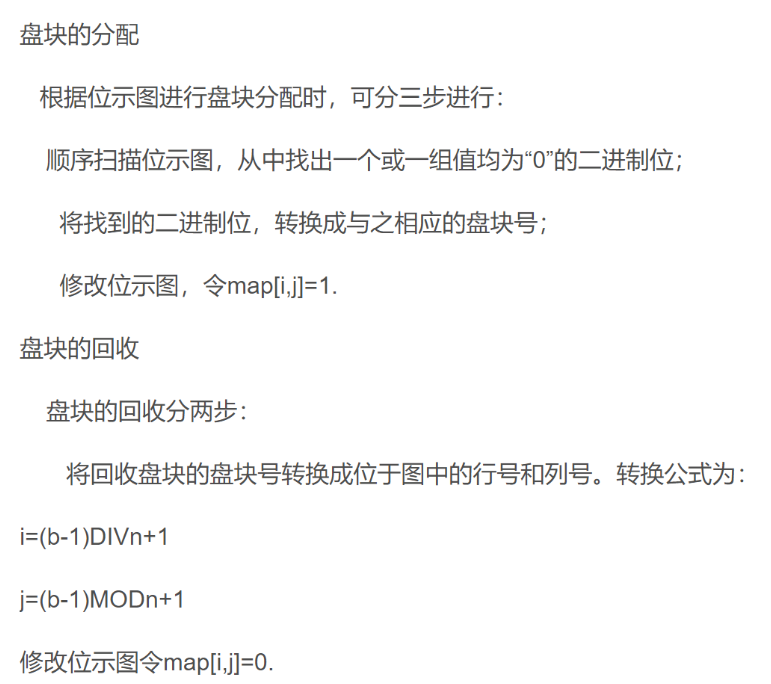
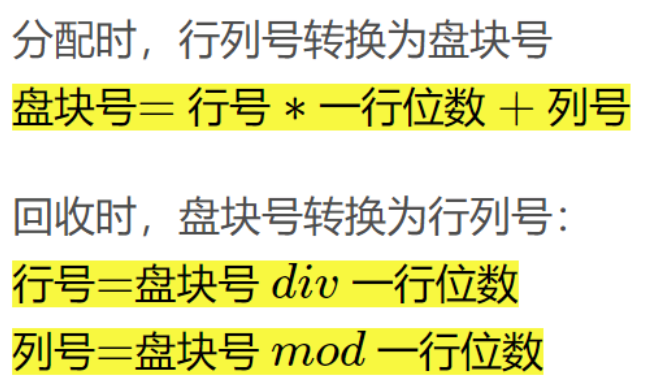
# 参考

[Java遍历一个目录下的所有文件 - 博客园 (cnblogs.com)](https://www.cnblogs.com/pejsidney/p/9084616.html)





# 学习心得

## 常量……

一些定义好的不变常量可以单独放到一个类里，作为static量展示。

应答的字符串也可以做为常量吗！！

## 注释的规范用法……

啊啊啊！！！为什么以前我不是这么写的？？？

# 代码

## 最终版本

xcopydir不支持同名重命名

package daily;  
import java.io.\*;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.time.LocalDateTime;  
import java.util.\*;  
  
public class FileSystem  
{  
 public static void main(String[] args) throws IOException, CloneNotSupportedException  
 {  
 //在D盘建立父文件夹  
 String filepath = "D:\\root";  
 Path path = Paths.get(filepath);  
 Files.createDirectory(path);  
 //一直注册，直到用户要login为止  
 Welcome.loginGuide();  
 //执行真正的文件操作  
 Welcome.Work();  
 }  
}  
class Welcome  
{  
 /\*\*  
 \* 当前的用户  
 \*/  
 private static User loginUser;  
  
 /\*\*  
 \* 登录引导界面  
 \*/  
 public static void loginGuide() throws IOException  
 {  
 boolean flag = true;  
 while(flag)  
 {  
 System.out.println("欢迎使用模拟文件操作系统!请根据提示输入指令");  
 System.out.println("quit ---- 退出系统");  
 System.out.println("register---- 注册新用户");  
 System.out.println("login ---- 用户登录");  
 System.out.print("请输入指令：");  
 Scanner scanner = new Scanner(System.in);  
 String s = scanner.nextLine();  
 switch (s)  
 {  
 case "quit" ->{  
 System.out.println("感谢您的使用！");  
 flag = false;  
 }  
 case "register" -> {  
 User user = new User();  
 String name, password;  
 System.out.print("用户名：");  
 name = scanner.nextLine();  
 //用户名查重  
 while (!User.isRepeat(name)) {  
 System.out.println("用户名重复！请重新输入用户名！");  
 System.out.print("用户名：");  
 name = scanner.nextLine();  
 }  
 user.setName(name);  
  
 System.out.print("密码：");  
 password = scanner.nextLine();  
 user.setPassword(password);  
  
 System.out.println("注册成功！可以登录了！");  
 User.addUser(user);  
 //return user;  
 }  
 case "login" -> {  
 String name, password;  
 if (User.allUser.size() == 0) {  
 System.out.println("当前还没有用户！请先去注册！");  
 }  
 else {  
 //检查用户名是否存在  
 System.out.print("用户名：");  
 name = scanner.nextLine();  
 while (!User.isRegistered(name)) {  
 System.out.println("用户" + name + "不存在！");  
 System.out.print("用户名：");  
 name = scanner.nextLine();  
 }  
 //检查密码是否正确  
 System.out.print("密码：");  
 password = scanner.nextLine();  
 while (!User.checkPwd(name, password)) {  
 System.out.println("密码错误！请重新输入！");  
 System.out.print("密码：");  
 password = scanner.nextLine();  
 }  
 System.out.println("登录成功！");  
  
