同济大学

操作系统——文件管理

模拟磁盘文件系统实现

姓名：赖慧琳

学号：1652802

专业：软件工程

目录

1. 项目目的

1、熟悉文件存储空间的管理；
2、熟悉文件的物理结构、目录结构和文件操作；
3、熟悉文件系统管理实现；
4、加深对文件系统内部功能和实现过程的理解

2. 开发工具

选择语言：java
系统平台：windows10
Jdk: 1.8
IDE: Eclipse Java Oxygen

3. 项目需求

3.1 基本要求

1、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。
2、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。

3.2 具体技术细节

1、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）
2、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。
3、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。可在这里增加一些其他信息。

3.3 可提供的操作

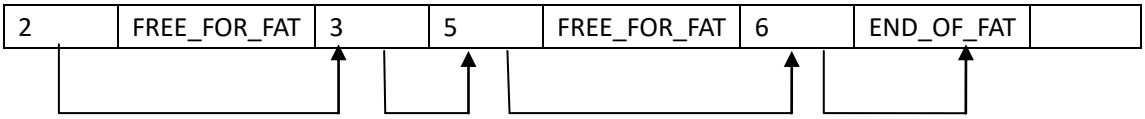
要求该文件系统至少提供以下操作：

- 格式化
- 创建子目录
- 删除子目录
- 显示目录
- 更改当前目录
- 创建文件
- 打开文件
- 关闭文件
- 写文件
- 读文件
- 删除文件

4. 项目设计

4.1 结构设计

1、本系统用一个大小为[128][256]的二维 char 型数组来模拟一个共 128 个盘块，每个盘块 256 字节的内存空间。
2、本系统用一个大小为[128]的一维 char 型数组来模拟 FAT 表，同时空闲空间管理采用位图方式，位图与 FAT 表相结合。FAT 表的结构如下图所示：



3、文件目录信息的总长度设置为 32 字节，其中 16 字节为名字，1 字节类型（指明是文件还是子目录），1 字节文件大小（子目录该位不用），1 字节 FAT 起始块号。如下图所示：

文件名	文件类型	文件大小	FAT 起始块号	未用
16 字节	1 字节	1 字节	1 字节	13 字节

当存储的是子目录时，会利用盘块的前两个文件目录项，保存当前文件夹和上一级文件夹。'.'代表当前文件夹；'..'代表上级文件夹。

4.2 类的设计

主要有三个 java 源文件
Utility：集中定义了程序中需要的具体参数值。
主要属性：WIDTH HEIGHT DIR_LEN POS_NAME POS_TYPE POS_SIZE FILE
POS_FAT NUM_OF_BLOCK FREE_FOR_FAT END_OF_FAT DIRECTORY

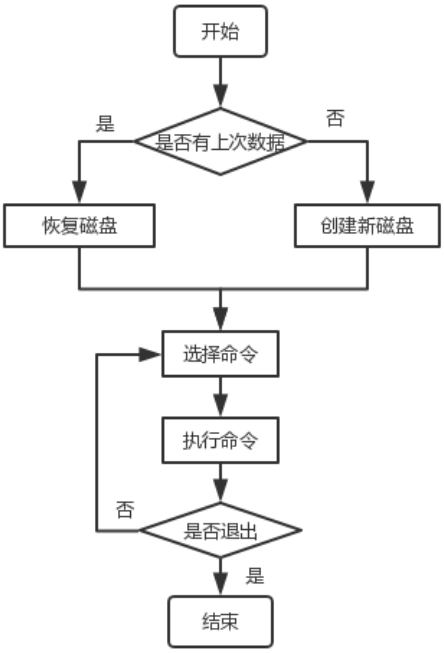
FileManager：包含了界面的总体布局。正上方是当前目录显示，接下来是八个选项选择操作。文件和子目录的选择采用复选框的形式，下拉菜单选择。编辑文件时会跳出文件内容框，可以更改文件内容，选择保存或者取消。

