目录

[**1.课题的主要功能和基本设计思想 15**](#_Toc90920984)

[**2.程序设计思路 15**](#_Toc90920985)

[**2.1. 功能模块的划分 15**](#_Toc90920986)

[**2.2. 设计思路说明 16**](#_Toc90920987)

[**3.主要功能的实现 16**](#_Toc90920988)

[**3.1.主要功能介绍 16**](#_Toc90920989)

[**3.2.程序框图 18**](#_Toc90920990)

[**3.3. 类的层次关系 19**](#_Toc90920991)

[**3.3.1.层次关系图 19**](#_Toc90920992)

[**3.3.2. 自定义类的说明 19**](#_Toc90920993)

[**3.4.类的结构关系 27**](#_Toc90920994)

[**3.5. 类中主要方法 27**](#_Toc90920995)

[**3.5.1. 覆盖、重载关系 27**](#_Toc90920996)

[**3.5.1.1.覆盖 27**](#_Toc90920997)

[**3.5.1.2.重载 28**](#_Toc90920998)

[**3.5.2. 方法实现的功能说明 28**](#_Toc90920999)

[**3.6. 事件监听器 30**](#_Toc90921000)

[**3.7. 内部类使用情况 34**](#_Toc90921001)

[**4.程序运行效果及存在的问题。 35**](#_Toc90921002)

[**4.1. 程序运行效果 35**](#_Toc90921003)

[**4.2. 存在的问题 38**](#_Toc90921004)

[**5.总结 38**](#_Toc90921005)

[**6.源程序清单 40**](#_Toc90921006)

[**6.1. maven的pom.xml文件 40**](#_Toc90921007)

[**6.2.data 44**](#_Toc90921008)

[**6.2.1. Configuration 44**](#_Toc90921009)

[**6.3.io 61**](#_Toc90921010)

[**6.3.1.SHA 61**](#_Toc90921011)

[**6.3.1.1.MD5 61**](#_Toc90921012)

[**6.3.2. Configuration 69**](#_Toc90921013)

[**6.3.3. ErrorLog 74**](#_Toc90921014)

[**6.3.4. File 78**](#_Toc90921015)

[**6.4.UI 88**](#_Toc90921016)

[**6.4.1. About 88**](#_Toc90921017)

[**6.4.2. Color\_JtextArea 90**](#_Toc90921018)

[**6.4.3. ErrorLog 95**](#_Toc90921019)

[**6.4.4. FileInformation 98**](#_Toc90921020)

[**6.4.5. FontSetting 102**](#_Toc90921021)

[**6.4.6. InstructionsForUse 106**](#_Toc90921022)

[**6.4.7. JTextArea\_Border 108**](#_Toc90921023)

[**6.4.8. MainPanel 118**](#_Toc90921024)

[**6.4.9. Replace 164**](#_Toc90921025)

[**6.4.10. Search 169**](#_Toc90921026)

[**6.5. Run 173**](#_Toc90921027)

[**计算机与通信学院课程设计评分表 178**](#_Toc90921028)

# 6. 源程序清单

## 6.1. maven的pom.xml文件

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<!--

-maven项目核心配置文件-

Project name(项目名称)：java课程设计 Swing实现文本编辑器

Author(作者）: mao

Author QQ：1296193245

GitHub：https://github.com/maomao124/

Date(创建日期)： 2021/12/7

Time(创建时间)： 12:41

-->

<groupId>org.example</groupId>

<artifactId>java\_course\_design\_Swing\_implements\_text\_editor</artifactId>

<!--更改项，不能有中文，名称-->

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>16</maven.compiler.source>

<maven.compiler.target>16</maven.compiler.target>

</properties>

<!--依赖包配置放入位置-->

<dependencies>

<!--此依赖用于识别文件编码-->

<dependency>

<groupId>com.ibm.icu</groupId>

<artifactId>icu4j</artifactId>

<version>58.1</version>

</dependency>

</dependencies>

<build>

<finalName>java课程设计\_Swing实现文本编辑器</finalName>

<!--更改项，也可以不改，打包的jar文件名称-->

<plugins>

<plugin>

<artifactId>maven-assembly-plugin</artifactId>

<configuration>

<appendAssemblyId>false</appendAssemblyId>

<descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>

</descriptorRefs>

<archive>

<manifest>

<!-- 此处指定main方法入口的class -->

<mainClass>Run</mainClass>

</manifest>

</archive>

</configuration>

<executions>

<execution>

<id>make-assembly</id>

<phase>package</phase>

<goals>

<goal>assembly</goal>

</goals>

</execution>

</executions>

</plugin>

<!--jar包依赖插件放入位置-->

<!--可选模块，添加console\_hide模式的jar文件 -->

<!--https://github.com/maomao124/run-jar-tool-3.0-->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-antrun-plugin</artifactId>

<version>1.8</version>

<executions>

<execution>

<id>package</id>

<phase>package</phase>

<configuration>

<target>

<echo message="\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*install-or-package\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"/>

<mkdir dir="${basedir}/target/classes"/>

<!--创建文件夹-->

<copy todir="${project.build.directory}/classes" overwrite="true">

<fileset dir="${project.build.directory}"

erroronmissingdir="false">

<include name="\*.jar"/>

</fileset>

</copy>

<move file="${project.build.directory}/classes/java课程设计\_Swing实现文本编辑器.jar"

tofile="${project.build.directory}/java课程设计\_Swing实现文本编辑器\_hide.jar"/>

<copy todir="${project.build.directory}/classes" overwrite="true">

<fileset dir="${project.build.directory}"

erroronmissingdir="false">

<include name="\*.jar"/>

</fileset>

</copy>

<move file="${project.build.directory}/classes/java课程设计\_Swing实现文本编辑器.jar"

tofile="${project.build.directory}/java课程设计\_Swing实现文本编辑器\_args.jar"/>

<move file="${project.build.directory}/classes/java课程设计\_Swing实现文本编辑器\_hide.jar"

tofile="${project.build.directory}/java课程设计\_Swing实现文本编辑器\_args\_save.jar"/>

<!--替换的名称-->

<!--jar包备份-->

<copy todir="H:/jar包/" overwrite="true">

<fileset dir="${project.build.directory}"

erroronmissingdir="false">

<include name="\*.jar"/>

</fileset>

</copy>

<copy todir="${project.build.directory}/" overwrite="true">

<fileset dir="H:/jar包/"

erroronmissingdir="false">

<include name="\*.bat"/>

</fileset>

</copy>

<copy todir="${project.build.directory}/" overwrite="true">

<fileset dir="H:/jar包/"

erroronmissingdir="false">

<include name="jar启动器.7z"/>

</fileset>

</copy>

</target>

</configuration>

<goals>

<goal>run</goal>

</goals>

</execution>

<execution>

<id>clean</id>

<phase>clean</phase>

<configuration>

<target>

<echo message="\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*clean\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"/>

