《Java 企业级应用》实验报告

年级、专业、班级	2021 级计卓 1 班	姓名	李宽宇	学号	20215279
实验题目	基于命令行的文件管理器				
实验时间	2024.3.30	实验地点	DS3401		
学年学期	2023-2024(2)	实验性质	□验证性 ■设计性 □综合性		

一、实验目的

1. 本次实验的目的是掌握 Java 企业级应用开发工具的使用方法，掌握 Java 面向对象程序编程技术，掌握常用工具类的使用。理解面向对象的分析和设计方法，理解和使用设计模式。
2. 设计开发完成一个基于命令行的文件资源管理软件。熟练掌握文件资源的处理方法和工具类。理解多线程工作机制。提高 OOP 能力。
3. 抄袭计 0 分。

二、实验项目内容

设计开发完成“基于命令行的文件管理器”软件。要求如下：

1、可以设置**当前工作文件夹**，默认在当前文件夹下进行文件的增删改查操作，文件夹的增删改查操作。也可以操作指定的文件夹。

2、实现当前文件夹下的内容罗列；可以根据文件名、文件大小、文件类型、文件日期等过滤特定类型的文件。罗列的时候可以排序，依据名称、大小、时间等。罗列文件的时候，显示要规范。

3、可以直接查看和显示一个文本文件的内容。

4、实现文件拷贝和文件夹拷贝（文件夹拷贝指深度拷贝，包括所有子目录和文件）；能指定目标名称和位置，能计算拷贝时间，能显示拷贝进度。

5、可以利用线程机制，支持后台长时任务异步执行。不影响前端操作。例如在拷贝大文件或者文件夹的时候，可以选择后台执行，还是前台执行。如果前台执行，显示时间和进度。

6、可以对指定文件进行加密和解密；加密后形成新的文件，可以指定文件名。加密与解密方式自己选择。

7、可以对文件或者文件夹进行压缩，或者解压。压缩与解压方式自己选择。

8、可以实现自定义功能。注意操作使用的方便性，注意类和类之间的关系。充分利用继承，多态等特性，使用上抽象类，接口，泛型，内部类等设计元素，使用好集合类、多线程、IO 工具类、NIO 工具类等。注意程序的总执行流程和分支执行流程。注意设计思想的表达，注意优化代码结构，优化

类的职责分工，注意使用设计模式。代码有注释。

9、在报告中注明自己的创新点、特色等。

10、提交：（1）本实验报告，（2）源代码压缩文件 zip，（3）软件演示的 MP4 视频，视频大小不超过 40M，视频请在**搜狗浏览器或者 QQ 浏览器**测试能否正常播放。注意源代码加注释。注意文件名称的规范性。文件名：学号姓名 2.docx，学号姓名 2.zip, 学号姓名 2.mp4。**三个文件分别提交**。

三、实验过程或算法（写明创新点或特色、设计思想、设计模式的使用、程序的结构、功能关系图、类的说明和类之间的关系图、程序主要执行流程图，最后是核心源代码，截图等）

1. 创新点或特色

(1) 设计模式方面，采用了单例模式（Singleton Pattern）、策略模式（Strategy Pattern），桥接（Bridge），所有类满足单一职责原则。做到了高内聚、低耦合

（2)重点使用 IO 工具类、NIO 工具类,如 File,Files,字节流和 channel, FileInputStream、FileChannel、ByteBuffer、BufferedReader、FileReader

（3）使用 callback,使得后台异步的线程结束后，返回提示信息到主线程。

（4）数据可视化

图 1 拷贝文件进度条

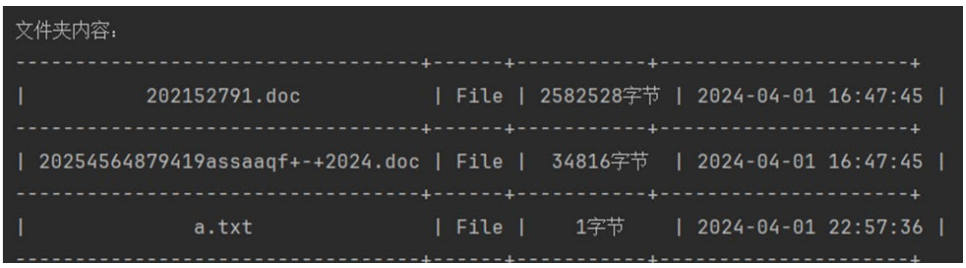

图 2 工作文件夹的文件列表

2. 设计思想

（1）程序设计重点使用 IO 工具类、NIO 工具类，采用了多种设计元素，包括但不限于继承、接口、匿名对象 Lamda 表达式；使用多种工具，包括集合框架 HashMap、多线程、日期类 LocalDate，格式化器、字节流、缓冲区、zip 工具。充分利用异常处理，使用 maven 管理项目。

（2）从代码的规范角度，包名:全小写;类名:首字母大写,每个单词的首字母大写;方法名:小写字母开头,每个单词的首字母大写;变量名: 写字母开头,每个单词的首字母大写;常量名:基本类型的常量名全大写。

（3）前后端分离的设计思路，前端设计了命令行界面类，后端仿照关系型数据库进行构建，界面类调用封装好的函数对数据进行操作，使得程序呈现出高内聚、低耦合的特点。

(4)设计模式方面，采用了单例模式（Singleton Pattern）、策略模式（Strategy Pattern），桥接（Bridge），所有类满足单一职责原则。

