项目报告

目录

项目小组：金卓远、高家涵、陈嘉祺

一、 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。

二、 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。

三、 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

（2） 文件检查模块

*文件检查模块包含以下两个功能:

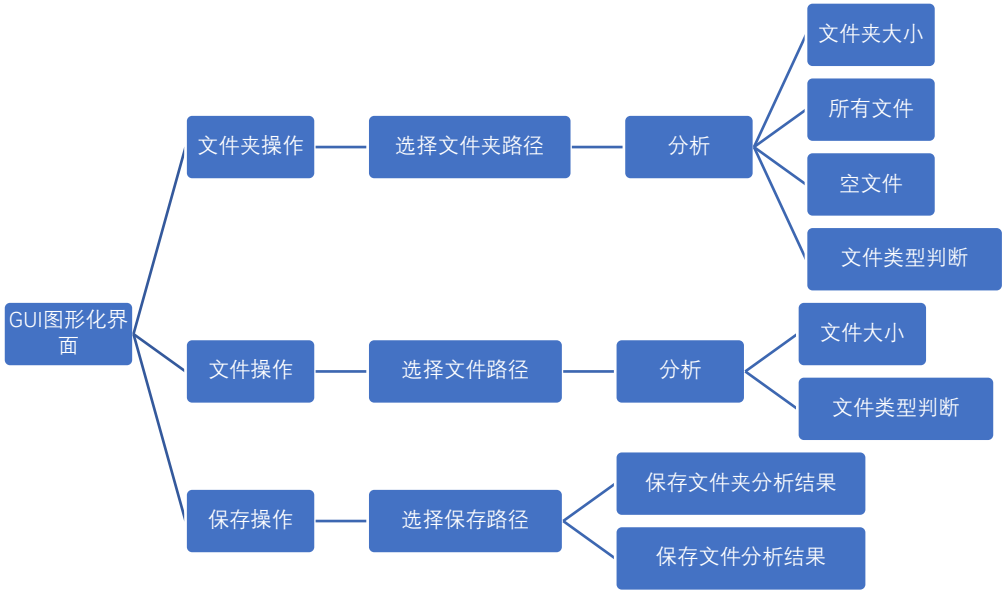
1. 查看文件的大小（若文件为空则对用户做出警告）
2. 查看文件后缀名和 MIME 类型，并判断类型是否正确

（3） 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

四、 项目思路



五、 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;
    }
```

2．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);
            }
        }
    }
```

3．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);
            }
        }
    }
```

4．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：

```
private static void getAllFileType() {
    FILE_TYPE_MAP.put("ffd8ffe000104a464946", "jpg"); // JPEG (jpg)
    FILE_TYPE_MAP.put("89504e470d0a1a0a0000", "png"); // PNG (png)
    FILE_TYPE_MAP.put("47494638396126026f01", "gif"); // GIF (gif)
    FILE_TYPE_MAP.put("49492a00227105008037", "tif"); // TIFF (tif)
```

*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";
      }
    }
```

（2） 文件操作

文件操作部分算法实现与文件夹相同。

（3） 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：

```
//窗口属性设置
UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
this.setTitle("文件分析程序");
this.setLayout(new GridLayout( rows: 7,  cols: 1));
this.setSize( width: 500,  height: 400);
this.setLocationRelativeTo(null);
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

//布局1 文件夹目录输入
JPanel jp1 = new JPanel();
jb1.addMouseListener(new ButtonListener1());
jp1.add(jl1);
jp1.add(jtf1);
jp1.add(jb1);
this.add(jp1);

//布局5 文件夹分析按钮
JPanel jp5=new JPanel();
jButton1.addMouseListener(new FilesCheck1());
jButton2.addMouseListener(new FilesCheck2());
jButton3.addMouseListener(new FilesCheck3());
jButton4.addMouseListener(new FilesCheck4());
jp5.add(jButton1);
jp5.add(jButton2);
jp5.add(jButton3);
jp5.add(jButton4);
this.add(jp5);

//布局2 文件目录输入
JPanel jp2 = new JPanel();
jb2.addMouseListener(new ButtonListener2());
jp2.add(jl2);
jp2.add(jtf2);
jp2.add(jb2);
this.add(jp2);
```

```
//布局6 文件分析按钮
JPanel jp6=new JPanel();
jButton5.addMouseListener(new FilesCheck5());
jButton6.addMouseListener(new FilesCheck6());
jp6.add(jButton5);
jp6.add(jButton6);
this.add(jp6);

//布局7 保存路径输入
JPanel jp7=new JPanel();
jButton_save.addMouseListener(new ButtonListener3());
jp7.add(jLabel_save);
jp7.add(jTextField_save);
jp7.add(jButton_save);
this.add(jp7);

//布局3 保存按钮
JPanel jp3 = new JPanel();
jbAnalyse1.addMouseListener(new Analyse1());
jbAnalyse2.addMouseListener(new Analyse2());
jp3.add(jbAnalyse1);
jp3.add(jbAnalyse2);
this.add(jp3);

//布局4 分析完成提示
JPanel jp4 = new JPanel();
jp4.add(jl3);
this.add(jp4);
```

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）

文件夹功能内部类

```
//文件夹大小
1 个用法
1 class FilesCheck1 extends MouseAdapter{...}
//查看文件夹内所有文件
1 个用法
2 class FilesCheck2 extends MouseAdapter{...}
//检查文件夹内空文件
1 个用法
3 class FilesCheck3 extends MouseAdapter{...}
//类型不匹配的文件
1 个用法
4 class FilesCheck4 extends MouseAdapter{...}
```

文件功能内部类

```
//文件大小
1 个用法
5 class FilesCheck5 extends MouseAdapter{...}
//判断文件后缀名是否正确
1 个用法
6 class FilesCheck6 extends MouseAdapter{...}
```

文件选择窗口

```
//选择文件夹
1 个用法
class ButtonListener1 extends MouseAdapter {...}
//选择文件
1 个用法
class ButtonListener2 extends MouseAdapter {...}
//选择保存目录
1 个用法
class ButtonListener3 extends MouseAdapter {...}
```

文件夹操作

```
1 个用法
class Analyse1 extends MouseAdapter {
    @Override
    public void mouseClicked(MouseEvent e) {
        Thread t = new Thread(new Runnable() {  //创建一个分析文件夹文件的线程
            @Override
            public void run() {...}
        });
        t.start();
    }
}
```

文件操作

```
1 个用法
class Analyse2 extends MouseAdapter {
    //Todo 分析文件
    @Override
    public void mouseClicked(MouseEvent e) {
        Thread t = new Thread(new Runnable() {  //创建一个文件的线程
            @Override
            public void run() {...}
        });
        t.start();
    }
}
```

六、 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!