<delete dir="target"/>

<mkdir dir="${basedir}/target/classes"/>

</target>

</configuration>

<goals>

<goal>run</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

## 6.2.data

### 6.2.1. Configuration

package data;

import java.io.Serializable;

/\*\*

\* Project name(项目名称)：java课程设计 Swing实现文本编辑器

\* Package(包名): data

\* Class(类名): Configuration

\* Author(作者）: mao

\* Author QQ：1296193245

\* GitHub：https://github.com/maomao124/

\* Date(创建日期)： 2021/12/9

\* Time(创建时间)： 13:04

\* Version(版本): 1.0

\* Description(描述)： 配置文件的数据类 可序列化

\*/

public class Configuration implements Serializable

{

private int width = 1280; //窗口大小 默认1280\*720

private int height = 720;

private String fontName = "宋体"; //字体设置

private int fontStyle = 0;

private int fontSize = 20;

private int font\_color\_r = 0; //字体颜色 默认黑色

private int font\_color\_g = 0;

private int font\_color\_b = 0;

private int cursor\_color\_r = 0; //光标颜色 默认黑色

private int cursor\_color\_g = 0;

private int cursor\_color\_b = 0;

private int background\_color\_r = 255; //背景颜色 默认白色

private int background\_color\_g = 255;

private int background\_color\_b = 255;

private int selected\_color\_r = 0; //选择颜色 默认白色

private int selected\_color\_g = 0;

private int selected\_color\_b = 0;

private int rendering\_color\_r = 0; //渲染颜色 默认天蓝色

private int rendering\_color\_g = 160;

private int rendering\_color\_b = 200;

private int Layout\_left = 30; //边框大小

private int Layout\_right = 30;

private int Layout\_up = 0;

private int Layout\_down = 15;

boolean wrap = true; //文本域是否换行

boolean isAutoClear = false; //是否自动清理

public Configuration() //无参构造方法

{

}

public int getWidth() //get和set方法

{

return width;

}

public void setWidth(int width)

{

if (width >= 0)

{

this.width = width;

}

else

{

this.width = 0;

}

}

public int getHeight()

{

return height;

}

public void setHeight(int height)

{

if (height >= 0)

{

this.height = height;

}

else

{

this.height = 0;

}

}

public String getFontName()

{

return fontName;

}

public void setFontName(String fontName)

{

this.fontName = fontName;

}

public int getFontStyle()

{

return fontStyle;

}

public void setFontStyle(int fontStyle)

{

if (fontStyle >= 0)

{

this.fontStyle = fontStyle;

}

else

{

this.fontStyle = 0;

}

}

public int getFontSize()

{

return fontSize;

}

public void setFontSize(int fontSize)

{

if (fontSize >= 0)

{

this.fontSize = fontSize;

}

else

{

this.fontSize = 0;

}

}

public int getFont\_color\_r()

{

return font\_color\_r;

}

public void setFont\_color\_r(int font\_color\_r)

{

if (font\_color\_r > 255)

{

font\_color\_r = 255;

}

if (font\_color\_r >= 0)

{

this.font\_color\_r = font\_color\_r;

}

else

{

this.font\_color\_r = 0;

}

}

public int getFont\_color\_g()

{

return font\_color\_g;

}

public void setFont\_color\_g(int font\_color\_g)

{

if (font\_color\_g > 255)

{

font\_color\_g = 255;

}

if (font\_color\_g >= 0)

{

this.font\_color\_g = font\_color\_g;

}

else

{

this.font\_color\_g = 0;

}

}

public int getFont\_color\_b()

{

return font\_color\_b;

}

public void setFont\_color\_b(int font\_color\_b)

{

if (font\_color\_b > 255)

{

font\_color\_b = 255;

}

if (font\_color\_b >= 0)

{

this.font\_color\_b = font\_color\_b;

}

else

{

this.font\_color\_b = 0;

}

}

public int getCursor\_color\_r()

{

return cursor\_color\_r;

}

public void setCursor\_color\_r(int cursor\_color\_r)

{

if (cursor\_color\_r > 255)

{

cursor\_color\_r = 255;

}

if (cursor\_color\_r >= 0)

{

this.cursor\_color\_r = cursor\_color\_r;

}

else

{

this.cursor\_color\_r = 0;

}

}

public int getCursor\_color\_g()

{

return cursor\_color\_g;

}

public void setCursor\_color\_g(int cursor\_color\_g)

{

if (cursor\_color\_g > 255)

{

cursor\_color\_g = 255;

}

if (cursor\_color\_g >= 0)

{

this.cursor\_color\_g = cursor\_color\_g;

}

else

{

this.cursor\_color\_g = 0;

}

}

public int getCursor\_color\_b()

{

return cursor\_color\_b;

}

public void setCursor\_color\_b(int cursor\_color\_b)

{

if (cursor\_color\_b > 255)

{

cursor\_color\_b = 255;

}

if (cursor\_color\_b >= 0)

{

this.cursor\_color\_b = cursor\_color\_b;

}

else

{

this.cursor\_color\_b = 0;

}

}

public int getBackground\_color\_r()

{

return background\_color\_r;

}

public void setBackground\_color\_r(int background\_color\_r)

{

if (background\_color\_r > 255)

{

background\_color\_r = 255;

}

if (background\_color\_r >= 0)

{

this.background\_color\_r = background\_color\_r;

}

else

{

this.background\_color\_r = 0;

}

}

public int getBackground\_color\_g()

{

return background\_color\_g;

}

public void setBackground\_color\_g(int background\_color\_g)

{

if (background\_color\_g > 255)

{

background\_color\_g = 255;

}

if (background\_color\_g >= 0)

{

this.background\_color\_g = background\_color\_g;

}

else

{

this.background\_color\_g = 0;

}

}

public int getBackground\_color\_b()

{

return background\_color\_b;

}

public void setBackground\_color\_b(int background\_color\_b)

{

if (background\_color\_b > 255)

{

background\_color\_b = 255;

}

if (background\_color\_b >= 0)

{

this.background\_color\_b = background\_color\_b;

}

else

{

this.background\_color\_b = 0;

}

}

public int getSelected\_color\_r()

{

return selected\_color\_r;

}

public void setSelected\_color\_r(int selected\_color\_r)

{

if (selected\_color\_r > 255)