（5）注重程序的实用性与操作使用的方便性。加密算法设计了 AES、DES

两种供用户选择。显示文件的表格整齐清晰。

3. 设计模式的使用

（1）单例模式：单例模式（Singleton Pattern）这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。这种模式涉及到一个单一的类，该类负责创建自己的对象，同时确保只有单个对象被创建。这个类提供了一种访问其唯一的对象的方式，可以直接访问，不需要实例化该类的对象。

管理器将工作文件夹抽象成 1 个单例类，提供了一种访问其唯一的对象的方式。

```
PathManager pathManager = PathManager.getInstance();
```

图 3 PathManager 单例类的访问

（2）策略模式：在策略模式（Strategy Pattern）中一个类的行为或其算法可以在运行时更改。这种类型的设计模式属于行为型模式。在策略模式定义了一系列算法或策略，并将每个算法封装在独立的类中，使得它们可以互相替换。通过使用策略模式，可以在运行时根据需要选择不同的算法，而不需要修改客户端代码。

管理器将对所有文件和文件夹的操作，都采用策略模式，文件夹 mystategy 下所有类实现同一个接口。

```
public interface FOperationInterface {
    14 implementations
    void execute(String[] args);
}
```

图 4 FoperationInterface

进行文件或者文件夹操作时，设置不同的 FoperationInterface，选择不同的算法。

```
public class FOperations {

    16 usages
    public void setStrategy(FOperationInterface strategy,String[] args)
        strategy.execute(args);
    }
```

图 5 Foperations 设置策略函数

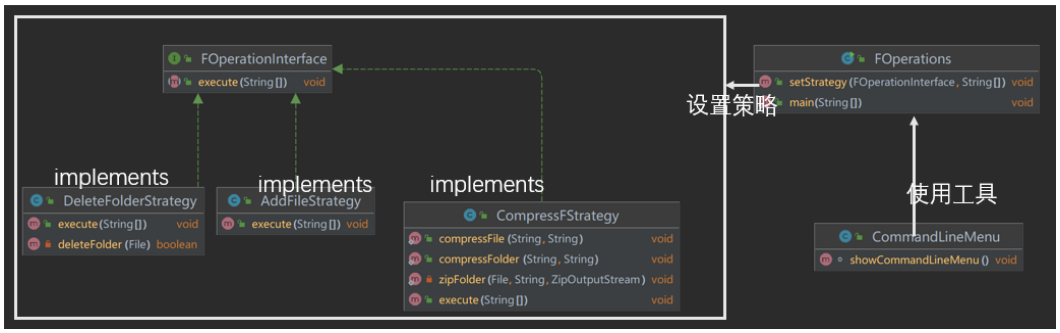

图 6 策略模式图解

（3）桥接（Bridge）是用于把抽象化与实现化解耦，使得二者可以

独立变化。这种类型的设计模式属于结构型模式。

在设计绘制表格工具类时（用于显示工作文件夹下的文件信息），采用了这一设计模式，目的是将抽象与实现分离，使它们可以独立地变化， 将绘制表格的 draw()抽象出来成为接口。符合依赖倒置原则。

```
1 usage   1 implementation
public interface Drawable {
    12 usages   1 implementation
    void draw();
}
```

