项目报告

目录

**一、项目背景2**

**二、项目任务2**

**三、项目分解2**

**四、项目思路3**

**五、项目实现3**

**六、项目总结8**

项目小组：金卓远、高家涵、陈嘉祺

1. 项目背景

现在大多数人的普遍看法时文件后缀名决定文件的类型，这个观点在一般情况下是正确的，当文件真实类型与其扩展名一致时，扩展名就是文件的真实类型。但是文件扩展名并不固定，可以人为修改，当文件扩展名被修改后就无法反应文件的真实类型。

1. 项目任务

根据法方老师给出的项目要求，我们需要基于java开发一个实现对文件及目录进行分析操作的小型应用程序。该小型应用程序需要实现，检测用户数据文件上的异常，检查特定文件（是否时空文件、扩展名和MIME类型是否一致），并支持检查分析文件夹中的一组文件。

1. 项目分解

我们小组将应用程序分解为三个模块：

* 1. 文件夹检查模块

\*文件夹检查模块包含以下四个功能：

* + - * 1. 查看文件夹大小
        2. 查看文件夹内所有文件
        3. 检查文件夹内空文件
        4. 检查文件夹内类型不正确的文件
  1. 文件检查模块

\*文件检查模块包含以下两个功能：

* + - * 1. 查看文件的大小（若文件为空则对用户做出警告）
        2. 查看文件后缀名和MIME类型，并判断类型是否正确
  1. 保存模块

\*保存模块分为以下两个功能：

* + - * 1. 将整个文件夹的分析结果保存到指定目录
        2. 将单个文件的分析结果保存到指定目录

1. 项目思路
2. 项目实现(\*具体项目界面与运行结果见ppt)
   1. 文件夹操作
      * 1. 计算文件夹大小

项目组采用递归算法和File实现对文件夹大小的计算。首先使用isFile()方法判断用户输入的路径下是文件夹还是文件，如果是文件则使用length()方法计算文件大小并累加到计数器len中，如果是文件夹，则进行递归进入到下一层目录，直到最里层。

\*代码如下：

public static long get\_file\_size(File src) {

//计数器

long len = 0;

//创建数组储存文件夹

File[] files = src.listFiles();

//遍历数组

assert files != null;

for (File f : files) {

//判断：

if (f.isFile()) {

//如果是文件则累加

len += f.length();

} else {

//如果不是文件则递归

len += get\_file\_size(f);

}

}

return len;

}

* + - 1. 遍历所有文件

项目组采用递归算法实现遍历文件夹内所有文件，首先将String的路径转化为File类型，创建File数组，遍历数组内的元素，通过getName() 获取文件名并打印。

\*代码如下：

public static void print\_all\_file\_name(File path) {

//进入文件

File[] files = path.listFiles();

//遍历数组

assert files != null;

for (File f : files) {

//判断：文件or文件夹

if (f.isFile()) {

//是文件则打印文件名

System.out.println(f.getName());

} else {

//是文件夹

//先打印文件夹名

System.out.println("--"+f.getName());

//递归

print\_all\_file\_name(f);

}

}

}

* + - 1. 空文件

首先判断是文件夹还是文件，文件夹则通过get\_file\_size(File src)方法计算文件夹大小，文件则通过length()方法计算出文件大小，然后进行判断，如果文件大小为0则警告用户，如果不为0，则告知用户文件大小。递归算法与打印文件名相同。

\*代码如下：

public static void file\_null(String path) {

File f = new File(path);

//判断path路径下的是否是文件夹

if (f.isFile()) {

//路径是文件

long size = f.length();

String name = f.getName();

// System.out.println(size);

//判断文件大小是否为0

if (0 == size) {

System.out.println("文件 " + name + " 的大小为0字节！");

System.out.println("空文件的路径是 "+path);

}

} else {

//路径是文件夹

//计算文件夹大小

long size = get\_file\_size(f);

String name = f.getName();

//判断文件夹大小是否为0

if (0 == size) {

System.out.println("文件夹 " + name + " 的大小为0字节！");

System.out.println("空文件的夹路径是 "+path);

