骤** | **预期结果** |
| --- | --- | --- |
| 双击目录切换 | 双击目录列表中的目录项 | 自动切换到该目录，路径更新 |
| 按钮状态管理 | 启动程序（未格式化状态） | 除格式化按钮外，其他按钮均为禁用状态 |
| 文件内容编辑 | 在文件内容选项卡中编辑文本 | 文本内容实时更新，可用于写入文件 |
| 操作日志记录 | 执行任意文件系统操作 | 日志面板显示带时间戳的操作记录 |
| 路径显示 | 切换到不同目录 | 顶部路径标签实时更新当前位置 |

**5.3 错误处理测试用例**

| **测试场景** | **输入参数** | **预期结果** |
| --- | --- | --- |
| 非法文件名 | 包含"/"的文件名 | 操作失败，文件名不合法 |
| 重复创建 | 已存在的文件/目录名 | 操作失败，名称已存在 |
| 读取未打开文件 | 未打开的文件名 | 操作失败，文件未打开 |
| 切换不存在目录 | 不存在的目录名 | 操作失败，目录不存在 |
| 未格式化操作 | 任意文件系统操作 | 操作失败，需要先格式化文件系统 |

**图形用户界面

AI 生成的内容可能不正确。**

**六、安全特性与约束**

**6.1 输入验证**

* 文件和目录名称不能包含"/"字符
* 禁止使用"."和".."作为用户自定义名称
* 空字符串和空白字符串检查

**6.2 状态管理**

* 文件必须先打开才能进行读写操作
* 目录删除前检查是否为空
* 文件系统必须先格式化才能执行其他操作

**6.3 内存管理**

* 使用 HashSet 管理打开文件状态，避免重复
* 文件删除时自动清理打开状态
* 合理的对象引用管理，避免内存泄漏