DiskManager：包含文件操作类的定义，主要是对内存和 FAT 表的操作。
主要属性：fatTable: char[] //FAT 表 disc: char[][] //内存数据空间
nowBLOCK: int //当前盘块号 nowNumb:int //当前偏移量
主要方法：formatAll():void //格式化信息
reload(ArrayList, ArrayList):void //重新载入上次数据
addFile(int, String):boolean //创建新文件
addDirectory(int, String):boolean //创建新子目录
assignBlock():int //分配 FAT 表块
getFileContent(int, String):String //得到文件内容

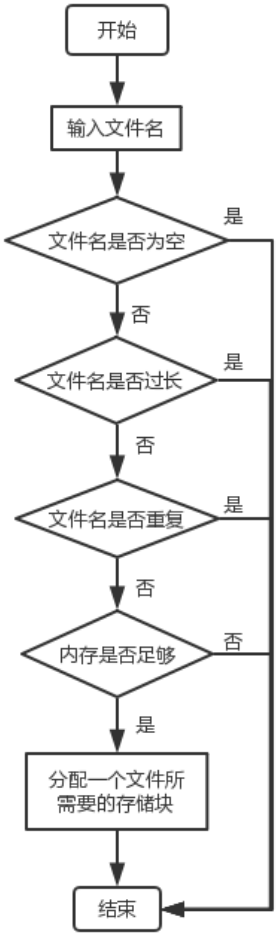
```
delDirectory(int, String):void          //按名字删除子目录
delDirectory(int):void                  //按位置删除子目录
delFile(int, String):void               //按名字删除文件
delFile(int, int):void                  //按位置删除文件
nextDirectory(int, String, ArrayList):int  //下一级目录
lastDirectory(int, String, ArrayList):int  //上一级目录
saveFile(int, String, String):boolean   //保存文件
```

4.3 流程图

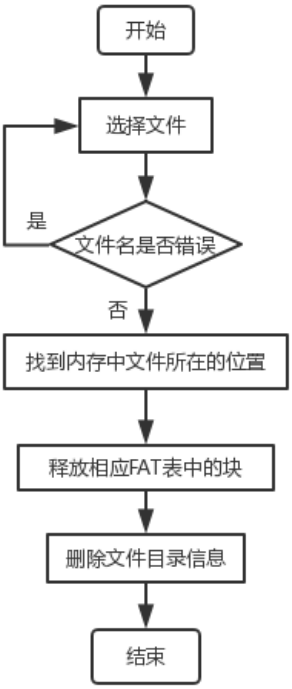
4.3.1 整体流程图



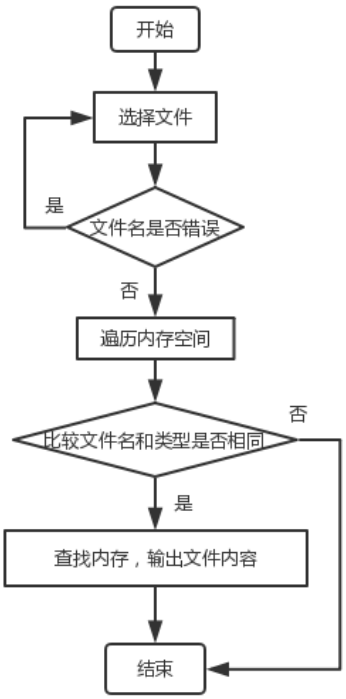
4.3.2 创建文件流程图


开始
输入文件名
文件名是否为空
是
否
文件名是否过长
是
否
文件名是否重复
是
否
内存是否足够
否
是
分配一个文件所
需要的存储块
结束

4.3.3 删除文件流程图



4.3.4 打开文件流程图



5. 项目实现

5.1 创建新文件

```
public boolean addFile(int curBlock,String name) {
 int i;
 //找到数据区的空闲块
 for(i = 0; i < Utility.NUM_OF_SUBFILE; i++) {
     if(disc[curBlock][i*Utility.DIR_LEN] == ' ') {
         break;
     }
 }
 //分配文件信息
 if(i != Utility.NUM_OF_SUBFILE) {
     for(int j = 0; j < name.length(); j++) {
         disc[curBlock][i*Utility.DIR_LEN + j] = name.charAt(j);
     }
     disc[curBlock][i*Utility.DIR_LEN + Utility.POS_TYPE] = Utility.FILE;
     disc[curBlock][i*Utility.DIR_LEN + Utility.POS_SIZE] = 0;
     disc[curBlock][i*Utility.DIR_LEN + Utility.POS_FAT] = (char) assignBlock();
     return true;
 }
 //无空闲块
 else {
     return false;
 }
}
```