}

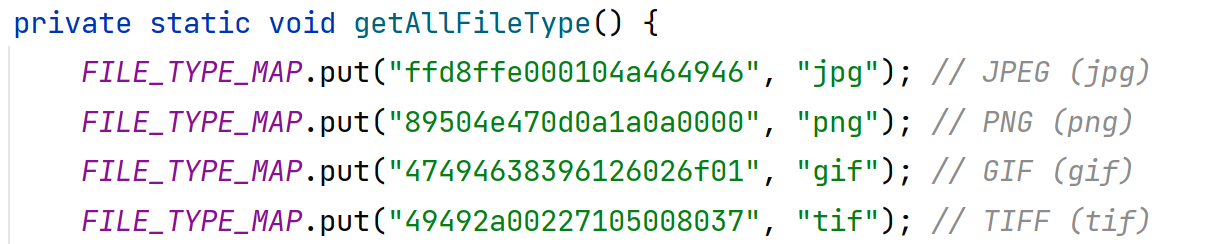
}

}

* + - 1. 文件类型判断

项目组采用不同文件头文件不同的原理判断文件MIME类型，首先将日常使用中经常使用的文件格式（及其对应的MIME类型）全部导入库中，然后分别获取文件后缀名和MIME类型：使用bytesToHexString(byte[] src)方法获取用户输入路径下文件的头文件，再使用type\_true(String path)方法根据头文件获取文件的MIME类型；同时使用type\_suffix(String path)方法获取文件的后缀名，最后使用is\_correct(String path)方法判断文件后缀名和MIME类型是否匹配，如果不匹配则警告用户。

\*文件格式类型过多此处只列举部分：



\*代码如下：

/\*\*

\* 得到上传文件的文件头

\*/

public static String bytesToHexString(byte[] src) {

StringBuilder stringBuilder = new StringBuilder();

if (src == null || src.length <= 0) {

return null;

}

for (int i = 0; i < src.length; i++) {

int v = src[i] & 0xFF;

String hv = Integer.toHexString(v);

if (hv.length() < 2) {

stringBuilder.append(0);

}

stringBuilder.append(hv);

}

return stringBuilder.toString();

}

/\*\*

\* 根据文件的文件头判断文件类型

\*/

public static String type\_true(String path) {

String res = null;

try {

is = new FileInputStream(path);

byte[] b = new byte[10];

is.read(b, 0, b.length);

String fileCode = bytesToHexString(b);

Iterator<String> keyIter = FILE\_TYPE\_MAP.keySet().iterator();

while (keyIter.hasNext()) {

String key = keyIter.next();

// 验证前5个字符比较

if (key.toLowerCase().startsWith(fileCode.toLowerCase().substring(0, 5)) || fileCode.toLowerCase().substring(0, 5).startsWith(key.toLowerCase())) {

res = FILE\_TYPE\_MAP.get(key);

break;

}

}

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

return res;

}

/\*\*

\* 根据文件的后缀名判断文件类型

\*/

public static String type\_suffix(File f) {

String fileName = f.getName();

String suffix = fileName.substring(fileName.lastIndexOf(".")+1);

return suffix;

}

/\*\*

\* 验证文件后缀是否正确

\*/

public static void is\_correct(String path){

String re=null;

if (Get\_file\_type.type\_true(path).equals(Get\_file\_type.type\_suffix(path))){

System.out.println("文件类型匹配");

// re="The file name suffix is correct";

}else {

System.out.println("文件类型不匹配");

// re = "The file name suffix isn't correct";

}

}

* 1. 文件操作

文件操作部分算法实现与文件夹相同。

* 1. GUI图形化界面

项目组运用JPanel框架将界面分为七个部分：

* + - 1. 文件夹目录输入（jp1）
      2. 文件夹分析按钮（jp5）
      3. 文件目录输入（jp2）
      4. 文件分析按钮（jp6）
      5. 保存路径输入（jp7）
      6. 保存按钮（jp3）
      7. 分析完成提示（jp4）