图 7 绘图接口

4. 程序的结构

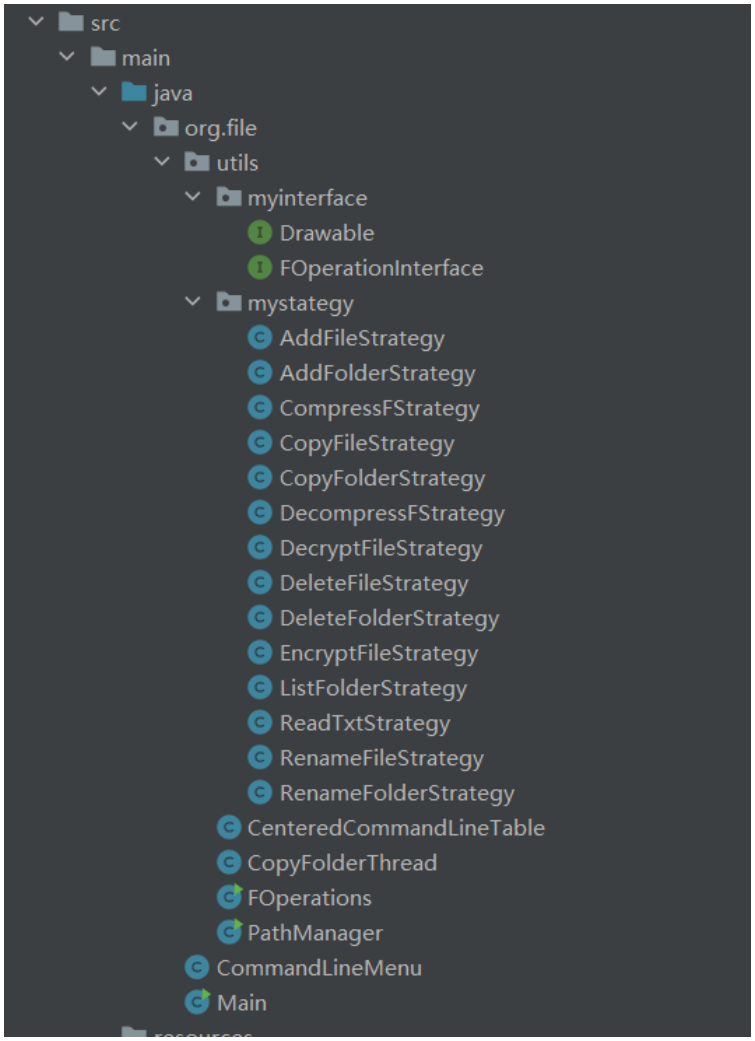

图 8 程序结构图

5. 功能关系图

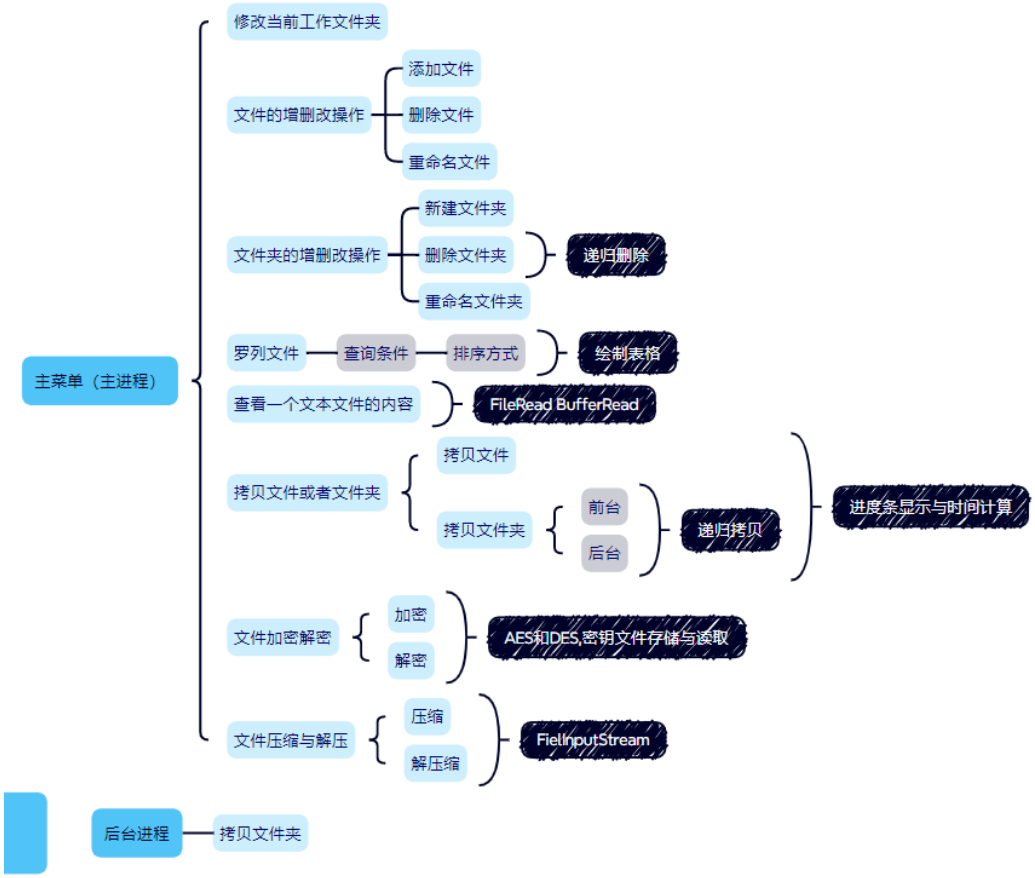

图 9 功能关系图

6.类的说明和类之间的关系图

（1）实体类

PathManager:单例类，包括属性 currentPath

图 10：PathManager 类的成员变量和函数

（2）工具类

Foperations:调用 setStategy 方法，即可执行不同的策略

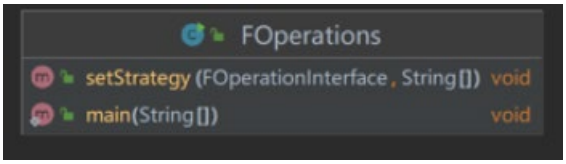

图 11：Foperations 类的成员函数

策略

AddFileStrategy　AddFolderStrategy　CompressFStrategy
CopyFileStrategy　CopyFileStrategyBackground　CopyFolderStrategy
CopyFolderStrategyBackground DecompressFStrategy
DecryptFileStrategy　DeleteFileStrategy　DeleteFolderStrategy
EncryptFileStrategy　ListFolderStrategy　ReadTxtStrategy
RenameFileStrategy RenameFolderStrategy 对文件和文件夹操作的各种策略，每个类都重写了 execute()

