一、项目需求

在内存中开辟一个空间作为文件存储器，在其上实现一个简单的文件系统。退出这个文件系统时，需要该文件系统的内容保存到磁盘上，以便下次可以将其恢复到内存中来。

二、具体技术细节

- 文件存储空间管理可采取显式链接（如FAT）或者其他方法。（即自选一种方法）
- 空闲空间管理可采用位图或者其他方法。如果采用了位图，可将位图和FAT表合二为一。
- 文件目录采用多级目录结构。至于是否采用索引节点结构，自选。目录项目中应包含：文件名、物理地址、长度等信息。同学可在这里增加一些其他信息。

三、开发工具

- 环境: intellij
- 语言: Java

四、文件系统管理方案

4.1 存储空间概述

- 盘区数量：10
- 盘区大小：1024KB

4.2 存储空间管理方式

FAT:在本次项目中，我的文件系统管理方案采用了显式链接的方式，在每一个盘区都开了一个文件用来存储当前盘区所有存储块的使用情况。并记录盘区内所有文件所占用内存块的情况

优点：不存在内外内存碎片，最大程度地利用了盘区空间。

4.3 空闲空间管理方式

位图：在本次项目中，我的文件系统空闲空间管理方案采用了位图。使用一串0和1来标识内存块的状态。0为未使用，1表示已经被占用。创建文件时从最前方开始检索空闲块，确定使用后将0置为1。

优点：一目了然，易于管理。

4.4 文件目录结构

多级目录结构

4.5 FCB

4.5.1 概述

该文件系统的FCB将直接显示在生成的txt文本中，以文本头的形势呈现。需要编辑内容可以将内容写在FCB下方的划分线内。

4.5.2 内容

- 大小
- 文件名
- 地址
- 最近更新时间

五、程序操作指南

5.1 特殊文件说明

5.1.1 BitMap&&Fat.txt

- 该文件存储了盘区的存储块的状态，用0表示还未分配，1表示已经分配
- 该文件存储了盘区内存在的文件按照链接方式实现的的内存分配情况，列出了对应内存块的位置

5.1.2 recover.txt

该文件是当前盘区的备份文档，在再次启动该应用程序的时候通过读取该文件的信息还原出上次的运行状态，从而实现复盘

5.2 程序概述

5.2.1 界面构成

- 搜索框
- 文件树
- 显示框
- 底部盘信息栏

5.2.1.1 搜索框

搜索框位于顶部可以在搜索栏中键入文件名或者文件夹名然后点击“start”来进行检索，如果检索成功将直接打开，失败则弹框提醒

5.2.1.2 算法

采用dfs深度搜索的方式

```
// Search a file
    public boolean searchFile(String fileName, File parent){
        File [] files = parent.listFiles();
        for (File myFile:files){
            if (myFile.getName().equals(fileName)){
                try {
                    if(Desktop.isDesktopSupported()) {
                        Desktop desktop = Desktop.getDesktop();
                        desktop.open(myFile);
                        return true;
                    }
                } catch (IOException e1) {
                    JOptionPane.showMessageDialog(null, myFile.getPath() + " Sorry, some thing
wrong!", "Fail to open",
                            JOptionPane.ERROR_MESSAGE);
                    return true;
                }
            }
            if (myFile.isDirectory() && myFile.canRead()){
                if(searchFile(fileName, myFile)){
                    return true;
                }
            }
        }
        return false;
    }
```

5.2.1.3 文件树

文件树位于左侧，呈现文件的目录结构，详情见目录结构

5.2.1.4 显示框

显示框位于程序右侧，用于显示文件的名称、路径、类型、大小和最近修改时间

5.2.1.5 底部盘信息栏

底部盘信息栏位于程序的底部，用于显示当前选中盘区的名称、已使用空间、未使用空间、文件数量

5.2.2 菜单项

- create a file(生成一个文件）
- create a dir(生成一个文件夹)
- delete(删除一个文件或文件夹)
- format(格式化)
- rename(重命名)

5.3 注意事项

- 必须先选择一个系统内的文件夹作为模拟工作目录
- 文件操作需要先选中一个文件树中的节点右键才能进行操作
- 本程序重在模拟，并不是真正地为文件开了这么大的空间
- 仅支持生成txt文件，在输入文件名时不需要输入.txt
- 相同目录下已经存在该文件重命名将失败
- 文本中将自动生成FCB，不支持用户去修改FCB
- 非法输入都将导致文件生成失败
- 如果存档文件recover.txt被破坏，将无法在再次打开该程序时恢复盘区信息