{

selected\_color\_r = 255;

}

if (selected\_color\_r >= 0)

{

this.selected\_color\_r = selected\_color\_r;

}

else

{

this.selected\_color\_r = 0;

}

}

public int getSelected\_color\_g()

{

return selected\_color\_g;

}

public void setSelected\_color\_g(int selected\_color\_g)

{

if (selected\_color\_g > 255)

{

selected\_color\_g = 255;

}

if (selected\_color\_g >= 0)

{

this.selected\_color\_g = selected\_color\_g;

}

else

{

this.selected\_color\_g = 0;

}

}

public int getSelected\_color\_b()

{

return selected\_color\_b;

}

public void setSelected\_color\_b(int selected\_color\_b)

{

if (selected\_color\_b > 255)

{

selected\_color\_b = 255;

}

if (selected\_color\_b >= 0)

{

this.selected\_color\_b = selected\_color\_b;

}

else

{

this.selected\_color\_b = 0;

}

}

public int getRendering\_color\_r()

{

return rendering\_color\_r;

}

public void setRendering\_color\_r(int rendering\_color\_r)

{

if (rendering\_color\_r > 255)

{

rendering\_color\_r = 255;

}

if (rendering\_color\_r >= 0)

{

this.rendering\_color\_r = rendering\_color\_r;

}

else

{

this.rendering\_color\_r = 0;

}

}

public int getRendering\_color\_g()

{

return rendering\_color\_g;

}

public void setRendering\_color\_g(int rendering\_color\_g)

{

if (rendering\_color\_g > 255)

{

rendering\_color\_g = 255;

}

if (rendering\_color\_g >= 0)

{

this.rendering\_color\_g = rendering\_color\_g;

}

else

{

this.rendering\_color\_g = 0;

}

}

public int getRendering\_color\_b()

{

return rendering\_color\_b;

}

public void setRendering\_color\_b(int rendering\_color\_b)

{

if (rendering\_color\_b > 255)

{

rendering\_color\_b = 255;

}

if (rendering\_color\_b >= 0)

{

this.rendering\_color\_b = rendering\_color\_b;

}

else

{

this.rendering\_color\_b = 0;

}

}

public int getLayout\_left()

{

return Layout\_left;

}

public void setLayout\_left(int layout\_left)

{

if (layout\_left >= 0)

{

this.Layout\_left = layout\_left;

}

else

{

this.Layout\_left = 0;

}

}

public int getLayout\_right()

{

return Layout\_right;

}

public void setLayout\_right(int layout\_right)

{

if (layout\_right >= 0)

{

this.Layout\_right = layout\_right;

}

else

{

this.Layout\_right = 0;

}

}

public int getLayout\_up()

{

return Layout\_up;

}

public void setLayout\_up(int layout\_up)

{

if (layout\_up >= 0)

{

this.Layout\_up = layout\_up;

}

else

{

this.Layout\_up = 0;

}

}

public int getLayout\_down()

{

return Layout\_down;

}

public void setLayout\_down(int layout\_down)

{

if (layout\_down >= 0)

{

this.Layout\_down = layout\_down;

}

else

{

this.Layout\_down = 0;

}

}

public boolean isWrap()

{

return wrap;

}

public void setWrap(boolean wrap)

{

this.wrap = wrap;

}

public boolean isAutoClear()

{

return isAutoClear;

}

public void setAutoClear(boolean autoClear)

{

isAutoClear = autoClear;

}

@Override

public boolean equals(Object o)

{

if (this == o) //引用同一个对象

{

return true;

}

if (o == null) //检测obj是否为null

{

return false;

}

//if(!(otherObject instanceof ClassName)) //如果所有的子类都拥有统一的语义

if (this.getClass() != o.getClass()) //比较this与obj是否属于同一个类

{

return false;

}

//Object类向下转型

Configuration that = (Configuration) o;

if (width != that.width)

{

return false;

}

if (height != that.height)

{

return false;

}

if (fontStyle != that.fontStyle)

{

return false;

}

if (fontSize != that.fontSize)

{

return false;

}

if (font\_color\_r != that.font\_color\_r)

{

return false;

}

if (font\_color\_g != that.font\_color\_g)

{

return false;

}

if (font\_color\_b != that.font\_color\_b)

{

return false;

}

if (cursor\_color\_r != that.cursor\_color\_r)

{

return false;

}

if (cursor\_color\_g != that.cursor\_color\_g)

{

return false;

}

if (cursor\_color\_b != that.cursor\_color\_b)

{

return false;

}

if (background\_color\_r != that.background\_color\_r)

{

return false;

}

if (background\_color\_g != that.background\_color\_g)

{

return false;

}

if (background\_color\_b != that.background\_color\_b)

{

return false;

}

if (selected\_color\_r != that.selected\_color\_r)

{

return false;

}

if (selected\_color\_g != that.selected\_color\_g)

{

return false;

}

if (selected\_color\_b != that.selected\_color\_b)

{

return false;

}

if (rendering\_color\_r != that.rendering\_color\_r)

{

return false;

}

if (rendering\_color\_g != that.rendering\_color\_g)

{

return false;

}

if (rendering\_color\_b != that.rendering\_color\_b)

{

return false;

}

if (Layout\_left != that.Layout\_left)

{

return false;

}

if (Layout\_right != that.Layout\_right)

{

return false;

}

if (Layout\_up != that.Layout\_up)

{

return false;

}

if (Layout\_down != that.Layout\_down)

{

return false;

}

if (wrap != that.wrap)

{

return false;

}

if (isAutoClear != that.isAutoClear)

{

return false;

}

return fontName.equals(that.fontName);

}

@Override

public int hashCode()

{

int result = width;

result = 31 \* result + height;

result = 31 \* result + fontName.hashCode();

result = 31 \* result + fontStyle;

result = 31 \* result + fontSize;

result = 31 \* result + font\_color\_r;

result = 31 \* result + font\_color\_g;

result = 31 \* result + font\_color\_b;

result = 31 \* result + cursor\_color\_r;

result = 31 \* result + cursor\_color\_g;

result = 31 \* result + cursor\_color\_b;

result = 31 \* result + background\_color\_r;

result = 31 \* result + background\_color\_g;

result = 31 \* result + background\_color\_b;

result = 31 \* result + selected\_color\_r;

result = 31 \* result + selected\_color\_g;

result = 31 \* result + selected\_color\_b;

result = 31 \* result + rendering\_color\_r;

result = 31 \* result + rendering\_color\_g;

result = 31 \* result + rendering\_color\_b;

result = 31 \* result + Layout\_left;

result = 31 \* result + Layout\_right;

result = 31 \* result + Layout\_up;

result = 31 \* result + Layout\_down;

result = 31 \* result + (wrap ? 1 : 0);

result = 31 \* result + (isAutoClear ? 1 : 0);

return result;