 //找到对应的user  
 loginUser = User.getUser(name);  
 //新建其对应的文件夹  
 assert loginUser != null;  
 String filepath = "D:\\root" + FileControlBlock.separator + loginUser.getName();  
 Path path = Paths.get(filepath);  
 Files.createDirectory(path);  
  
 //更新位示图和FAT  
 int startNum = Disk.setBitmap(loginUser.getUserRoot().getBlockNum());  
 //通过位示图的返回值确定好起始块位置后，初始化该FCB  
 loginUser.setFCB(true, filepath, name, startNum, 1);  
 //加到Disk.totalFiles里  
 Disk.totalFiles.put(filepath, loginUser.getUserRoot());  
 flag = false;  
 }  
 }  
 default -> {  
 }  
 }  
 }  
  
 }  
 /\*\*  
 \* 命令菜单  
 \*/  
 public static void welcomeMenu()  
 {  
 System.out.println("quit ---- 退出系统");  
 System.out.println("help ---- 显示所有指令");  
 System.out.println("dir XXX -- 显示XXX路径的文件目录");  
 System.out.println("treedir XXX -- 循环列出XXX的子子孙孙目录和文件形式，并以树形显示");  
 System.out.println("mkdir XXX ---- 在当前目录下创建XXX的目录");  
 System.out.println("deldir XXX -- 删除XXX的空目录，若不为空则提醒");  
 System.out.println("xcopydir XXX YYY-- 给定目录XXX，将它连同其子子孙孙复制到给定的路径YYY下");  
 System.out.println("cd XXX-- 切换给定路径为XXX，若为文件名则提醒出错");  
 System.out.println("create XXX YYY -- 创建路径XXX下名为YYY的文件");  
 System.out.println("open/close XXX YYY -- 打开/关闭XXX下名为YYY的文件，供下面的read/write操作调用");  
 System.out.println("read XXX YYY-- 读路径XXX下名为YYY的文件，支持通过光标详细浏览文件");  
 System.out.println("write XXX YYY-- 写路径XXX下名为YYY的文件，追加模式下写入文件内容");  
 System.out.println("copy YYY XXX-- 将名为YYY的文件复制到路径XXX下");  
 System.out.println("import XXX YYY -- 将windows 路径XXX下的YYY文件导入到该系统当前目录");  
 System.out.println("export YYY XXX -- 将该系统当前目录下的YYY文件导出到本地XXX路径");  
 System.out.println("show -- 打印位示图");  
 }  
  
 /\*\*  
 \* 获取指定路径的FCB  
 \* @param path 路径  
 \* @return 路径对应的 FCB  
 \*/  
 public static FileControlBlock searchFCB(String path)  
 {  
 for (Map.Entry<String, FileControlBlock> entry : Disk.totalFiles.entrySet())  
 {  
 if (entry.getKey().equals(path)){  
 return entry.getValue();  
 }  
 }  
 return null;  
 }  
  
 /\*\*  
 \* 获取新FCB  
 \* @param fatherFCB 要找的目录FCB  
 \* @param name 子FCB名字  
 \* @return 子FCB  
 \*/  
 public static FileControlBlock searchSubFCB(FileControlBlock fatherFCB, String name)  
 {  
 for (Map.Entry<String, FileControlBlock> entry : fatherFCB.getSubDir().entrySet())  
 {  
 if (entry.getKey().equals(name)) {  
 return entry.getValue();  
 }  
 }  
 return null;  
 }  
 public static void Work() throws IOException, CloneNotSupportedException  
 {  
 /\*当前的路径\*/  
 String nowPath = loginUser.getUserRoot().getPath();  
 System.out.println("欢迎使用模拟文件操作系统 ! ");  
 System.out.print("minFos> ");  
 Scanner scanner = new Scanner(System.in);  
 welcomeMenu();  
  
 label:  
 while(true)  
 {  
 System.out.print("minFos> ");  
 String s = scanner.nextLine();  
 //把这句字符串整理成字符串数组  
 String[] text = s.split(" ");  
 //退出  
 switch (text[0].toLowerCase()) {  
 case "quit":  
 System.out.println("感谢您的使用！");  
 break label;  
  