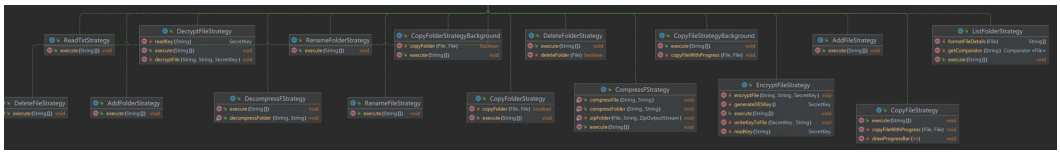

图 11：所有策略的实现类图

（3）接口

Drawable: 为绘制表格实现了对外的绘图接口

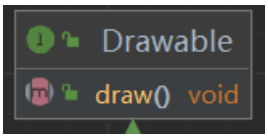

图 12：Drawable 接口

FoperationInterface:策略模式的各个策略都实现这个接口

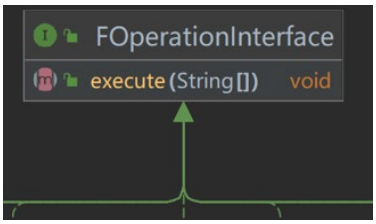

图 13：FoperationInterface 接口

(4)线程

CimmandLineMenu 主线程
CopyFolderThread 工作线程，后台拷贝时调用

图 14：线程

（6）类之间的关系图

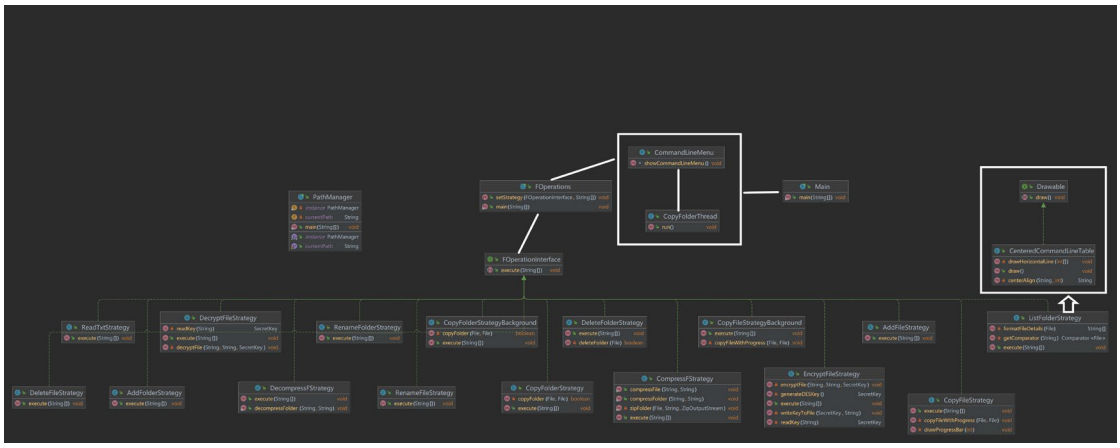

图 15 类之间的关系图

7. 程序主要执行流程图

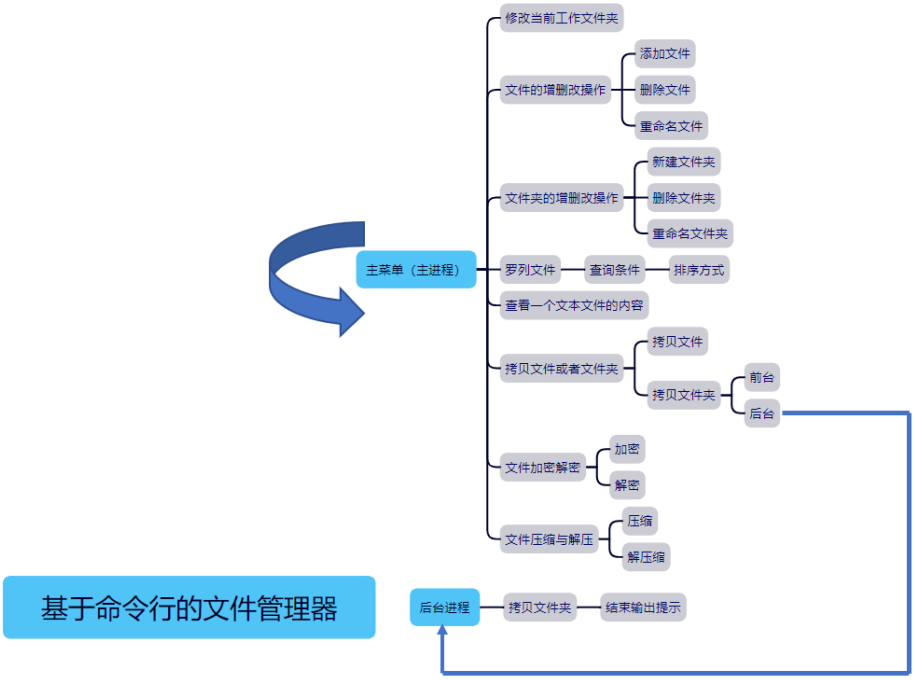

图 16 程序主要执行流程图

8. 核心源代码、截图

（1）接口：

文件或者文件夹操作的接口 FOperationInterface，带实现的函数 void execute(String[] args);

```
package org.file.utils.myinterface;

16 implementations
public interface FOperationInterface {
    16 implementations
    void execute(String[] args);
}
```

图 17 FoperationInterface 接口实现

（2）实现接口：

以添加文件 AddFileStrategy 为例，实现了接口 FoperationInterface，重写了 execute

```
public class AddFileStrategy implements FOperationInterface {
    @Override
    public void execute(String[] args) {
//      参数 创建文件夹的名称
        String fileName = args[0];
        PathManager pathManager = PathManager.getInstance();
        String curPath = pathManager.getCurrentPath();
        String newFilePath = curPath + File.separator + fileName;
        File newFile = new File(newFilePath);
        try {
            if (newFile.createNewFile()) {
                System.out.println("文件已创建: " + newFile.getAbsolutePath());
            } else {
                System.out.println("文件已存在: " + newFile.getAbsolutePath());
            }
        } catch (IOException e) {
            System.out.println("创建文件时出现异常: " + e.getMessage());
        }
    }
}
```