5.2 创建新子目录

```
public boolean addDirectory(int curBlock,String name) {
 int freeBlock = assignBlock();
 if(freeBlock == -1) {
     return false;
 }
 int i;
 for(i = 0; i < Utility.NUM_OF_SUBFILE; i++) {
     if(disc[curBlock][i*Utility.DIR_LEN] == ' ') {
         break;
     }
 }
```

```
//分配子目录信息
if(i != Utility.NUM_OF_SUBFILE) {
    for(int j = 0; j < name.length(); j++) {
        disc[curBlock][i*Utility.DIR_LEN + j] = name.charAt(j);
    }
    disc[curBlock][i*Utility.DIR_LEN + Utility.POS_TYPE] = Utility.DIRECTORY;
    disc[curBlock][i*Utility.DIR_LEN + Utility.POS_SIZE] = 0;
    disc[curBlock][i*Utility.DIR_LEN + Utility.POS_FAT] = (char)freeBlock;
    //利用前两块，  .代表当前文件夹  ..代表上级文件夹
    disc[freeBlock][Utility.POS_NAME] = '.';
    disc[freeBlock][Utility.POS_TYPE] =Utility.DIRECTORY;
    disc[freeBlock][Utility.POS_FAT] = (char)freeBlock;
    disc[freeBlock][Utility.DIR_LEN] = '.';
    disc[freeBlock][Utility.DIR_LEN + 1]='.';
    disc[freeBlock][Utility.DIR_LEN + Utility.POS_TYPE] = Utility.DIRECTORY;
    disc[freeBlock][Utility.DIR_LEN + Utility.POS_FAT] = (char)curBlock;
    return true;
}
else {
    return false;
}
}
```

5.3 删除文件

```
//删除文件(按位置)
public void delFile(int block,int numb) {
 delBlocks((int)disc[block][numb * Utility.DIR_LEN + Utility.POS_FAT]);
 clearDirectory(block,numb);
}
//删除单条目录文件信息
public void clearDirectory(int block, int numb) {
 for(int i = 0; i < Utility.DIR_LEN; i++) {
     disc[block][numb * Utility.DIR_LEN + i] = ' ';
 }
}
//递归删除相应的fat表中的块
public void delBlocks(int blockNumb) {
 if(fatTable[blockNumb] != Utility.FREE_FOR_FAT
     && fatTable[blockNumb] != Utility.END_OF_FAT) {
     delBlocks((int)fatTable[blockNumb]);
 }
 fatTable[blockNumb] = Utility.FREE_FOR_FAT;
}
```

5.4 删除子目录

```
public void delDirectory(int block) {
 for(int i = 2; i < Utility.NUM_OF_SUBFILE; i++) {
     if(disc[block][i * Utility.DIR_LEN] == ' ') {
         continue;
     }
     //目录下有文件
     if(disc[block][i * Utility.DIR_LEN + Utility.POS_TYPE] == Utility.FILE) {
         delFile(block,i);
     }
     //目录下有子目录
     if(disc[block][i * Utility.DIR_LEN + Utility.POS_TYPE] == Utility.DIRECTORY){
         int subBlock = disc[block][i * Utility.DIR_LEN + Utility.POS_FAT];
         delDirectory(subBlock);
     }
 }
 fatTable[block] = Utility.FREE_FOR_FAT;   //FAT表设置为空闲
}
```