}

@Override

@SuppressWarnings("all")

public String toString()

{

final StringBuilder stringbuilder = new StringBuilder();

stringbuilder.append("width：").append(width).append('\n');

stringbuilder.append("height：").append(height).append('\n');

stringbuilder.append("fontName：").append(fontName).append('\n');

stringbuilder.append("fontStyle：").append(fontStyle).append('\n');

stringbuilder.append("fontSize：").append(fontSize).append('\n');

stringbuilder.append("font\_color\_r：").append(font\_color\_r).append('\n');

stringbuilder.append("font\_color\_g：").append(font\_color\_g).append('\n');

stringbuilder.append("font\_color\_b：").append(font\_color\_b).append('\n');

stringbuilder.append("cursor\_color\_r：").append(cursor\_color\_r).append('\n');

stringbuilder.append("cursor\_color\_g：").append(cursor\_color\_g).append('\n');

stringbuilder.append("cursor\_color\_b：").append(cursor\_color\_b).append('\n');

stringbuilder.append("background\_color\_r：").append(background\_color\_r).append('\n');

stringbuilder.append("background\_color\_g：").append(background\_color\_g).append('\n');

stringbuilder.append("background\_color\_b：").append(background\_color\_b).append('\n');

stringbuilder.append("selected\_color\_r：").append(selected\_color\_r).append('\n');

stringbuilder.append("selected\_color\_g：").append(selected\_color\_g).append('\n');

stringbuilder.append("selected\_color\_b：").append(selected\_color\_b).append('\n');

stringbuilder.append("rendering\_color\_r：").append(rendering\_color\_r).append('\n');

stringbuilder.append("rendering\_color\_g：").append(rendering\_color\_g).append('\n');

stringbuilder.append("rendering\_color\_b：").append(rendering\_color\_b).append('\n');

stringbuilder.append("Layout\_left：").append(Layout\_left).append('\n');

stringbuilder.append("Layout\_right：").append(Layout\_right).append('\n');

stringbuilder.append("Layout\_up：").append(Layout\_up).append('\n');

stringbuilder.append("Layout\_down：").append(Layout\_down).append('\n');

stringbuilder.append("wrap：").append(wrap).append('\n');

stringbuilder.append("isAutoClear：").append(isAutoClear).append('\n');

return stringbuilder.toString();

}

}

## 6.3.io

### 6.3.1.SHA

#### 6.3.1.1.MD5

package io.SHA;

import io.File;

import java.io.\*;

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

/\*\*

\* Project name(项目名称)：java课程设计 Swing实现文本编辑器

\* Package(包名): io.SHA

\* Class(类名): MD5

\* Author(作者）: mao

\* Author QQ：1296193245

\* GitHub：https://github.com/maomao124/

\* Date(创建日期)： 2021/12/7

\* Time(创建时间)： 12:50

\* Version(版本): 1.0

\* Description(描述)： 散列算法 MD5

\*/

public class MD5

{

//存储小组

//private long[] groups = null;

//存储结果

private String resultMessage = "";

//四个寄存器的初始向量IV,采用小端存储

private static final long A = 0x67452301L;

private static final long B = 0xefcdab89L;

private static final long C = 0x98badcfeL;

private static final long D = 0x10325476L;

//java不支持无符号的基本数据(unsigned),所以选用long数据类型

private long[] result = {A, B, C, D};

private static final long[][] T =

{

{0xd76aa478, 0xe8c7b756, 0x242070db, 0xc1bdceee,

0xf57c0faf, 0x4787c62a, 0xa8304613, 0xfd469501,

0x698098d8, 0x8b44f7af, 0xffff5bb1, 0x895cd7be,

0x6b901122, 0xfd987193, 0xa679438e, 0x49b40821},

{0xf61e2562, 0xc040b340, 0x265e5a51, 0xe9b6c7aa,

0xd62f105d, 0x02441453, 0xd8a1e681, 0xe7d3fbc8,

0x21e1cde6, 0xc33707d6, 0xf4d50d87, 0x455a14ed,

0xa9e3e905, 0xfcefa3f8, 0x676f02d9, 0x8d2a4c8a},

{0xfffa3942, 0x8771f681, 0x6d9d6122, 0xfde5380c,

0xa4beea44, 0x4bdecfa9, 0xf6bb4b60, 0xbebfbc70,

0x289b7ec6, 0xeaa127fa, 0xd4ef3085, 0x04881d05,

0xd9d4d039, 0xe6db99e5, 0x1fa27cf8, 0xc4ac5665},

{0xf4292244, 0x432aff97, 0xab9423a7, 0xfc93a039,

0x655b59c3, 0x8f0ccc92, 0xffeff47d, 0x85845dd1,

0x6fa87e4f, 0xfe2ce6e0, 0xa3014314, 0x4e0811a1,

0xf7537e82, 0xbd3af235, 0x2ad7d2bb, 0xeb86d391}

};

//表示X[k]中的的k取值，决定如何使用消息分组中的字

private static final int[][] k =

{

{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15},

{1, 6, 11, 0, 5, 10, 15, 4, 9, 14, 3, 8, 13, 2, 7, 12},

{5, 8, 11, 14, 1, 4, 7, 10, 13, 0, 3, 6, 9, 12, 15, 2},

{0, 7, 14, 5, 12, 3, 10, 1, 8, 15, 6, 13, 4, 11, 2, 9}

};

//各次迭代中采用的做循环移位的s值

private static final int[][] S =

{

{7, 12, 17, 22},

{5, 9, 14, 20},

{4, 11, 16, 23},

{6, 10, 15, 21}

};

//4轮循环中使用的生成函数(轮函数)g

private static long g(int i, long b, long c, long d)

{

switch (i)

{

case 0:

return (b & c) | ((~b) & d);

case 1:

return (b & d) | (c & (~d));

case 2:

return b ^ c ^ d;

case 3:

return c ^ (b | (~d));

default:

return 0;

}

}

//开始使用MD5加密

private String start(String message)

{

//转化为字节数组

byte[] inputBytes = message.getBytes();

//6A 61 6E 6b 69 6e 67

//获取字节数组的长度

int byteLen = inputBytes.length;

//得到K值（以bit作单位的message长度）

long K = (long) (byteLen << 3);

//完整小组(512bit)(64byte)的个数

int groupCount = byteLen / 64;

//分块

for (int i = 0; i < groupCount; i++)