六、文件系统实现功能

6.1 文件目录

文件目录将呈现在程序的左侧，以十个以数字命名的盘为基目录。 点击文件夹左侧箭头可以将文件夹展开

当执行创建和删除文件等操作，必须重新展开一次父目录才能刷新

6.2 创建文件

6.2.1 概述

选中某一文件后，右键展开功能选项，选择第一项生成文件或第二项就会弹出对话框要求键入文件名和文件大小。文件生成成功之后只要重新展开父目录就可以看到新生的文件。双击程序右侧的显示区的文件就可以打开该文件看到它的FCB

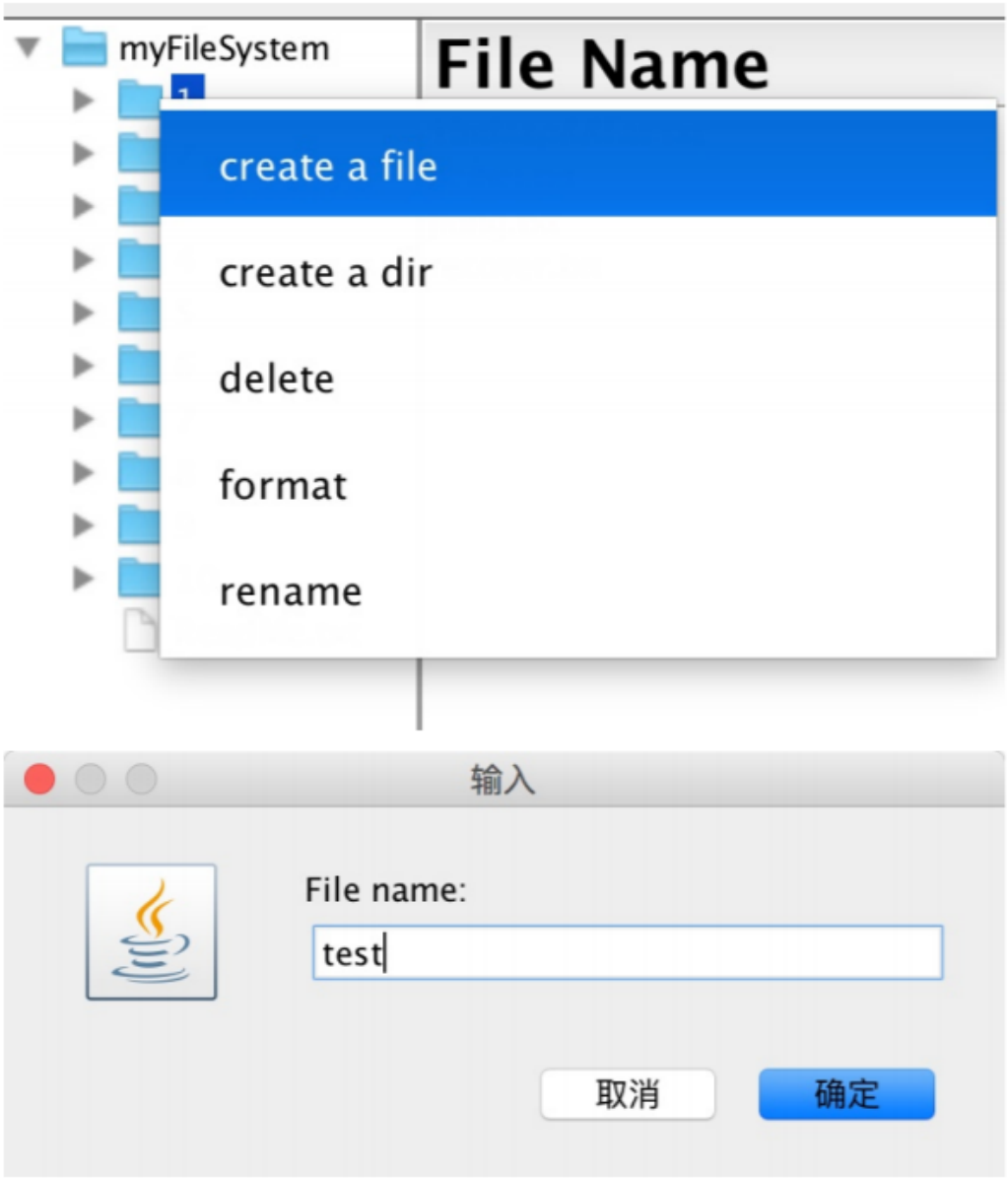
6.2.2 实现方法

通过调用File类的createFile()方法来实现，文件创建成功之后将刷新FAT和位图，如果存储空间不足或是输入非法将导致文件生成失败

```java
// create a file
    public  boolean createFile(String filePath){
        boolean result = false;
        File file = new File(filePath);
        if(!file.exists()){
            try {
                result = file.createNewFile();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        return result;
    }
```