图 18 AddFileStrategy 的实现

（3）FOperations 类，其中 setStrategy 函数接收实现了的 FoperationInterface 如 AddFileStrategy，及其相应的参数，调用策略的 execute()函数即可。

```
public class FOperations {
    17 usages
    public void setStrategy(FOperationInterface strategy,String[] args) {
        strategy.execute(args);
    }
```

图 19 FOperations 类

（4）主线程中调用策略，先创建 Foperations 的对象，设置策略，传递对应参数，即可实现相应功能。故在主线程中，只需知道策略的名称，不需关注具体的实现

```
FOperations fOperations = new FOperations();
fOperations.setStrategy(new AddFolderStrategy(),new String[]{"file"});
```

图 20 调用 Foperations 使用策略

（5）后台异步线程与 callback

(5.1)定义回调接口 CopyFolderThreadCallback

```
public interface CopyFolderThreadCallback {
    1 usage   1 implementation
    void onComplete(String result);
}
```

图 21 定义回调接口 CopyFolderThreadCallback

(5.2)复制文件夹的线程 run 函数，将字符信息传递给在主线程中重写了的 callback.onComplete 函数。

```
public void run() {
    FOperations fOperations = new FOperations();
    fOperations.setStrategy(new CopyFolderStrategyBackground(), new String[]{
    CompletableFuture.runAsync(() -> {
        if (callback != null) {
            callback.onComplete( result: this.sourceFolderPath + " 文件夹拷贝成功: "
        }
    });
}
```

图 22 线程 run 函数

（5.3）在主线程中实现 CopyFolderThreadCallback 的接口

```
CopyFolderThreadCallback callback = new CopyFolderThreadCallback() {
    1 usage
    @Override
    public void onComplete(String result) {
        System.out.println("Callback received: " + result);
    }
};
CopyFolderThread thread = new CopyFolderThread(new String[]{line, line2},callb
thread.start();
break;
```

图 23 实现 CopyFolderThreadCallback 的接口并运行线程

（6）读取文件

```
try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
    String line;
    // 逐行读取文本文件内容并打印到控制台
    while ((line = reader.readLine()) != null) {
        System.out.println(line);
    }
```

图 24 读取文件

（7）压缩文件

```java
public static void compressFile(String sourceFile, String compressedFile) {
    try (FileInputStream fis = new FileInputStream(sourceFile);
         FileOutputStream fos = new FileOutputStream(compressedFile);
         ZipOutputStream zipOut = new ZipOutputStream(fos)) {

        ZipEntry zipEntry = new ZipEntry(new File(sourceFile).getName());
        zipOut.putNextEntry(zipEntry);

        byte[] bytes = new byte[1024];
        int length;
        while ((length = fis.read(bytes)) >= 0) {
            zipOut.write(bytes, off: 0, length);
        }

        System.out.println("文件成功压缩");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

图 25 压缩文件

（8）递归的压缩文件夹

```java
private static void zipFolder(File folder, String parentFolder, ZipOutputStream zipOut) throws IOException {
    for (File file : folder.listFiles()) {
        if (file.isDirectory()) {
            zipFolder(file, parentFolder: parentFolder + "/" + file.getName(), zipOut);
            continue;
        }

        FileInputStream fis = new FileInputStream(file);
        ZipEntry zipEntry = new ZipEntry( name: parentFolder + "/" + file.getName());
        zipOut.putNextEntry(zipEntry);

        byte[] bytes = new byte[1024];
        int length;
        while ((length = fis.read(bytes)) >= 0) {
            zipOut.write(bytes, off: 0, length);
        }

        fis.close();
    }
}
```

图 26 递归的压缩文件夹

（9）文件拷贝显示进度条，可以设置用于拷贝缓冲区的大小。

（9.1）读取文件到 FileInputStream，设置 FileChannel

```
 FileOutputStream outputStream = new FileOutputStream(destinationFile)
 FileChannel inChannel = inputStream.getChannel();
 FileChannel outChannel = outputStream.getChannel()) {

long fileSize = sourceFile.length();
long transferred = 0;
final int bufferSize = 16;
ByteBuffer buffer = ByteBuffer.allocate(bufferSize);

while (inChannel.read(buffer) != -1) {
    buffer.flip();
    outChannel.write(buffer);
    buffer.clear();
    transferred += bufferSize;
```

图 27 复制文件

（9.2）根据进度绘制进度条，　System.out.flush();起到刷新作用

```
vate void drawProgressBar(int progress) {
System.out.print("\r[");
for (int i = 0; i < PROGRESS_BAR_LENGTH; i++) {
    if (i < progress) {
        System.out.print(PROGRESS_CHAR);
    } else {
        System.out.print(" ");
    }
}
Instant endTime = Instant.now(); // 拷贝完成时间
Duration duration = Duration.between(this.startTime, endTime); // 计算拷贝时间
System.out.print("] " + progress * 2 + "%    当前用时:" + duration.toMillis() +
System.out.flush();
```

图 28 绘制进度条

（10）加密文件，

（10.1）要获取待加密文件名称、加密后文件名称、密钥存储的文件名、加密的算法

```
    @Override
    public void execute(String[] args) {
//        FilePath 待加密文件名称
//        newName 加密后文件名称
//        keyPath 密钥存储的文件名
//        encryptionAlgorithm 密钥存储的文件名
        this.fileName = args[0];
        this.newName = args[1];
        this.keyFileName = args[2];
        this.encryptionAlgorithm = args[3];
        this.keyLength = Integer.valueOf(args[4]);
        try {
```

图 29 加密文件

（10.2）如果没有现成的密钥文件，则会生成密钥，有的话就会读取，下面时密钥生成的函数