 //菜单  
 case "help":  
 welcomeMenu();  
 break;  
 //在当前目录下创建XXX的目录  
 case "mkdir":  
 if (text.length < 2) {  
 System.out.println("文件夹名字不能为空！");  
 } else {  
 //查重，重名文件夹不行  
 FileControlBlock fatherFCB = null;  
 //获取老FCB  
 for (Map.Entry<String, FileControlBlock> entry : Disk.totalFiles.entrySet()) {  
 if (entry.getKey().equals(nowPath)) {  
 fatherFCB = entry.getValue();  
 break;  
 }  
 }  
 assert fatherFCB != null;  
 //获取老FCB的子集的名字的集合  
 Set<String> names = fatherFCB.getSubDir().keySet();  
 if (names.contains(text[1])) {  
 System.out.println("文件夹名字与该目录下其他目录名重复了！");  
 }  
 //没重名  
 else {  
 //获取当前目录的绝对路径  
 String path = nowPath + FileControlBlock.separator + text[1];  
 //新建FCB,一边做好FAT和位示图一边就返回FAT起始位置了  
 int startBlock = Disk.setBitmap(1);  
 FileControlBlock fileControlBlock = new FileControlBlock(true, path, text[1], startBlock);  
 //新FCB加到老集合中  
 fatherFCB.setSubDir(text[1], fileControlBlock);  
 //把新FCB加到所有FCB集合中  
 Disk.totalFiles.put(path, fileControlBlock);  
 //真的新建一个文件夹  
 Path pth = Paths.get(path);  
 Files.createDirectory(pth);  
 System.out.println("已在" + nowPath + "下创建名为" + text[1] + "的目录！");  
 }  
 }  
 break;  
 //给定目录XXX，将它连同其子子孙孙复制到给定路径YYY下  
 case "xcopydir":  
 if (text.length < 3) {  
 System.out.println("指令的参数太少了！不知道要做什么！");  
 } else {  
 String fromDir = text[1];  
 String toDir = text[2];  
 //找到该路径对应的FCB  
 FileControlBlock fromFCB = null, toFCB = null;  
 //获取老FCB  
 for (Map.Entry<String, FileControlBlock> entry : Disk.totalFiles.entrySet()) {  
 if (entry.getKey().equals(fromDir)) {  
 fromFCB = entry.getValue();  
 }  
 if (entry.getKey().equals(toDir)) {  
 toFCB = entry.getValue();  
 }  
 }  
 if (toFCB == null) {  
 System.out.println("给定的路径是空的！不知道要复制到哪里去！");  
 }  
 if (fromFCB == null) {  
 System.out.println("给定的目录是空的！不知道要从哪里去找要复制的文件！");  
 }  
 //是文件  
 else if (!fromFCB.getDirectory()) {  
 System.out.println(fromDir + "是一个文件，不是一个目录！");  
 }  
 //好了，鲁棒性检查结束了  
 else if (toFCB != null) {  
 //如果重名的话不能被copy  
 Set<String> toName = toFCB.getSubDir().keySet();  
 //有重名文件  
 if (toName.contains(fromFCB.getFileName())) {  
 System.out.println("路径下已有重名文件!");  
 }  
 else {  
 toFCB.setSubDir(fromFCB.getFileName(), fromFCB);  
 int depth = 1;  
 //输出用  
 xCopyDir(fromDir, depth, toFCB, toFCB.getFileName());  
 //真的这么做  
 copyFile(new File(fromDir), toDir, false, "anyName");  
 }  
  
 }  
 }  
 break;  
 case "import":  
 {  
 // XXX:D:\\  
 // YYY:a.txt  
 //filePath = D:\\a.txt  
 String fileName = text[2];  
 String filePath = text[1] + text[2];  
  
 File file = new File("d:\\" + text[2]);  
 if (!file.exists()) {  
 System.out.println(filePath + "不存在！");  
 } else {  
 int startBlock = Disk.setBitmap((int) (file.length() / 1024 / 8));  
 FileControlBlock fileFCB = new FileControlBlock(false, filePath, fileName, startBlock);  
 FileControlBlock fatherFCB = null;  
 //获取老FCB  
 for (Map.Entry<String, FileControlBlock> entry : Disk.totalFiles.entrySet()) {  
 if (entry.getKey().equals(nowPath)) {  
 fatherFCB = entry.getValue();  
 break;  
 }  
 }  
 //读取文件内容,加到自己的文件内容数组里  
 try (Scanner sc = new Scanner(new FileReader(filePath))) {  
 //按行读取  
 while (sc.hasNextLine()) {  
 String line = sc.nextLine();  
 fileFCB.setContent(fileFCB.getContent().append(line));  
 }  
 }  
 assert fatherFCB != null;  
 fatherFCB.setSubDir(fileName, fileFCB);  
 //不复制了!  
 System.out.println("已将 " + text[1] + " 下的文件 " + fileName + " 加入虚拟内存中！");  
 }  
 break;  
 }  
 //导出  
 case "export": {  
 // YYY: a.txt XXX:路径  
 //要导出的路径  
 String name = text[1];  
 FileControlBlock fatherFCB = searchFCB(nowPath);  
 FileControlBlock fileFCB = null;  
 //获取文件FCB  
 assert fatherFCB != null;  
 for (Map.Entry<String, FileControlBlock> entry : fatherFCB.getSubDir().entrySet()) {  
 if (entry.getValue().getFileName().equals(name)) {  
 fileFCB = entry.getValue();  
 break;  
 }  
 }  
 if (fileFCB == null || fileFCB.getDirectory()) {  
 System.out.println("该目录下没有名叫 " + name + " 的文件！");  
 } else {  
 String filePath = text[2] + text[1];  
 File file = new File(filePath);  
 if (!file.exists()) {  
 try {  
 file.createNewFile();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 System.out.println("文件已导出!");  
 } else {  
 System.out.println("文件已存在!");  
 }  
 //向文件中写入内容  
 try (FileWriter fw = new FileWriter(filePath, true)) {  
 fw.append(fileFCB.getContent());  
 }  
 }  
 break;  
 }  
 //打开文件  
 case "open":  
 if (text.length < 3) {  
 System.out.println("指令信息不全！不知道要打开什么！");  
 } else {  
 String path = text[1];  
 String fileName = text[2];  
 //找到该路径对应的FCB  
 FileControlBlock fatherFCB = searchFCB(path), sonFCB;  
 if (fatherFCB == null) {  
 System.out.println("路径不对！");  
 } else {  
 //获取新FCB  
 sonFCB = searchSubFCB(fatherFCB, fileName);  
 if (sonFCB == null) {  
 System.out.println(path + " 路径下没有名叫 " + fileName + " 的文件！");  
 } else if (sonFCB.getDirectory()) {  
 System.out.println("你要打开的" + fileName + "是一个目录！");  
 }  
 //路径正确，且路径下存在该 文件  
 else {  
 sonFCB.setIsOpened(true);  
 System.out.println("文件已打开，可以向里面写东西~");  
 }  
 }  
 }  
 break;  
 //关闭文件  
 case "close":  
 if (text.length < 3) {  
 System.out.println("指令信息不全！不知道要关闭什么！");  
 } else {  
 String path = text[1];  
 String fileName = text[2];  
 //找到该路径对应的FCB  
 FileControlBlock fatherFCB = searchFCB(path), sonFCB = null;  
 if (fatherFCB == null) {  
 System.out.println("路径不对！");  
 } else {  
 //获取新FCB  
 for (Map.Entry<String, FileControlBlock> entry : fatherFCB.getSubDir().entrySet()) {  
 if (entry.getKey().equals(fileName)) {  
 sonFCB = entry.getValue();  
 break;  
 }  
 }  
 if (sonFCB == null) {  
 System.out.println(path + " 路径下没有名叫 " + fileName + " 的文件！");  
 } else if (sonFCB.getDirectory()) {  
 System.out.println("你要关闭的" + fileName + "是一个目录！");  
 }  
 //路径正确，且路径下存在该 文件  
 else {  
 sonFCB.setIsOpened(false);  
 System.out.println("文件已关闭！~");  
 }  
 }  
 }  
 break;  
 case "read":  
 if (text.length < 3) {  
 System.out.println("指令信息不全！不知道要读什么！");  
 } else {  
 String path = text[1];  
 String fileName = text[2];  
 //找到该路径对应的FCB  
 FileControlBlock fatherFCB = null, sonFCB = null;  
 //获取老FCB  
 for (Map.Entry<String, FileControlBlock> entry : Disk.totalFiles.entrySet()) {  
 if (entry.getKey().equals(path)) {  
 fatherFCB = entry.getValue();  
 break;  
 }  
 }  
 if (fatherFCB == null) {  
 System.out.println("路径不对！");  
 } else {  
 //获取新FCB  
 for (Map.Entry<String, FileControlBlock> entry : fatherFCB.getSubDir().entrySet()) {  
 if (entry.getKey().equals(fileName)) {  
 sonFCB = entry.getValue();  
 break;  
 }  
 }  
 if (sonFCB == null) {  
 System.out.println(path + " 路径下没有名叫 " + fileName + " 的文件！");  
 } else if (sonFCB.getDirectory()) {  
 System.out.println("你要读的" + fileName + "是一个目录！");  
 }  
 //路径正确，且路径下存在该 文件  
 else {  
 if (!sonFCB.getIsOpened()) {  
 System.out.println("你还没有打开文件！不能进行读操作！");  
 } else {  
 System.out.println("文件内容如下：");  
 System.out.println(sonFCB.getContent());  
 }  
 }  
 }  
 }  
 break;  
 case "write":  
 if (text.length < 3) {  
 System.out.println("指令信息不全！不知道要写入什么！");  
 } else {  
 String path = text[1];  
 String fileName = text[2];  
 //找到该路径对应的FCB  
 FileControlBlock fatherFCB = searchFCB(path), sonFCB;  
 if (fatherFCB == null) {  
 System.out.println("路径不对！");  
 } else {  
 sonFCB = searchSubFCB(fatherFCB, fileName);  
  