{

//每次取512bit

//处理一个分组

H(divide(inputBytes, i \* 64));

}

//填充

int rest = byteLen % 64;

//即将填充的一个分组

byte[] paddingBytes = new byte[64];

//原来的尾部数据

for (int i = 0; i < rest; i++)

{

paddingBytes[i] = inputBytes[byteLen - rest + i];

}

//即小于448bit的情况，先填充100...0再填充K值的低64位

//此时只会新增一个分组

if (rest <= 56)

{

//填充100...0

if (rest < 56)

{

//填充10000000

paddingBytes[rest] = (byte) (1 << 7);

//填充00000000

for (int i = 1; i < 56 - rest; i++)

{

paddingBytes[rest + i] = 0;

}

}

//填充K值低64位

for (int i = 0; i < 8; i++)

{

paddingBytes[56 + i] = (byte) (K & 0xFFL);

K = K >> 8;

}

//处理分组

H(divide(paddingBytes, 0));

//即大于448bit的情况，先填充100...0再填充K值的低64位

//此时会新增两个分组

}

else

{

//填充10000000

paddingBytes[rest] = (byte) (1 << 7);

//填充00000000

for (int i = rest + 1; i < 64; i++)

{

paddingBytes[i] = 0;

}

//处理第一个尾部分组

H(divide(paddingBytes, 0));

//填充00000000

for (int i = 0; i < 56; i++)

{

paddingBytes[i] = 0;

}

//填充低64位

for (int i = 0; i < 8; i++)

{

//这里很关键，使用小端方式，即Byte数组先存储len的低位数据，然后右移len

paddingBytes[56 + i] = (byte) (K & 0xFFL);

K = K >> 8;

}

//处理第二个尾部分组

H(divide(paddingBytes, 0));

}

//将Hash值转换成十六进制的字符串

//小端方式!

for (int i = 0; i < 4; i++)

{

//解决缺少前置0的问题

resultMessage += String.format("%02x", result[i] & 0xFF) +

String.format("%02x", (result[i] & 0xFF00) >> 8) +

String.format("%02x", (result[i] & 0xFF0000) >> 16) +

String.format("%02x", (result[i] & 0xFF000000) >> 24);

}

return resultMessage;

}

//从inputBytes的index开始取512位，作为新的512bit的分组

private static long[] divide(byte[] inputBytes, int start)

{

//存储一整个分组,就是512bit,数组里每个是32bit，就是4字节，为了消除符号位的影响，所以使用long

long[] group = new long[16];

for (int i = 0; i < 16; i++)

{

//每个32bit由4个字节拼接而来

//小端的从byte数组到bit恢复方法

group[i] = byte2unsign(inputBytes[4 \* i + start]) |

(byte2unsign(inputBytes[4 \* i + 1 + start])) << 8 |

(byte2unsign(inputBytes[4 \* i + 2 + start])) << 16 |

(byte2unsign(inputBytes[4 \* i + 3 + start])) << 24;

}

return group;

}

//其实byte相当于一个字节的有符号整数，这里不需要符号位，所以把符号位去掉

private static long byte2unsign(byte b)

{

return b < 0 ? b & 0x7F + 128 : b;

}

// groups[] 中每一个分组512位（64字节）

// MD5压缩函数

private void H(long[] groups)

{

//缓冲区（寄存器）数组

long a = result[0], b = result[1], c = result[2], d = result[3];

//四轮循环

for (int n = 0; n < 4; n++)

{

//16轮迭代

for (int i = 0; i < 16; i++)

{

result[0] += (g(n, result[1], result[2], result[3]) & 0xFFFFFFFFL) + groups[k[n][i]] + T[n][i];

result[0] = result[1] + ((result[0] & 0xFFFFFFFFL) << S[n][i % 4] | ((result[0] & 0xFFFFFFFFL) >>> (32 - S[n][i % 4])));

//循环轮换

long temp = result[3];

result[3] = result[2];

result[2] = result[1];

result[1] = result[0];

result[0] = temp;

}

}

//加入之前计算的结果

result[0] += a;

result[1] += b;

result[2] += c;

result[3] += d;

//防止溢出

for (int n = 0; n < 4; n++)

{

result[n] &= 0xFFFFFFFFL;

}

}

public static String getMD5(String message)

{

String result = "";

MD5 md5 = new MD5();

result = md5.start(message);

md5 = null;

return result;

}

public static String getMD5toUpperCase(String message)

{

String result = "";

MD5 md5 = new MD5();

md5.start(message);

result = md5.resultMessage.toUpperCase();

md5 = null;

return result;

}

public static String getFileMD5(String filePath) //获得文件的MD5值

{

try

{

InputStream fis = new FileInputStream(filePath);

MessageDigest md = MessageDigest.getInstance("MD5");

byte[] buffer = new byte[1024];

int length = -1;

while ((length = fis.read(buffer, 0, 1024)) != -1)

{

md.update(buffer, 0, length);

}

fis.close();

//转换并返回包含16个元素字节数组,返回数值范围为-128到127

byte[] md5Bytes = md.digest();

BigInteger bigInt = new BigInteger(1, md5Bytes);

return bigInt.toString(16);

}

catch (Exception e)

{

e.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

return "";

}

}

private static String SHA(final String strText)

{

String strResult = null;

if (strText != null && strText.length() > 0)

{

try

{

MessageDigest messageDigest = MessageDigest.getInstance("MD5");

messageDigest.update(strText.getBytes(File.encoding));

byte[] byteBuffer = messageDigest.digest();

StringBuilder strHexString = new StringBuilder();

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

{

String hex = Integer.toHexString(0xff & byteBuffer[i]);

if (hex.length() == 1)

{

strHexString.append('0');

}

strHexString.append(hex);

}

strResult = strHexString.toString();

}

catch (NoSuchAlgorithmException | UnsupportedEncodingException e)

{

e.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

}

return strResult;

}

public static String getMD5API(String strText)

{

return SHA(strText);

}

public static String getMD5APItoUpperCase(String strText)

{

return SHA(strText).toUpperCase();

}

}

### 6.3.2. Configuration

package io;

import javax.swing.\*;

import java.awt.\*;

import java.io.\*;

import java.io.File;

/\*\*

\* Project name(项目名称)：java课程设计 Swing实现文本编辑器

\* Package(包名): io

\* Class(类名): Configuration

\* Author(作者）: mao

\* Author QQ：1296193245

\* GitHub：https://github.com/maomao124/

\* Date(创建日期)： 2021/12/9

\* Time(创建时间)： 13:04

\* Version(版本): 1.0

\* Description(描述)： 配置文件的输入和输出

\*/

public class Configuration

{

public static data.Configuration config; //配置文件对象

public static boolean config\_is\_not\_null; //配置文件的对象是否为空

public static void write() //将配置写入内存

{

FileOutputStream fileOutputStream = null;