```
private SecretKey generateDESKey() throws NoSuchAlgorithmException {
    KeyGenerator keyGenerator = KeyGenerator.getInstance(this.encryptionAlgorithm);
    keyGenerator.init(this.keyLength); // 使用128位密钥
    return keyGenerator.generateKey();
}
```

图 30 生成密钥

（10.3）密钥写到密钥文件

```
private void writeKeyToFile(SecretKey secretKey, String filePath) throws IOException {
    byte[] encodedKey = secretKey.getEncoded();
    String encodedKeyString = Base64.getEncoder().encodeToString(encodedKey);
    try (FileWriter fileWriter = new FileWriter(filePath);
         BufferedWriter bufferedWriter = new BufferedWriter(fileWriter))
    {
        bufferedWriter.write(encodedKeyString);
    }
}
```

图 31 密钥写到密钥文件

（10.4）加密文件，Cipher 先初始化算法和密钥，将文件输入流转化 成机密的输出流，再写入到文件中。

```
private void encryptFile(String inputFilePath, String outputFilePath, SecretKey secretKey) throws NoSuchAlgorithmException
    Cipher cipher = Cipher.getInstance(this.encryptionAlgorithm);
    cipher.init(Cipher.ENCRYPT_MODE, secretKey);
    try (FileInputStream inputStream = new FileInputStream(inputFilePath);
         FileOutputStream outputStream = new FileOutputStream(outputFilePath);
         CipherOutputStream cipherOutputStream = new CipherOutputStream(outputStream, cipher)) {

        byte[] buffer = new byte[8192];
        int bytesRead;
        while ((bytesRead = inputStream.read(buffer)) != -1) {
            cipherOutputStream.write(buffer, off: 0, bytesRead);
        }
    }
}
```

图 32　加密文件

四、实验结果及分析和（或）源程序调试过程（界面截图和文字）、实验总结与体会

（一）实验结果及分析

(1)运行程序进入主菜单

```
============== 基于命令行的文件管理器 ==============
当前路径: C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem
======== 主菜单 ========
1. 选项一: 修改当前工作文件夹
2. 选项二: 文件的增删改操作
3. 选项三: 文件夹的增删改操作
4. 选项四: 罗列文件
5. 选项五: 查看一个文本文件的内容
6. 选项六: 拷贝文件或者文件夹
7. 选项七: 文件加密解密
8. 选项八: 文件压缩与解压
9. 退出
请选择操作:
```

图 33 主菜单

（2）输入 1，修改当前工作文件夹，根据提示输入目标路径

```
请选择操作: 1
======== 修改当前工作文件夹 ========
当前路径: C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem
请输入target路径:
C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file
当前路径: C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file
```

图 34 修改当前工作文件夹

（3）输入 2，进行文件的增删改操作

```
请选择操作: 2
======== 文件的增删改操作 ========
1. 选项一: 添加文件
2. 选项二: 删除文件
3. 选项三: 重命名文件
4. 选项四: 返回主菜单
```

图 35 进行文件的增删改操作

（3.1）输入 1，添加文件

```
1
======== 添加文件 ========
请输入文件名:
d.txt
文件已创建: C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file\d.txt
```

图 36 添加文件

（3.2）输入 2，删除文件

```
======== 删除文件 ========
请输入文件名:
d.txt
文件已成功删除: C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file\d.txt
```

图 37 删除文件

（3.3）输入 3，重命名文件

```
3
======== 重命名文件 ========
请输入旧文件名：
key.txt
请输入新文件名：
newkey.txt
文件重命名成功：C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file\newkey.txt
```

图 38 重命名文件

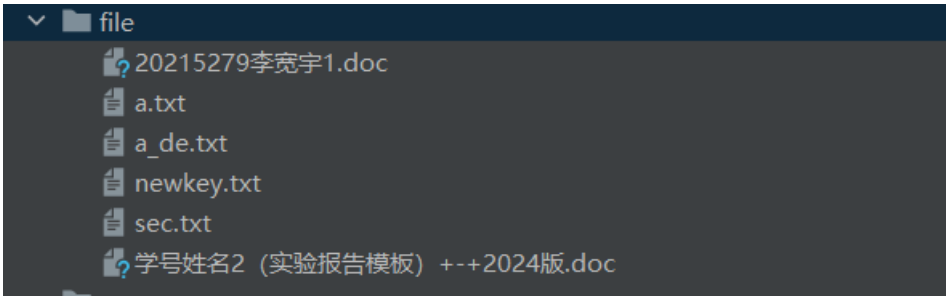

图 39 结果截屏

（3.4）输入 4，返回主菜单

（4）输入 4，罗列当前工作文件夹下的文件，可以根据文件名过滤特定类型的文件。罗列的时候可以排序，依据名称、大小、时间等。

```
请选择操作： 4
======== 罗列文件 ========
输入查询条件（如文件名的一部分，输入回车跳过过滤）：

选择排序方式，1.名称    2.文件大小    3. 日期
1
文件夹内容：
----------------------------------+------+-----------+---------------------+
|          202152791.doc           | File | 2582528字节 | 2024-04-01 16:47:45 |
----------------------------------+------+-----------+---------------------+
| 20254564879419assaaqf+-+2024.doc | File |  34816字节  | 2024-04-01 16:47:45 |
----------------------------------+------+-----------+---------------------+
|              a.txt               | File |    1字节    | 2024-04-01 22:57:36 |
----------------------------------+------+-----------+---------------------+
|             a_de.txt             | File |   24字节    | 2024-04-01 16:47:45 |
----------------------------------+------+-----------+---------------------+
|            newkey.txt            | File |   24字节    | 2024-04-01 16:47:45 |
----------------------------------+------+-----------+---------------------+
|             sec.txt              | File |   32字节    | 2024-04-01 16:47:45 |
----------------------------------+------+-----------+---------------------+
```

图 40 罗列文件

（5）输入 5，查看一个文本文件的内容