 if (sonFCB == null) {  
 System.out.println(path + " 路径下没有名叫 " + fileName + " 的文件！");  
 } else if (sonFCB.getDirectory()) {  
 System.out.println("你要写入的" + fileName + "是一个目录！");  
 }  
 //路径正确，且路径下存在该 文件  
 else {  
 if (!sonFCB.getIsOpened()) {  
 System.out.println("你还没有打开文件！不能进行写操作！");  
 } else {  
 System.out.print("请输入你希望追加的内容：");  
 String addContent = scanner.nextLine();  
 sonFCB.setContent(sonFCB.getContent().append(addContent));  
 System.out.print("写入成功，现在文件的内容是：");  
 System.out.println(sonFCB.getContent());  
 //更新文件最后修改时间  
 sonFCB.setUpdateTime(LocalDateTime.now());  
 //真的写  
 try (FileWriter fw = new FileWriter(sonFCB.getPath(), true)) {  
 fw.append(sonFCB.getContent());  
 }  
 }  
 }  
 }  
 }  
 break;  
 //copy YYY XXX 将 名为 YYY的文件复制到路径XXX下  
 case "copy": {  
 String fileName = text[1];  
 String path = text[2];  
 //找到该路径对应的FCB  
 FileControlBlock fileFCB = null, pathFCB = null;  
 //获取fileFCB  
 for (Map.Entry<String, FileControlBlock> entry : Disk.totalFiles.entrySet()) {  
 if (entry.getValue().getFileName().equals(fileName)) {  
 fileFCB = entry.getValue();  
 break;  
 }  
 }  
 if (fileFCB == null) {  
 System.out.println("文件不存在！");  
 } else {  
 //获取pathFCB  
 for (Map.Entry<String, FileControlBlock> entry : Disk.totalFiles.entrySet()) {  
 if (entry.getKey().equals(path)) {  
 pathFCB = entry.getValue();  
 break;  
 }  
 }  
 if (pathFCB == null) {  
 System.out.println("路径错误或不存在！");  
 }  
 //文件存在，路径也存在  
 else {  
 //名称查重  
 Set<String> toName = pathFCB.getSubDir().keySet();  
 //有重名文件  
 if (toName.contains(pathFCB.getFileName())) {  
 System.out.println("路径下已有重名文件，是要重命名还是放弃复制？（x/y）：");  
 String ans = scanner.nextLine();  
 if (ans.equals("x")) {  
 System.out.print("那么新的文件（夹）名是：");  
 ans = scanner.nextLine();  
 pathFCB.setSubDir(ans, fileFCB);  
 System.out.println("文件 " + ans + " 已被复制到 " + path + " 下！");  
 //真复制  
 copyFile(new File(fileFCB.getPath()), path, true, ans);  
  