5.5 进入下一级目录

```
public int nextDirectory(int curBlock, String dirName, ArrayList curDirs,
ArrayList curFiles) {
    locDir(curBlock, dirName);
    int newCurBlock = disc[nowBLOCK][nowNumb * Utility.DIR_LEN +
    Utility.POS_FAT];
    curDirs.clear();
    curFiles.clear();
    for(int i = 2; i < Utility.NUM_OF_SUBFILE; i++) {
        String name = getDirectoryName(newCurBlock, i);
     if(name != "" && disc[newCurBlock][i*Utility.DIR_LEN +
     Utility.POS_TYPE]==Utility.DIRECTORY) {
            curDirs.add(name);
        }
     else if(name != "" && disc[newCurBlock][i*Utility.DIR_LEN +
     Utility.POS_TYPE]==Utility.FILE) {
            curFiles.add(name);
        }
    }
    return newCurBlock;
    }
```

5.6 返回上一级目录

```
public int lastDirectory(int curBlock, ArrayList curDirs, ArrayList curFiles) {
     if(disc[curBlock][Utility.DIR_LEN + Utility.POS_TYPE] == Utility.DIRECTORY) {
          int newCurBlock = disc[curBlock][Utility.DIR_LEN + Utility.POS_FAT];
          curDirs.clear();
          curFiles.clear();
          for(int i = 2; i < Utility.NUM_OF_SUBFILE; i++) {
               String name = getDirectoryName(newCurBlock, i);
          if(name != "" && disc[newCurBlock][i*Utility.DIR_LEN +
Utility.POS_TYPE]==Utility.DIRECTORY) {
                    curDirs.add(name);
               }
          else if(name != "" && disc[newCurBlock][i*Utility.DIR_LEN +
Utility.POS_TYPE]==Utility.FILE) {
                    curFiles.add(name);
               }
          }
          return newCurBlock;
     }
     else {
          return -1;   //已经是根目录
     }
 }
```

5.7 保存文件

```
public boolean saveFile(int curBlock,String fileName, String text) {
        int i;
    for(i = 0; i < Utility.NUM_OF_SUBFILE; i++) {
        if(fileName.equals(getDirectoryName(curBlock, i))) {
            break;
        }
    }
    disc[curBlock][i*Utility.DIR_LEN + Utility.POS_SIZE] = (char)text.length();
    int textBlockNo = (int) disc[curBlock][i*Utility.DIR_LEN + Utility.POS_FAT];
    if(fatTable[textBlockNo] != Utility.FREE_FOR_FAT && fatTable[textBlockNo] !=
Utility.END_OF_FAT) {
        delBlocks((int)fatTable[textBlockNo]);
    }
    for(int j = 0; j <text.length(); j++) {
        disc[textBlockNo][j] = text.charAt(j);
    }//一块盘区的算法
```