```
请选择操作: 5
========= 查看一个文本文件的内容 =========
请输入文件夹名:
a.txt
    才饮长沙水,又食武昌鱼。万里长江横渡,极目楚天舒。不管风吹浪打,胜似闲庭信步,今日得宽余。子在川上曰:逝者如斯夫!
    风樯动,龟蛇静,起宏图。一桥飞架南北,天堑变通途。更立西江石壁,截断巫山云雨,高峡出平湖。神女应无恙,当惊世界殊。
        —毛泽东《水调歌头·游泳》
```

图 41 查看文件

（6）输入 6，拷贝文件或者文件夹

（6.1）输入 1，拷贝文件

```
请选择操作: 6
======== 拷贝文件或者文件夹 ========
1. 选项一: 拷贝文件
2. 选项二: 拷贝文件夹
3. 选项三: 返回主菜单
1
======== 拷贝文件 ========
请输入文件绝对路径 （形如C:\Users\a3840\a.txt）
C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file\202152791.doc
请输入复制到的绝对路径
C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file\202152792.doc
正在拷贝文件：C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file\202152791.doc   到   C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file\
[████████
```

图 42 拷贝文件

（6.2）输入 2，拷贝文件夹

```
========= 拷贝文件夹 =========
请输入文件夹绝对路径: （形如C:\Users\a3840\file）
C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file
请输入复制到的绝对路径: （形如C:\Users\a3840\newFile）
C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\copyFile
文件夹较大，输入1 前台拷贝，输入2后台拷贝
```

图 43 拷贝文件夹

（6.2.1）输入 1，前台拷贝

```
======== 拷贝文件或者文件夹 ========
1. 选项一: 拷贝文件
2. 选项二: 拷贝文件夹
3. 选项三: 返回主菜单
2
======== 拷贝文件夹 ========
请输入文件夹绝对路径: （形如C:\Users\a3840\file）
C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file
请输入复制到的绝对路径: （形如C:\Users\a3840\newFile）
C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\newfile
文件夹较大，输入1 前台拷贝，输入2后台拷贝
1
正在拷贝文件：C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file\202152791.doc   到   C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\newfile\
[██████████████████████████████████████████] 100%    当前用时:9446 毫秒    文件拷贝完成.用时:9447 毫秒
正在拷贝文件：C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file\202152792.doc   到   C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\newfile\
[███████                                   ] 18%    当前用时:1826 毫秒
```

图 44 前台拷贝

（6.2.2）输入 2，后台拷贝

```
文件夹较大，输入1 前台拷贝，输入2后台拷贝
2
======== 主菜单 ========
1. 选项一: 修改当前工作文件夹
2. 选项二: 文件的增删改操作
3. 选项三: 文件夹的增删改操作
4. 选项四: 罗列文件
5. 选项五: 查看一个文本文件的内容
6. 选项六: 拷贝文件或者文件夹
7. 选项七: 文件加密解密
8. 选项八: 文件压缩与解压
9. 退出
请选择操作: 5
======== 查看一个文本文件的内容 ========
请输入文件夹名:
Callback received: C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file 文件夹拷贝成功：C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file2
```

图 45 后台拷贝

此时可以继续经行别的操作，后台拷贝结束会提示

（6.3）输入 3，返回主菜单

（7）输入 7，文件加密解密

（7.1）输入 1，加密，输入待加密文件名称 a.txt，输入加密后文件名称 a_en.txt，输入密钥文件名 key.txt，选择加密算法 AES

```
请选择操作： 7
======== 文件或文件夹加密解密 ========
输入1 加密，输入2 解密
1
请输入待加密文件名称
a.txt
请输入加密后文件名称
a_en.txt
请输入密钥文件名
key.txt
请选择加密算法，   1.AES 2.DES
1
加密成功
```

图 45 文件加密

```
======== 查看一个文本文件的内容 ========
请输入文件夹名：
key.txt
SfgEjZXx4hpd9tt8Cd3OOA==
```

图 46　密钥文件显示

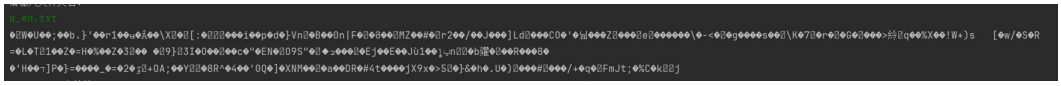

图 47 加密后文件都是乱码

（7.2）输入 2，解密，输入解密文件名称 a_en.txt，输入解密后文件名称 a_de.txt，输入密钥文件名 key.txt，选择加密算法 AES