 } else if (ans.equals("y")) {  
 System.out.println("你退出了该指令！");  
 }  
 }  
 //没有重名文件，放心复制即可  
 else {  
 pathFCB.setSubDir(fileFCB.getFileName(), fileFCB);  
 System.out.println("文件 " + fileFCB.getFileName() + " 已被复制到 " + path + " 下！");  
 //真复制  
 copyFile(new File(fileFCB.getPath()), path, false, fileFCB.getFileName());  
 }  
 }  
 }  
 break;  
 }  
 //切换给定路径XXX为当前目录，如果是文件名就提醒出错  
 case "cd": {  
 if(text.length == 1){  
 System.out.println("仍在当前目录下！");  
 break;  
 }  
 String path = text[1];  
 //找到该路径对应的FCB  
 FileControlBlock fatherFCB = searchFCB(path);  
  
 if (fatherFCB == null) {  
 System.out.println("没有这个目录！");  
 } else {  
 //是不是文件名  
 if (!fatherFCB.getDirectory()) {  
 System.out.println("这个路径代表的是文件！不能这样做！");  
 } else {  
 nowPath = path;  
 System.out.println("已切换当前路径至" + nowPath + "!");  
 }  
 }  
 break;  
 }  
 //删空目录，或提醒  
 case "deldir":  
 if (text.length < 2) {  
 System.out.println("你还没有输入要删除的目录呢！");  
 } else {  
 //绝对路径  
 String path = text[1];  
 //获取指定路径的FCB  
 FileControlBlock fatherFCB = searchFCB(path);  
  
 if (fatherFCB == null) {  
 System.out.println("该目录不存在！");  
 } else {  
 //得到subFCB  
 Map<String, FileControlBlock> mp = fatherFCB.getSubDir();  
 //处理FAT， 位示图  
 for (Map.Entry<String, FileControlBlock> entry : mp.entrySet()) {  
 FileControlBlock mapValue = entry.getValue();  
 //处理空文件夹的FAT， 位示图  
 if (mapValue.getSubDir().size() == 0) {  
 Disk.returnBitmap(mapValue.getStartBlock());  
 }  
 }  
 //自己，全局都删空目录  
 Disk.totalFiles.entrySet().removeIf(entry -> entry.getValue().getSubDir().size() == 0 && entry.getKey().contains(text[1]));  
 mp.values().removeIf(mapValue -> mapValue.getSubDir().size() == 0);  
  
 System.out.println("删除信息：");  
 //删除（物理）  
 FileUtils m = new FileUtils();  
 List<File> list = m.visitAll(new File(text[1]));  
 for (File temp : list) {  
 // 是目录且为空  
 if (temp.isDirectory() && Objects.requireNonNull(temp.listFiles()).length <= 0) {  
 temp.delete();  
 }  
 }  
 System.out.println("删除成功！");  
 //提醒  
 for (Map.Entry<String, FileControlBlock> entry : mp.entrySet()) {  
 FileControlBlock mapValue = entry.getValue();  
 System.out.println(mapValue.getFileName() + "内容不为空！");  
 }  
 }  
 }  
 break;  
 case "dir": {  
 //绝对路径  
 String path = text[1];  
 //获取指定路径的FCB  
 FileControlBlock fatherFCB = null;  
 for (Map.Entry<String, FileControlBlock> entry : Disk.totalFiles.entrySet()) {  
 if (entry.getKey().equals(path)) {  
 fatherFCB = entry.getValue();  
 break;  
 }  
 }  
 if (fatherFCB == null) {  
 System.out.println("该目录不存在！");  
 } else {  
 System.out.println("文件名 文件类型 文件大小 创建时间 最后更新时间");  
 Map<String, FileControlBlock> mp = fatherFCB.getSubDir();  
 for (Map.Entry<String, FileControlBlock> entry : mp.entrySet()) {  
 FileControlBlock mapValue = entry.getValue();  
 System.out.println(mapValue);  
 }  
 }  
 break;  
 }  
 //create XXX YYY  
 //创建指定路径下，给定文件名的文件  
 case "create":  
 //指令合法性  
 if (text.length < 3) {  
 System.out.println("指令长度不够！");  
 } else {  
 String dirPath = text[1];  
 String fileName = text[2];  
 //找到指定路径的FCB  
 FileControlBlock fatherFCB = null;  
 for (Map.Entry<String, FileControlBlock> entry : Disk.totalFiles.entrySet()) {  
 if (entry.getKey().equals(dirPath)) {  
 fatherFCB = entry.getValue();  
 break;  
 }  
 }  
 if (fatherFCB == null) {  
 System.out.println("这个路径不存在！");  
 }  
 //合法路径  
 else {  
 //新建FCB,一边做好FAT和位示图一边就返回FAT起始位置了  
 System.out.print("请输入文件所占的块数：");  
 String sLen = scanner.nextLine().trim();  
 int len = Integer.parseInt(sLen);  
 int startBlock = Disk.setBitmap(len);  
 String filepath = dirPath + FileControlBlock.separator;  
 FileControlBlock fileControlBlock = new FileControlBlock(false, filepath + fileName, fileName, startBlock);  
 //新FCB加到老集合中  
 fatherFCB.setSubDir(fileName, fileControlBlock);  
 //把新FCB加到所有FCB集合中  
 Disk.totalFiles.put(filepath + fileName, fileControlBlock);  
 //真的新建一个文件  
 File file = new File(filepath + fileName);  
 // 返回true表示文件成功,false 表示文件已经存在  
 if (file.createNewFile()) {  
 System.out.println("已在" + filepath + "下创建名为" + fileName + "的文件！");  
 } else {  
 System.out.println("文件已经存在, 不需要重复创建");  
 }  
 }  
 }  
 break;  
 case "treedir":  
 int depth = 1;  
 treeOut(text[1], depth);  
 break;  
 //展示位示图  
 case "show":  
 Disk.show();  
 break;  
 //指令错误  
 default:  
 System.out.println("指令错误，不能识别该指令！");  
 break;  
 }  
 }  
 }  
 /\*\*  
 \* 批量复制文件夹  
 \* @param source 原文件  
 \* @param dest 目的地地址  
 \*/  
 public static void copyFile(File source, String dest, boolean isRename, String newName)throws IOException  
 {  
// //创建目的地文件夹  
// File destFile = new File(dest);  
// if(!destFile.exists()){  
// destFile.mkdir();  
// }  
 //如果source是文件夹，则在目的地址中创建新的文件夹  
 if(source.isDirectory())  
 {  
 File file;  
 if (isRename){  
 file = new File(dest+"\\"+newName);  
 }  
 else {  
 //用目的地址加上source的文件夹名称，创建新的文件夹  
 file = new File(dest+"\\"+source.getName());  
 }  
 file.mkdir();  
 //得到source文件夹的所有文件及目录  
 File[] files = source.listFiles();  
 assert files != null;  
 if(files.length!=0)  
 {  
 for (File value : files) {  
 copyFile(value, file.getPath(),isRename, newName);  
 }  
 }  
 }  
 //source是文件，则用字节输入输出流复制文件  
 else if(source.isFile())  
 {  
 FileInputStream fis = new FileInputStream(source);  
 //创建新的文件，保存复制内容，文件名称与源文件名称一致  
 File dfile = new File(dest+"\\"+source.getName());  
 if(!dfile.exists()){  
 dfile.createNewFile();  
 }  
  