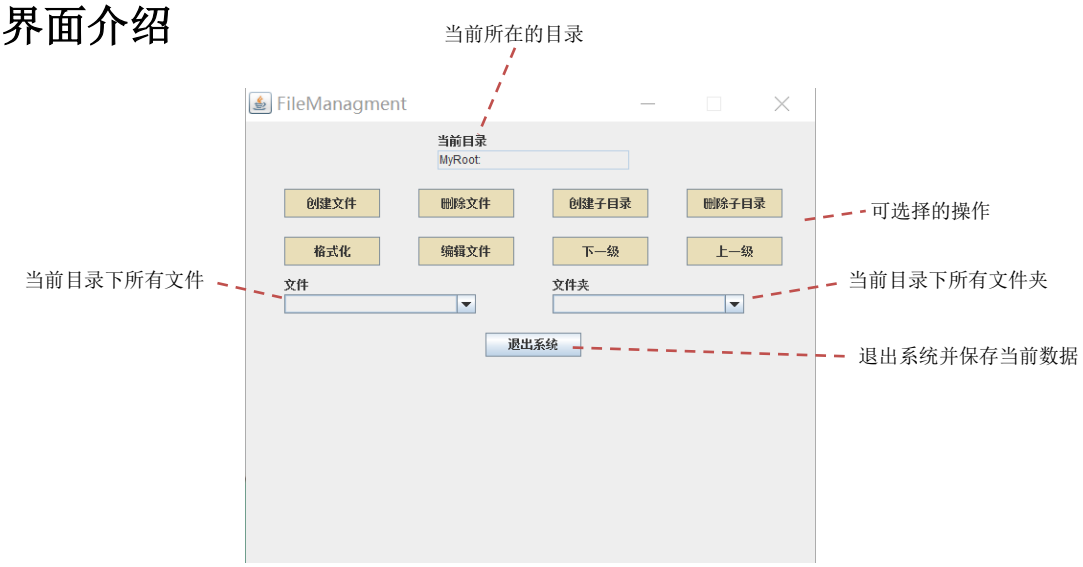
```
return true;
}
```

5.8 格式化

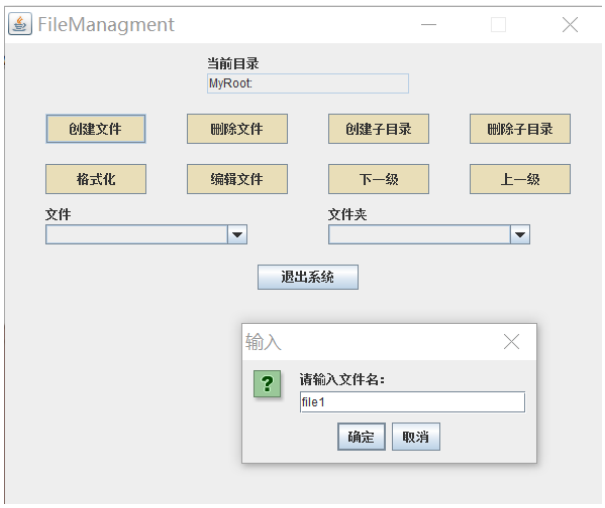
```
public void formatAll() {
    for(int i = 0; i < Utility.NUM_OF_DATABLOCK; i++) {
        fatTable[i] = Utility.FREE_FOR_FAT;
    }
    //初始化根目录,利用前两块，  .代表当前文件夹  ..代表上级文件夹
    fatTable[0] = 2;
    clearBlock(0);
    disc[0][Utility.POS_NAME] = '.';
    disc[0][Utility.POS_TYPE] = 0;
    disc[0][Utility.POS_FAT] = 0;
    disc[0][Utility.DIR_LEN] = '.';
    disc[0][Utility.DIR_LEN + 1] = '.';
    disc[0][Utility.DIR_LEN + Utility.LEN_OF_NAME] = 2;
}
```