ObjectOutputStream objectOutputStream = null;

try //文件流打开，文件读写

{

fileOutputStream = new FileOutputStream("Configuration.ini");

objectOutputStream = new ObjectOutputStream(fileOutputStream);

objectOutputStream.writeObject(config);

}

catch (FileNotFoundException e) //文件未找到

{

Toolkit.getDefaultToolkit().beep();

System.err.println("文件未找到！！！ " + "\n错误内容：" + e.toString());

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

catch (Exception e) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

finally

{

try //关闭流

{

if (fileOutputStream != null)

{

fileOutputStream.close();

}

if (objectOutputStream != null)

{

objectOutputStream.close();

}

}

catch (NullPointerException e) //空指针异常

{

Toolkit.getDefaultToolkit().beep();

System.err.println("文件已经被关闭，无法再次关闭！！！");

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

catch (Exception e) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

}

}

public static void read()

{

FileInputStream fileInputStream = null;

ObjectInputStream objectInputStream = null;

try //文件流打开，文件读写

{

fileInputStream = new FileInputStream("Configuration.ini");

objectInputStream = new ObjectInputStream(fileInputStream);

config = (data.Configuration) objectInputStream.readObject();

config\_is\_not\_null = true;

}

catch (FileNotFoundException e) //文件未找到

{

System.err.println("未找到配置文件");

config\_is\_not\_null = false;

}

catch (Exception e) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

config\_is\_not\_null = false;

}

finally

{

try //关闭流

{

if (fileInputStream != null)

{

fileInputStream.close();

}

if (objectInputStream != null)

{

objectInputStream.close();

}

}

catch (NullPointerException e) //空指针异常

{

Toolkit.getDefaultToolkit().beep();

System.err.println("文件已经被关闭，无法再次关闭！！！");

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

catch (Exception e) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

}

}

public static void delete()//删除此抽象路径名表示的文件或目录

{

int result;

//根据本机系统设置和硬件功能发出音频哔声

Toolkit.getDefaultToolkit().beep();

//调出一个对话框，其中选择的数量由optionType参数确定，其中messageType参数确定要显示的图标。 messageType参数主要用于提供外观的默认图标。

result = JOptionPane.showConfirmDialog(null, """

确认删除配置文件？这将删除所有已保存的个性化信息

包括窗口大小、字体、各颜色信息、边框信息、换行策略和自动清理

是否继续？"""

, "数据丢失警告！", JOptionPane.YES\_NO\_OPTION, JOptionPane.ERROR\_MESSAGE);

if (result == 0)

{

File file = new java.io.File("Configuration.ini");

if (!file.exists()) //不存在

{

//根据本机系统设置和硬件功能发出音频哔声

Toolkit.getDefaultToolkit().beep();

//调出一个对话框，其中选择的数量由optionType参数确定，其中messageType参数确定要显示的图标。 messageType参数主要用于提供外观的默认图标。

JOptionPane.showMessageDialog(null, "配置文件不存在！", "删除失败", JOptionPane.ERROR\_MESSAGE);

}

else

{

boolean result1;

result1 = file.delete(); //删除此抽象路径名表示的文件或目录

if (result1)

{

Configuration.config\_is\_not\_null = false;

//调出一个对话框，该对话框使用由messageType参数确定的默认图标显示消息

JOptionPane.showMessageDialog(null, "删除成功！重启软件生效",

"提示", JOptionPane.INFORMATION\_MESSAGE);

}

else

{

//根据本机系统设置和硬件功能发出音频哔声。

Toolkit.getDefaultToolkit().beep();

//调出一个对话框，该对话框使用由messageType参数确定的默认图标显示消息

JOptionPane.showMessageDialog(null, "删除失败！", "提示", JOptionPane.ERROR\_MESSAGE);

}

}

}

}

}

### 6.3.3. ErrorLog

package io;

import javax.swing.\*;

import java.awt.\*;

import java.io.\*;

import java.nio.charset.StandardCharsets;

import java.text.DecimalFormat;

import java.util.Calendar;

/\*\*

\* Project name(项目名称)：java课程设计 Swing实现文本编辑器

\* Package(包名): io

\* Class(类名): ErrorLog

\* Author(作者）: mao

\* Author QQ：1296193245

\* GitHub：https://github.com/maomao124/

\* Date(创建日期)： 2021/12/7

\* Time(创建时间)： 14:20

\* Version(版本): 1.0

\* Description(描述)： 错误日志类 读写

\*/

public class ErrorLog

{

public static void write(String message)

{

Calendar calendar = Calendar.getInstance();// 获取当前时间

int year = calendar.get(Calendar.YEAR);

int month = calendar.get(Calendar.MONTH);

month = month + 1; //月份从 0 开始，所以加 1

int day = calendar.get(Calendar.DATE);

int week = calendar.get(Calendar.DAY\_OF\_WEEK);

week = week - 1; //星期日为第一天

int hour = calendar.get(Calendar.HOUR\_OF\_DAY); //时

int minute = calendar.get(Calendar.MINUTE); //分

int second = calendar.get(Calendar.SECOND); //秒

int millisecond = calendar.get(Calendar.MILLISECOND); //毫秒

int dayOfMonth = calendar.get(Calendar.DAY\_OF\_MONTH); // 获取今天是本月第几天

int dayOfWeekInMonth = calendar.get(Calendar.DAY\_OF\_WEEK\_IN\_MONTH); // 获取今天是本月第几周

int many = calendar.get(Calendar.DAY\_OF\_YEAR); // 获取今天是今年第几天

StringBuilder stringBuffer1 = new StringBuilder();

DecimalFormat decimalFormat1 = new DecimalFormat("00");

//stringBuffer1.append(year).append("年 ").append(month).append("月 ").append(day).append("日 ")

// .append(hour).append("时 ").append(minute).append("分 ").append(second).append("秒");

stringBuffer1.append("日期：");

stringBuffer1.append(decimalFormat1.format(year)).append("/").append(decimalFormat1.format(month)).append("/")

.append(decimalFormat1.format(day)).append(" 时间：").append(decimalFormat1.format(hour))

.append(":").append(decimalFormat1.format(minute)).append(":").append(decimalFormat1.format(second));

//System.out.println(stringBuffer1);

stringBuffer1.append(" 主机：").append(System.getProperty("user.name"));

stringBuffer1.append(" 错误堆栈：\n");

stringBuffer1.append(message);

stringBuffer1.append("\n\n\n");