 FileOutputStream fos = new FileOutputStream(dfile);  
 // 读写数据  
 // 定义数组  
 byte[] b = new byte[1024];  
 // 定义长度  
 int len;  
 // 循环读取  
 while ((len = fis.read(b))!=-1) {  
 // 写出数据  
 fos.write(b, 0 , len);  
 }  
  
 //关闭资源  
 fos.close();  
 fis.close();  
 }  
 }  
 /\*\*  
 \* 得到某目录下所有文件夹  
 \*/  
 public static class FileUtils  
 {  
 List<File> list = new ArrayList<>();  
  
 // 得到某一目录下的所有文件夹  
 public List<File> visitAll(File root)  
 {  
 File[] dirs = root.listFiles();  
 if (dirs != null)  
 {  
 for (File dir : dirs)  
 {  
 if (dir.isDirectory())  
 {  
 System.out.println("name:" + dir.getPath());  
 list.add(dir);  
 }  
 visitAll(dir);  
 }  
 }  
 return list;  
 }  
 }  
 /\*\*  
 \* 树形输出子子孙孙目录  
 \* @param pathName 绝对路径名  
 \* @param depth 当前深度  
 \*/  
 public static void treeOut(String pathName,int depth)  
 {  
 FileControlBlock fatherFCB = null;  
 //获取老FCB  
 for (Map.Entry<String, FileControlBlock> entry : Disk.totalFiles.entrySet())  
 {  
 if (entry.getKey().equals(pathName))  
 {  
 fatherFCB= entry.getValue();  
 break;  
 }  
 }  
 if (fatherFCB == null)  
 {  
 System.out.println("该目录不存在！");  
 return;  
 }  
 //如果是一个文件则输出文件名  
 if (!fatherFCB.getDirectory())  
 {  
 System.out.println(fatherFCB.getFileName());  
 return;  
 }  
 for (int j = 0; j < depth; j++) {  
 System.out.print(" ");  
 }  
 System.out.print("|--");  
 System.out.println(fatherFCB.getFileName());  
 //获取此目录下的所有文件名与目录名  
 Map<String, FileControlBlock> subDir = fatherFCB.getSubDir();  
 int currentDepth=depth+1;  
 for (Map.Entry<String, FileControlBlock> entry : subDir.entrySet())  
 {  
 String name = entry.getKey();  
 FileControlBlock mapValue = entry.getValue();  
 if (mapValue.getDirectory())  
 {  
 //递归  
 treeOut(mapValue.getPath(), currentDepth);  
 }  
 else  
 {  
 //如果是文件，则直接输出文件名  
 for (int j = 0; j < currentDepth; j++) {  
 System.out.print(" ");  
 }  
 System.out.print("|--");  
 System.out.println(name);  
 }  
 }  
 }  
 /\*\*  
 \* 导入子子孙孙目录，xcopydir  
 \* @param pathName fromFCB绝对路径名  
 \* @param depth 当前深度  
 \*/  
 public static void xCopyDir(String pathName, int depth, FileControlBlock toFCB, String newName)  
 {  
 FileControlBlock fatherFCB = null;  
 //获取老FCB  
 for (Map.Entry<String, FileControlBlock> entry : Disk.totalFiles.entrySet())  
 {  
 if (entry.getKey().equals(pathName))  
 {  
 fatherFCB= entry.getValue();  
 break;  
 }  
 }  
 if (fatherFCB == null)  
 {  
 System.out.println("该目录不存在！");  
 return;  
 }  
 else  
 {  
 toFCB.setPath(toFCB.getPath());  
 System.out.println("已将目录" + fatherFCB.getFileName() + "加入目标路径 " + toFCB.getPath() +" 中！");  
 }  
 //是文件  
 if (!fatherFCB.getDirectory())  
 {  
 toFCB.setPath(toFCB.getPath());  
 System.out.println("已将 " + fatherFCB.getPath() +"下的文件" + fatherFCB.getFileName() + "加入目标路径 " + toFCB.getPath() +" 中！");  
 return;  
 }  
 //获取此目录下的所有文件名与目录名  
 Map<String, FileControlBlock> subDir = fatherFCB.getSubDir();  
 int currentDepth=depth+1;  
 for (Map.Entry<String, FileControlBlock> entry : subDir.entrySet())  
 {  
 //获取键值对  
 String name = entry.getKey();  
 FileControlBlock mapValue = entry.getValue();  
 //将该FCB加入目标目录下  
 if (mapValue.getDirectory())  
 {  
 //递归  
 treeOut(mapValue.getPath(), currentDepth);  
 }  
 }  
 }  
}  
class Disk  
{  
 /\*\*  
 \* 该磁盘上的全部文件及其对应的FCB目录项  
 \* String : filePath  
 \*/  
 public static Map<String, FileControlBlock> totalFiles = new HashMap<>();  
  