```
请选择操作： 5
======== 查看一个文本文件的内容 ========
请输入文件夹名：
a_de.txt
    才饮长沙水，又食武昌鱼。万里长江横渡，极目楚天舒。不管风吹浪打，胜似闲庭信步，今日得宽余。子在川上曰：逝者如斯夫！
    风樯动，龟蛇静，起宏图。一桥飞架南北，天堑变通途。更立西江石壁，截断巫山云雨，高峡出平湖。神女应无恙，当惊世界殊。
        ——毛泽东《水调歌头·游泳》
```

图 48 解密

（8）输入 8，进行文件压缩与解压

（8.1）输入 1，压缩输入待压缩文件和文件夹绝对路径 C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file，压缩后文件名 C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file.zip

```
请选择操作： 8
======== 文件压缩与解压 ========
输入1 压缩，输入2 解压缩
1
请输入待压缩文件和文件夹绝对路径 （形如C:\Users\a3840\file）
C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file
请输入压缩后文件名（形如C:\Users\a3840\file\zip））
C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file.zip
文件夹成功压缩
```

图 49 进行文件压缩与解压

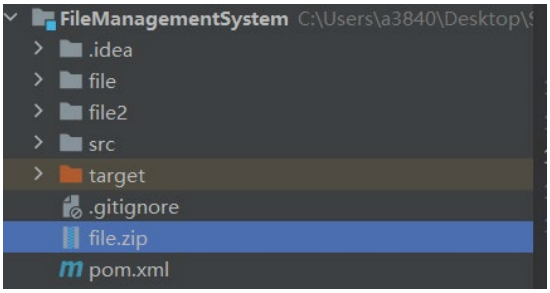

图 50 压缩后结果

（8.2）输入 2，压缩输入待压缩文件和文件夹绝对路径 C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file.zip，压缩后文件名 C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\fileDecompress

```
请选择操作： 8
======== 文件压缩与解压 ========
输入1 压缩，输入2 解压缩
2
请输入待解压文件绝对路径 （形如C:\Users\a3840\file.zip）
C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\file.zip
请输入解压到的文件夹绝对路径（形如C:\Users\a3840\file2））
C:\Users\a3840\Desktop\ScoreManagementSystem\FileManagementSystem\fileDecompress
```

图 51 解压缩

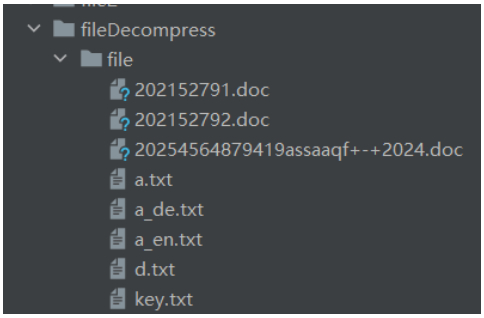

图 52 解压缩后结果

（9）输入 9，结束程序

```
9. 退出
请选择操作： 9

Process finished with exit code 0
```

图 53 结束程序

（二）实验总结与体会

遇到的问题和解决：

（1）如何高效的将文件读取出来并显示

解决:

创建了一个带有缓冲功能的字符流对象.

```java
try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
    String line;
    // 逐行读取文本文件内容并打印到控制台
    while ((line = reader.readLine()) != null) {
        System.out.println(line);
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

图 54 如何高效的将文件读取出来并显示

BufferedReader 和 FileReader 的结合使用，提高文件读取的效率和性能，FileReader 用于读取字符文件，它使用默认的字符编码来解码字节并将其转换为字符。它是 Reader 类的子类，适用于读取文本文件。BufferedReader 是 Reader 的装饰器类，它提供了缓冲功能，即在读取数据时会先将数据读入内存的缓冲区中，然后逐行读取缓冲区的内容，这样可以减少实际 IO 操作的次数，从而提高读取效率。它还提供了 readLine()方法，可以方便地逐行读取文本文件内容。

(2)后台拷贝文件时，调用的函数产生输出，干扰主线程 IO

解决：

最初，尝试对线程的输出重定向，但是使得主线程的输入输出也被重定向。（×）

之后，将输出与线程调用的函数分离开，即线程内不输出，finally 之后再输出。（√）

（3）解压缩文件夹时，路径出错。具体描述：在解压 a.zip 时，直接遍历到的文件(entry=zipInputStream.getNextEntry())路径可能为 a/b/c.txt，没有创建 b 文件夹，导致报错

```java
ZipEntry entry;
while ((entry = zipInputStream.getNextEntry()) != null) {
```

图 55 遇到的问题 3

解决：

分割路径成 String []，逐级创建文件夹，最后才创建文件

```java
// 使用\\分割字符串，并存储到字符数组中
String[] pathComponents = entryName.split( regex: "/");
String curPath = targetFilePath;
File entryFile;
for (int i = 0; i < pathComponents.length - 1; i++) {
    curPath += File.separator + pathComponents[i];
    entryFile = new File(curPath);
    if (!entryFile.exists()) {
        entryFile.mkdirs();
    }
}
```

图 56 逐级创建文件夹

（4）加密算法，DES，密钥长度 56 位，与 AES 长度不同。指定加密算法后没有修改密钥长度，导致报错

解决：

用键值对存储加密算法和密钥长度。

体会：

文件管理器需要实现的功能繁多，可以采用策略模式优化代码结构，当前工作路径唯一，可以采用单例模式。

实验报告填写说明：

1、第一、二部分由老师提供；

2、第三部分填写源程序或者算法，清单文件，资源文件等。源程序要符合程序编写风格（缩进、注释等）；

3、第四部分主要填写程序调试运行过程、结果（截图）、解决问题的方法、总结和体会等；

4、报告规范：包含报告页眉、报告的排版、内容是否填写，命名是否规范等。