当执行创建文件操作，必须重新展开一次父目录才能刷新



6.3 删除文件

6.3.1 概述

选中某一文件后，右键展开功能选项，选择第三项删除即可删除所选项

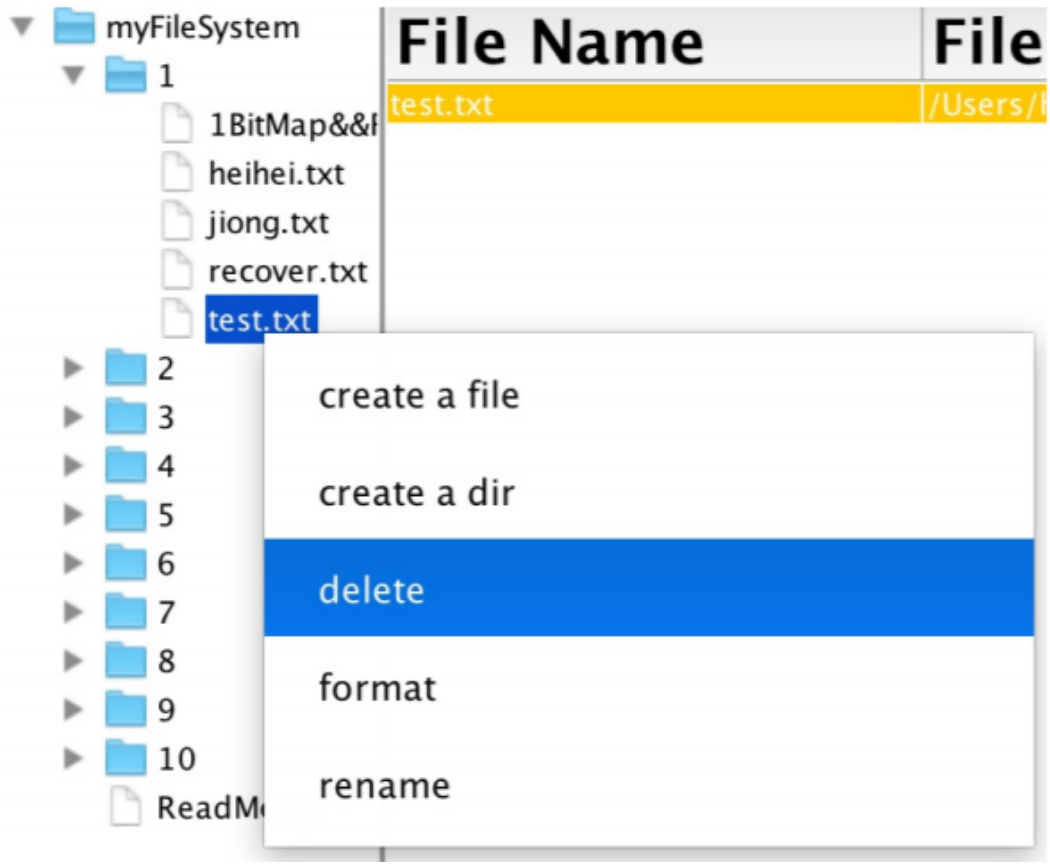
6.3.2 实现方法

通过调用File类的delete()删除 针对文件夹需要递归删除，文件删除成功之后将刷新FAT和位图

```
// delete a file
    public  boolean deleteFile(String filePath){
        boolean result = false;
        File file = new File(filePath);
        if(file.exists() && file.isFile()){
            result = file.delete();
        }

        return result;
    }
```