 /\*\*  
 \* 还剩下几个块能用  
 \*/  
 private static int restBlockNum = 8192;  
 /\*\*  
 \* FAT表,64MB / 8KB = 8192（块）  
 \* FAT[0]设为根目录的空间,  
 \* FAT[1]记录磁盘剩余块数  
 \*/  
 private static int[] FAT = new int[restBlockNum];  
 /\*\*  
 \* 位示图的行列数  
 \*/  
 private static final int ROW = 64;  
 private static final int COLUMN = 128;  
 /\*\*  
 \* 位示图  
 \*/  
 private static int[][] bitmap = new int[ROW][COLUMN];  
  
 /\*\*  
 \* 填充位示图和FAT  
 \* @param blockNum 这个文件大小是多少个块  
 \* @return 返回FAT起始块  
 \*/  
 public static int setBitmap(int blockNum)  
 {  
 int preLoc = 0, res = -1;  
 //顺序扫描位示图，从中找出一个值为0的位置  
 for (int i = 0; i < 64 && blockNum > 0; i++)  
 {  
 for (int j = 0; j < 128 && blockNum > 0; j++)  
 {  
 if (bitmap[i][j] == MAP\_FREE)  
 {  
 res = (res == -1 ? i \* 64 + j : res);  
 bitmap[i][j] = 1;  
 blockNum -= 1;  
 if (preLoc != 0) {  
 FAT[preLoc] = i \* 64 + j;  
 }  
 preLoc = i \* 64 + j;  
 }  
 }  
 }  
 //说明该文件/目录只占了一个内存块，此时到它自己就结束了  
 if (FAT[preLoc] == 0){  
 FAT[preLoc] = -1;  
 }  
 return res;  
 }  
  
 /\*\*  
 \* 删除文件或目录时，释放FAT和位示图的空间  
 \* @param startNum 这个FCB在FAT的起始地址  
 \*/  
 public static void returnBitmap(int startNum)  
 {  
 while (startNum != -1)  
 {  
 //回收时，盘块号转换为行列号  
 int row = startNum / ROW;  
 int col = startNum % COLUMN;  
 bitmap[row][col] = 0;  
 startNum = FAT[startNum];  
 }  
 }  
  
 /\*\*  
 \* 打印位示图  
 \*/  
 public static void show()  
 {  
 for (int i = 0; i < 64; i++)  
 {  
 for (int j = 0; j < 128; j++) {  
 System.out.print(bitmap[i][j] + "|");  
 }  
 System.out.println();  
 }  
 }  
  
 /\*\*  
 \* 1表示磁盘块已占用，0表示磁盘块为空.  
 \*/  
 private final static int MAP\_FULL = 1;  
 private final static int MAP\_FREE = 0;  
  
 /\*\*  
 \* 启动程序的初始构造函数  
 \*/  
 public Disk()  
 {  
 //可用块数-1  
 restBlockNum -= 1;  
 FAT[0] = restBlockNum;  
 bitmap[0][0] = MAP\_FULL;  
 }  
}  
class FileControlBlock  
{  
 /\*\*  
 \*路径分隔符  
 \*/  
 public static String separator = "\\";  
  
 /\*\*  
 \* 该FCB的子FCB集合  
 \* string：filename  
 \*/  
 private Map<String, FileControlBlock> subDir = new HashMap<>();  
  
 /\*\*  
 \* 是否是目录文件  
 \* true 是，false 不是  
 \*/  
 private boolean isDirectory;  
 /\*\*  
 \* 这个FCB的路径  
 \*/  
 private String path;  
 /\*\*  
 \* 若是txt文件类型，则用于追加  
 \*/  
 private StringBuffer content = new StringBuffer();  
 /\*\*  
 \* 文件类型是否被打开，默认否  
 \*/  
 private boolean isOpened = false;  
 /\*\*  
 \* 文件名（包括拓展名）  
 \*/  
 private String fileName;  
  
 /\*\*  
 \* 在FAT表中的起始盘块号  
 \*/  
 private int startBlock;  
  
 /\*\*  
 \* 一般文件用的盘块数  
 \*/  
 private int blockNum = 1;  
  
 /\*\*  
 \* 文件大小 = 一个盘块的大小(8) \* 所占用的盘块数  
 \*/  
 private int size;  
 /\*\*  
 \* 创建时间  
 \*/  
 private LocalDateTime createTime;  
  