//写入

FileOutputStream fileOutputStream = null;

try //文件流打开，文件读写

{

fileOutputStream = new FileOutputStream("error.log", true);

fileOutputStream.write(stringBuffer1.toString().getBytes(StandardCharsets.UTF\_8));

}

catch (FileNotFoundException e) //文件未找到

{

Toolkit.getDefaultToolkit().beep();

System.err.println("文件未找到！！！ " + "\n错误内容：" + e.toString());

}

catch (Exception e) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e.printStackTrace();

}

finally

{

try //关闭流

{

if (fileOutputStream != null)

{

fileOutputStream.close();

}

}

catch (NullPointerException e) //空指针异常

{

Toolkit.getDefaultToolkit().beep();

System.err.println("文件已经被关闭，无法再次关闭！！！");

}

catch (Exception e) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e.printStackTrace();

}

}

}

public static void read()

{

FileInputStream fileInputStream = null;

InputStreamReader InputStreamReader = null;

BufferedReader bufferedReader = null;

try //文件流打开，文件读写

{

fileInputStream = new FileInputStream("error.log");

InputStreamReader = new InputStreamReader(fileInputStream, StandardCharsets.UTF\_8);

bufferedReader = new BufferedReader(InputStreamReader);

String str;

JTextArea jTextArea = UI.ErrorLog.getjTextArea\_ErrorLog();

while ((str = bufferedReader.readLine()) != null)

{

jTextArea.append(str + "\n");

}

}

catch (FileNotFoundException e) //文件未找到

{

Toolkit.getDefaultToolkit().beep();

System.err.println("文件未找到！！！ " + "\n错误内容：" + e.toString());

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

catch (Exception e) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

finally

{

try //关闭流

{

if (fileInputStream != null)

{

fileInputStream.close();

}

if (InputStreamReader != null)

{

InputStreamReader.close();

}

if (bufferedReader != null)

{

bufferedReader.close();

}

}

catch (NullPointerException e) //空指针异常

{

Toolkit.getDefaultToolkit().beep();

System.err.println("文件已经被关闭，无法再次关闭！！！");

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

catch (Exception e) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

}

}

}

### 6.3.4. File

package io;

import UI.MainPanel;

import com.ibm.icu.text.CharsetDetector;

import com.ibm.icu.text.CharsetMatch;

import javax.swing.\*;

import java.awt.\*;

import java.io.\*;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

/\*\*

\* Project name(项目名称)：java课程设计 Swing实现文本编辑器

\* Package(包名): io

\* Class(类名): File

\* Author(作者）: mao

\* Author QQ：1296193245

\* GitHub：https://github.com/maomao124/

\* Date(创建日期)： 2021/12/7

\* Time(创建时间)： 13:52

\* Version(版本): 1.0

\* Description(描述)： 文件读写类

\*/

public class File

{

public static String encoding = "UTF-8"; //文件编码

public static String autoDiscernEncoding(java.io.File file)

{

String encoding = "UTF-8";

try

{

Path path = Paths.get(file.getPath());

byte[] data = Files.readAllBytes(path);

CharsetDetector detector = new CharsetDetector();

detector.setText(data);

CharsetMatch match = detector.detect();

encoding = match.getName();

System.out.println("文件：" + file.getName() + "的编码为：" + encoding);

return encoding;

}

catch (IOException e)

{

System.err.println("编码识别失败");

e.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

return encoding;

}

}

public static void read(java.io.File file, JTextArea jTextArea, JLabel label\_Information)

{

FileInputStream fileInputStream = null;

InputStreamReader InputStreamReader = null;

try //文件流打开，文件读写

{

fileInputStream = new FileInputStream(file); // test.autoDiscernEncoding(file)

encoding = autoDiscernEncoding(file);

InputStreamReader = new InputStreamReader(fileInputStream, encoding);

char[] buffer = new char[1024];

int count = 0;

while ((count = InputStreamReader.read(buffer)) != -1)

{

jTextArea.append(new String(buffer, 0, count));

//System.out.println(new String(buffer, 0, count));

}

label\_Information.setText("加载完成");

MainPanel.label\_encoding.setText("编码: " + encoding);

}

catch (FileNotFoundException e1) //文件未找到

{

Toolkit.getDefaultToolkit().beep();

System.err.println("文件未找到！！！ " + "\n错误内容：" + e1.toString());

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

catch (Exception e1) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e1.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

finally

{

try //关闭流

{

if (fileInputStream != null)

{

fileInputStream.close();

}

if (InputStreamReader != null)

{

InputStreamReader.close();

}

}

catch (NullPointerException e1) //空指针异常

{

Toolkit.getDefaultToolkit().beep();

System.err.println("文件已经被关闭，无法再次关闭！！！");

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

catch (Exception e1) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e1.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

}

}

public static void read(java.io.File file, JTextArea jTextArea, JLabel label\_Information, String encode)

{

FileInputStream fileInputStream = null;

InputStreamReader InputStreamReader = null;

try //文件流打开，文件读写

{

jTextArea.setText("");

fileInputStream = new FileInputStream(file); // test.autoDiscernEncoding(file)

encoding = encode;

InputStreamReader = new InputStreamReader(fileInputStream, encode);

char[] buffer = new char[1024];

int count = 0;

while ((count = InputStreamReader.read(buffer)) != -1)

{

jTextArea.append(new String(buffer, 0, count));

//System.out.println(new String(buffer, 0, count));

}

label\_Information.setText("加载完成");

MainPanel.label\_encoding.setText("编码: " + encoding);

}

catch (FileNotFoundException e1) //文件未找到

{

Toolkit.getDefaultToolkit().beep();

System.err.println("文件未找到！！！ " + "\n错误内容：" + e1.toString());

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

catch (Exception e1) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e1.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

finally

{

try //关闭流

{

if (fileInputStream != null)

{

fileInputStream.close();

}

if (InputStreamReader != null)

{

InputStreamReader.close();

}

}

catch (NullPointerException e1) //空指针异常

{

Toolkit.getDefaultToolkit().beep();

System.err.println("文件已经被关闭，无法再次关闭！！！");

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

catch (Exception e1) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e1.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

}

}

public static void write(java.io.File file, JTextArea jTextArea, JLabel label\_Information)

{

FileOutputStream fileOutputStream = null;

try //文件流打开，文件读写

{

label\_Information.setText("正在保存...");

fileOutputStream = new FileOutputStream(file);

fileOutputStream.write(jTextArea.getText().getBytes(encoding));

label\_Information.setText("保存成功");