当执行删除文件操作，必须重新展开一次父目录才能刷新



6.4 重命名（更改当前目录）

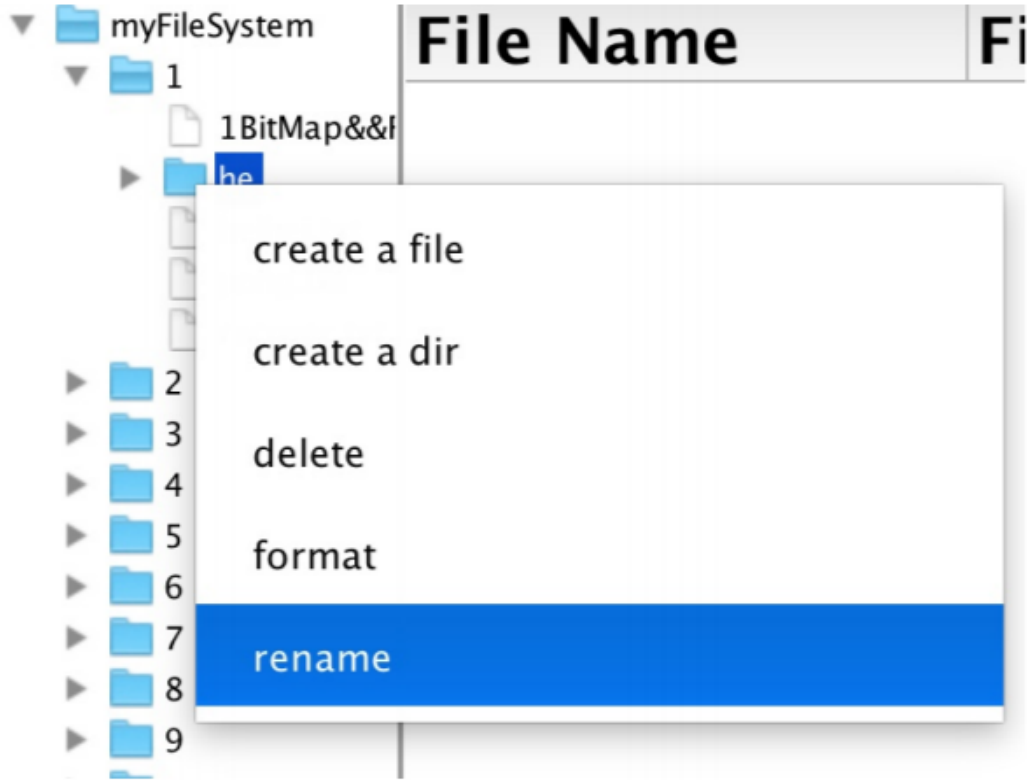
6.4.1 概述

选中某一文件后，右键展开功能选项，选择第四项重命名即可对文件或文件夹进行重命名

6.4.2 实现方法

通过调用File类的renameTo()方法进行重命名，如果相同目录下有相同的文件，则重命名将失败

当执行重命名文件操作，必须重新展开一次父目录才能刷新



6.5 打开文件

双击程序中右侧的显示面板中的文件即可打开对应文件

6.6 关闭文件

打开文件之后点击文件上方的关闭按钮就可以将对应文件关闭

6.7 写文件

打开文件之后在FCB的内容下方输入文本即可

6.8 读文件

双击程序右侧的显示面板中的文件即可打开对应文件进行查看

6.9 格式化

6.9.1 概述

选中某一文件夹之后右键选择第四项格式化就可以将该文件夹格式化

6.9.2 实现方法

即递归删除该目录下的所有文件并更新FAT和位图

七、 数据结构

盘区（Block）

```java
public class Block {
    // 名称
    private int blockName;
    // 盘区内的文件
    private File blockFile;
    // 位图存储文件
    private File blockBitMap;
    // 恢复文件
    private File recover;
    // 位图存储文件书写器
    private FileWriter bitWriter;
    // 恢复文件书写器
    private FileWriter recoverWriter;
    // 文件数量
    private int fileNum;
    // 盘区已占用空间
    private double space;
    // 内存块数组
    public int [][] bitmap = new int[32][32];
    // 文件和内存块对应表
    private Map<String, int[][] > filesBit = new HashMap<String, int[][]>();
    // 文件列表
    private ArrayList<File> files = new ArrayList<File>();

    public Block(int name, File file, boolean rec) throws IOException {
        // 初始化
    }

    public File getBlockFile(){
        return blockFile;
    }

    public void putFCB(File file, double capacity) throws IOException {
        // 将FCB写入文件
    }


    public void rewriteBitMap() throws IOException {
        // 重写位图存储文件
    }

    public void rewriteRecoverWriter() throws IOException{
        // 重写恢复文件
    }

    public boolean createFile(File file, double capacity) throws IOException {
        // 在盘区内创建文件
    }

    public boolean deleteFile(File file, double capacity){
        // 在盘区内删除文件
    }
```

```
    public boolean renameFile(File file, String name, double capacity) throws IOException {
        // 重命名盘区内的文件
    }

    public int getFileNum() {
        return fileNum;
    }

    public double getSpace() {
        return space;
    }
}
```