 /\*\*  
 \* 最后修改时间  
 \*/  
 private LocalDateTime updateTime;  
 @Override  
 public Object clone() throws CloneNotSupportedException  
 {  
 return super.clone();  
 }  
 public boolean getIsOpened() {  
 return isOpened;  
 }  
  
 public void setIsOpened(boolean opened) {  
 isOpened = opened;  
 }  
  
 /\*\*  
 \* 把子目录放到集合中  
 \* @param name 子目录名  
 \* @param subFCB 子目录项  
 \*/  
 public void setSubDir(String name, FileControlBlock subFCB) {  
 subDir.put(name, subFCB);  
 }  
 public Map<String, FileControlBlock> getSubDir() {  
 return subDir;  
 }  
  
 public void setDirectory(boolean directory) {  
 isDirectory = directory;  
 }  
  
 public boolean getDirectory() {  
 return isDirectory;  
 }  
  
 public String getPath() {  
 return path;  
 }  
  
 public void setPath(String path) {  
 this.path = path;  
 }  
  
 public StringBuffer getContent() {  
 return content;  
 }  
  
 public void setContent(StringBuffer content) {  
 this.content = content;  
 }  
  
 public String getFileName() {  
 return fileName;  
 }  
  
 public void setFileName(String fileName) {  
 this.fileName = fileName;  
 }  
  
 public int getStartBlock() {  
 return startBlock;  
 }  
  
 public void setStartBlock(int startBlock) {  
 this.startBlock = startBlock;  
 }  
  
 public int getBlockNum() {  
 return blockNum;  
 }  
  
 public void setBlockNum(int blockNum) {  
 this.blockNum = blockNum;  
 }  
  
 public void setSize(int size) {  
 this.size = size;  
 }  
  
 public void setCreateTime(LocalDateTime createTime) {  
 this.createTime = createTime;  
 }  
  
 public void setUpdateTime(LocalDateTime updateTime) {  
 this.updateTime = updateTime;  
 }  
  
 /\*\*  
 \* mkdir创建目录等的构造函数  
 \* @param isDirectory 是否是目录 true是，false不是  
 \* @param path 路径  
 \* @param fileName 目录名  
 \* @param startBlock FAT起始块  
 \*/  
 public FileControlBlock(boolean isDirectory,String path, String fileName, int startBlock) {  
 this.isDirectory = isDirectory;  
 this.path = path;  
 this.fileName = fileName;  
 this.startBlock = startBlock;  
 this.size = blockNum \* 8;  
 this.createTime = LocalDateTime.now();  
 this.updateTime = LocalDateTime.now();  
 }  
  
 //无参构造函数  
 public FileControlBlock() {}  
  
  
  
 /\*\*  
 \* 输出该FCB对应属性时使用  
 \* @return 字符串  
 \*/  
 @Override  
 public String toString() {  
 return fileName + " " +  
 (isDirectory ?"文件夹":"文件") +" " +  
 (isDirectory ?"8kb":(8 \* size + "kb")) +" " +  
 createTime + " " +  
 updateTime;  
 }  
}  
class User  
{  
 /\*\*  
 \* 用户持有自己的根目录  
 \*/  
 private FileControlBlock userRoot = new FileControlBlock();  
 /\*\*  
 \* 用户名，密码  
 \*/  
 private String name, password;  
 /\*\*  
 \* 每个用户及其对应的密码  
 \*/  
 private static HashMap<String, String> userInfo = new HashMap<>();  
 /\*\*  
 \* 所有用户信息  
 \*/  
 public static List<User> allUser = new ArrayList<>();  
  
 public FileControlBlock getUserRoot() {  
 return userRoot;  
 }  
  
 /\*\*  
 \* 通过名字获取对应的user，登录用  
 \* @param name：要获得的user  
 \* @return 名字对应的user  
 \*/  
 public static User getUser(String name)  
 {  
 for (User user : allUser) {  
 if (user.getName().equals(name)) {  
 return user;  
 }  
 }  
 return null;  
 }  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
 /\*\*  
 \* login时的设置函数  
 \* @param isDirectory 是否是文件  
 \* @param path 路径  
 \* @param fileName 名字，就是userName  
 \* @param startBlock 起始块数  
 \* @param blockNum 占多少个块  
 \*/  
 public void setFCB(boolean isDirectory, String path, String fileName, int startBlock, int blockNum)  
 {  
 userRoot.setPath(path);  
 userRoot.setFileName(fileName);  
 userRoot.setStartBlock(startBlock);  
 userRoot.setBlockNum(blockNum);  
 userRoot.setSize(blockNum \* 8);  
 userRoot.setCreateTime(LocalDateTime.now());  
 userRoot.setDirectory(isDirectory);  
 }  
 /\*\*  
 \* 检查该用户是否已经注册  
 \* @param name:登录用户名  
 \* @return true表示已经注册过，否则没有注册过  
 \*/  
 public static boolean isRegistered(String name)  
 {  
 for (User user : allUser)  
 {  
 if (user.getName().equals(name)) {  
 return true;  
 }  
 }  
 return false;  
 }  
 /\*\*  
 \* 检查登录的用户名与密码是否对得上  
 \*/  
 public static boolean checkPwd(String name, String password) {  
 return userInfo.get(name).equals(password);  
 }  
  
 /\*\*  
 \* 检查注册的用户名是否已经重名了  
 \* 有重名返回false  
 \*/  
 public static boolean isRepeat(String name)  
 {  
 //allName初始化始终在这里进行  
 //用户名集合，用于查重  
 Set<String> allName = userInfo.keySet();  
 for (String s : allName)  
 {  
 //有重名返回false  
 if (s.equals(name)){  
 return false;  
 }  
 }  
 return true;  
 }  
 /\*\*  
 \* 添加新用户  
 \*/  
 public static void addUser(User user)  
 {  
 //map里加入这个用户的登录信息  
 userInfo.put(user.getName(), user.getPassword());  
 //把这个user加入所有user集合中  
 allUser.add(user);  
 }  
}

# 成果

以后记得截图（吐血）