\*代码如下：

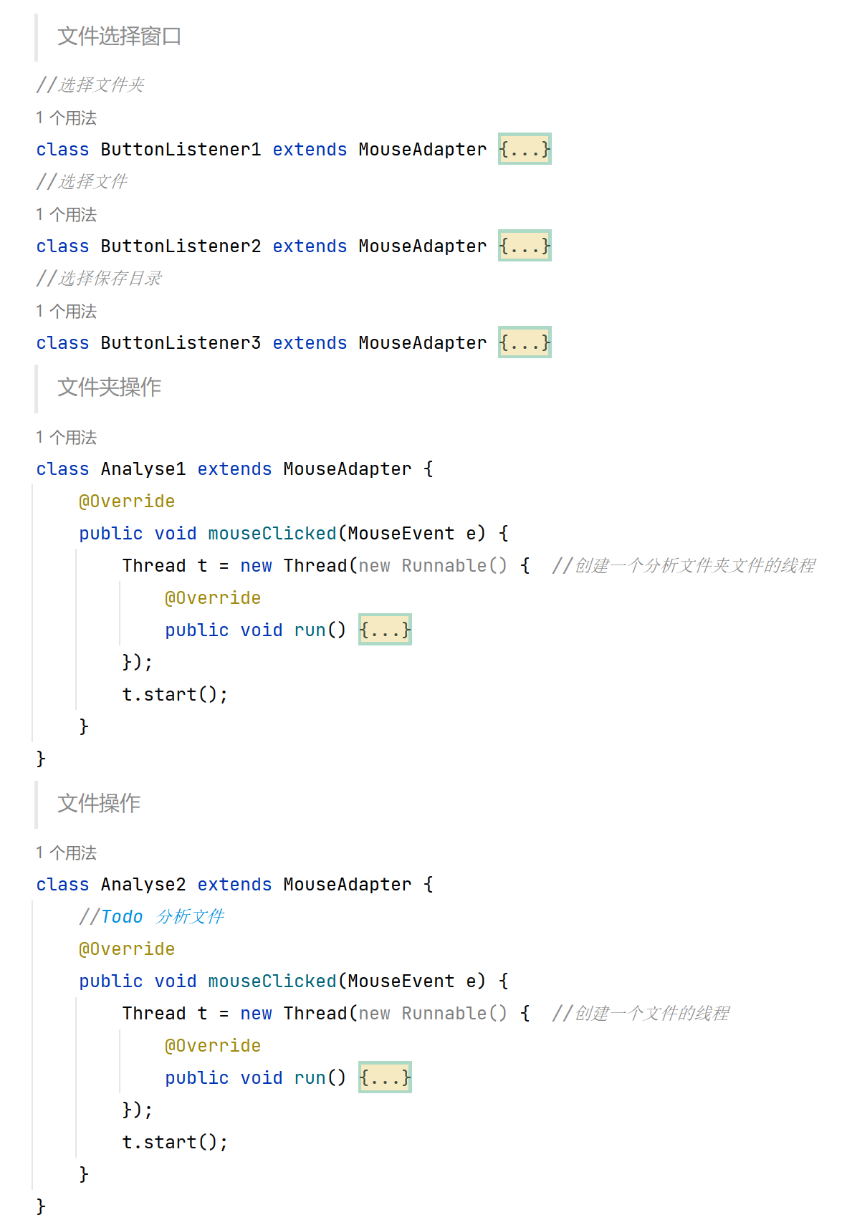




将每个功能写入内部类，添加到按钮监听器中：

并将整个文件夹分析新建一个线程来运行，代码如下：

（具体代码见源码）





1. 项目总结

在完成项目时学习到了很多知识，并且有很多知识是课堂上没有讲到过的，像swing一些高级组件的运用，File类的一些方法，需要自己去Java帮助文档或CSDN去搜索学习。

在项目开发过程中也遇到过很多难题，例如MIME类型怎么去准确判断，开始好多类型的MIME类型值都是null，后来通过头文件和图片判断像素，解压压缩包等方式来判断文件MIME类型。

再比如说，在做GUI图形化界面时，开始只能输出到控制台，无法将结果输出到Frame中，后来采用内部类的方法，给每一个结果窗口写一个内部类，在内部类中创建Frame使用 JTextArea 来输出结果，并给按钮添加监听器来实现。后来还学习了 JPanel 框架，让整个程序条例非常清晰简洁。

文件夹内容多时容易宕机，后来采用多线程的方式，将整个文件夹的分析程序和文件的分析程序在其他线程进行运算，解决了问题。

还有遇到了文件夹内文件太多窗口无法完全显示的情况，查阅资料学习了JScrollbar类，给JTextArea添加了滚动条解决了这个问题。

最后，知道学习计算机需要大量的自学，老师上课讲的内容非常有限，需要自己去网上学习更多的知识才能够完成项目的开发。加油！