6. 项目测试

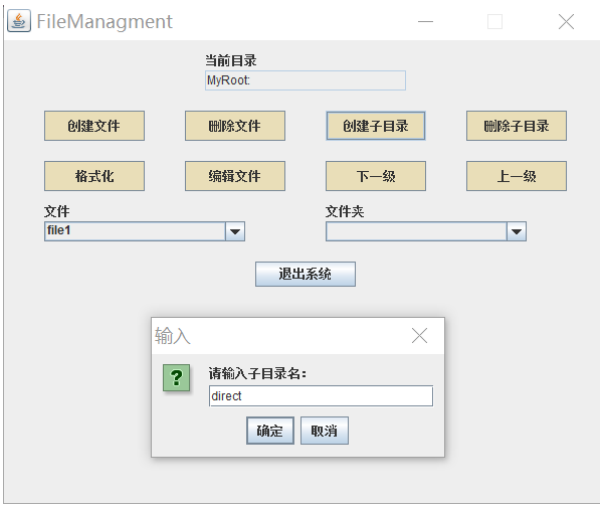
6.1 界面介绍

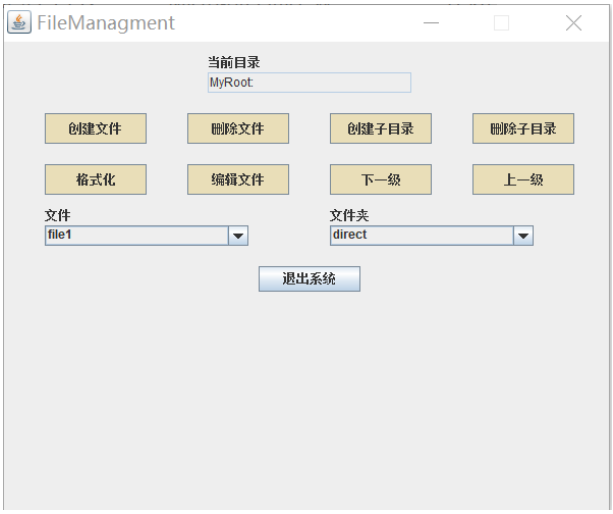


6.2 创建新文件

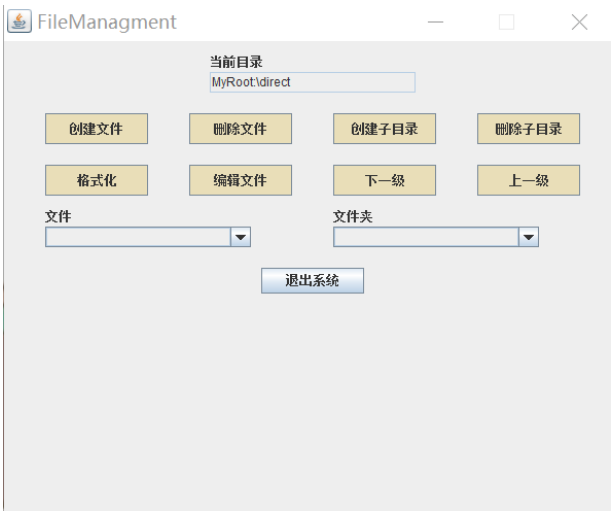


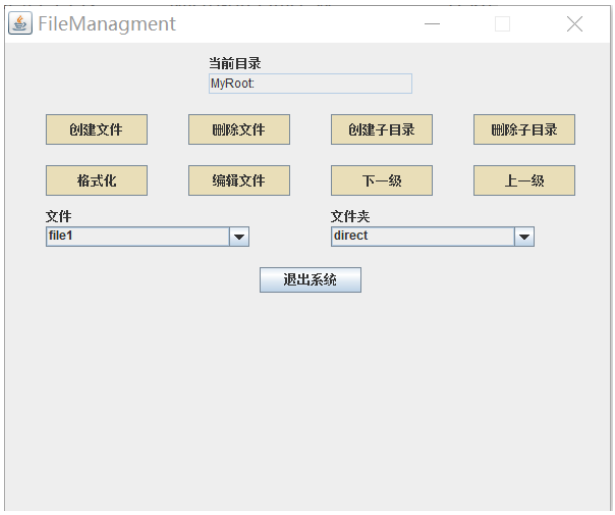
6.3 创建新子目录



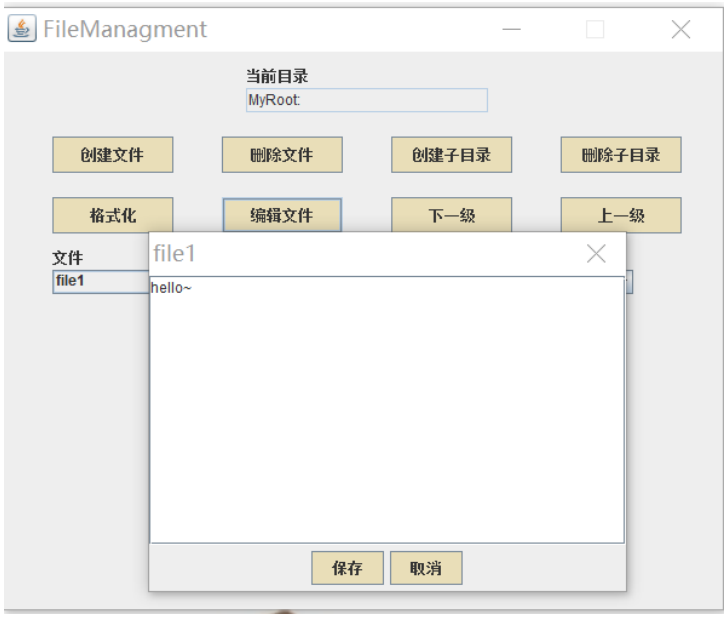


6.4 上下级目录切换





6.5 编辑文件



6.6 删除文件及子目录