}

catch (FileNotFoundException e1) //文件未找到

{

Toolkit.getDefaultToolkit().beep();

System.err.println("文件未找到！！！ " + "\n错误内容：" + e1.toString());

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

catch (Exception e1) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e1.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

finally

{

try //关闭流

{

if (fileOutputStream != null)

{

fileOutputStream.close();

}

}

catch (NullPointerException e1) //空指针异常

{

Toolkit.getDefaultToolkit().beep();

System.err.println("文件已经被关闭，无法再次关闭！！！");

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

catch (Exception e1) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e1.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

}

}

public static void write(java.io.File file, JTextArea jTextArea, JLabel label\_Information, String encode)

{

FileOutputStream fileOutputStream = null;

try //文件流打开，文件读写

{

String s = "123";

s.getBytes(encode); //测试编码，避免编码错误时创建文件

label\_Information.setText("正在保存...");

fileOutputStream = new FileOutputStream(file);

fileOutputStream.write(jTextArea.getText().getBytes(encode));

label\_Information.setText("保存成功");

}

catch (FileNotFoundException e1) //文件未找到

{

Toolkit.getDefaultToolkit().beep();

System.err.println("文件未找到！！！ " + "\n错误内容：" + e1.toString());

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

catch (UnsupportedEncodingException e)

{

Toolkit.getDefaultToolkit().beep();

System.out.println("编码\"" + encode + "\"无法识别！");

JOptionPane.showMessageDialog(null,

"编码\"" + encode + "\"无法识别！\n 编码输入错误，或者该编码不支持！", "编码错误", JOptionPane.ERROR\_MESSAGE);

}

catch (Exception e1) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e1.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

finally

{

try //关闭流

{

if (fileOutputStream != null)

{

fileOutputStream.close();

}

}

catch (NullPointerException e1) //空指针异常

{

Toolkit.getDefaultToolkit().beep();

System.err.println("文件已经被关闭，无法再次关闭！！！");

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

catch (Exception e1) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e1.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

}

}

public static void write(JTextArea jTextArea, JLabel label\_Information)

{

FileOutputStream fileOutputStream = null;

try //文件流打开，文件读写

{

label\_Information.setText("正在保存...");

fileOutputStream = new FileOutputStream(MainPanel.getFile());

fileOutputStream.write(jTextArea.getText().getBytes(encoding));

label\_Information.setText("保存成功");

}

catch (FileNotFoundException e1) //文件未找到

{

Toolkit.getDefaultToolkit().beep();

System.err.println("文件未找到！！！ " + "\n错误内容：" + e1.toString());

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

catch (Exception e1) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e1.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

finally

{

try //关闭流

{

if (fileOutputStream != null)

{

fileOutputStream.close();

}

}

catch (NullPointerException e1) //空指针异常

{

Toolkit.getDefaultToolkit().beep();

System.err.println("文件已经被关闭，无法再次关闭！！！");

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

catch (Exception e1) //其它异常

{

Toolkit.getDefaultToolkit().beep();

e1.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e1.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

}

}

public static void args\_read(java.io.File file, JTextArea jTextArea, JLabel label\_Information, JTextField jTextField\_FilePath)

{

if (file != null) //不为空

{

label\_Information.setText("开始加载...");

read(file, jTextArea, label\_Information);

jTextField\_FilePath.setText(file.getAbsolutePath());

}

}

}

## 6.4.UI

### 6.4.1. About

package UI;

import javax.swing.\*;

import javax.swing.border.EmptyBorder;

import java.awt.\*;

import java.net.URL;

/\*\*

\* Project name(项目名称)：java课程设计 Swing实现文本编辑器

\* Package(包名): UI

\* Class(类名): About

\* Author(作者）: mao

\* Author QQ：1296193245

\* GitHub：https://github.com/maomao124/

\* Date(创建日期)： 2021/12/8

\* Time(创建时间)： 20:52

\* Version(版本): 1.0

\* Description(描述)： 关于面板

\*/

public class About extends JFrame

{

public static ImageIcon createImageIcon(String path)

{

URL imgURL = MainPanel.class.getResource(path);

if (imgURL != null)

{

return new ImageIcon(imgURL);

}

else

{

System.err.println("文件未找到: " + path);

return null;

}

}

public About()

{

this.setSize(335, 480);

int x = MainPanel.getjFrame().getX();

int y = MainPanel.getjFrame().getY();

int width = MainPanel.getjFrame().getWidth();

int height = MainPanel.getjFrame().getHeight();

int search\_x = x + width / 2 - 500 / 2;

int search\_y = y + height / 2 - 150 / 2;

this.setLocation(search\_x, search\_y); //确保位于主面板的中央

this.setTitle("关于");

this.setLocationRelativeTo(null);

JPanel panel = new JPanel();

panel.setBorder(new EmptyBorder(10, 5, 10, 5));

JPanel panel1 = new JPanel();

panel1.setBorder(new EmptyBorder(10, 5, 10, 5));

panel.setLayout(new GridLayout(7, 1));

panel1.setLayout(new FlowLayout(FlowLayout.CENTER,5000,0));

JLabel label1 = new JLabel("作者：mao");

label1.setHorizontalAlignment(0);

JLabel label2 = new JLabel("完成时间：2021-12-08");

label2.setHorizontalAlignment(0);

JLabel label5 = new JLabel("最近更新：2021-12-14");

label5.setHorizontalAlignment(0);

JLabel label9 = new JLabel("git提交次数：143次");

label9.setHorizontalAlignment(0);

JLabel label3 = new JLabel("GitHub：https://github.com/maomao124/");

JLabel label6 = new JLabel("远程仓库名：");

JLabel label7 = new JLabel("java\_course\_design\_Swing\_implements\_text\_editor");

JLabel label8 = new JLabel("GitHub网址二维码");

label6.setHorizontalAlignment(0);

label7.setHorizontalAlignment(0);

label3.setHorizontalAlignment(0);

label8.setHorizontalAlignment(0);

ImageIcon icon = createImageIcon("二维码.png"); //获得图片资源

JLabel label4 = new JLabel(icon);

panel.add(label1);

panel.add(label2);

panel.add(label5);

panel.add(label9);

panel.add(label3);

panel.add(label6);

panel.add(label7);

panel1.add(label4);

panel1.add(label8);

JPanel panel2 = new JPanel();

panel2.setLayout(new BorderLayout());

panel2.add(panel, BorderLayout.NORTH);

panel2.add(panel1, BorderLayout.CENTER);

this.add(panel2);

}

}

### 6.4.2. Color\_JtextArea

package UI;

import io.Configuration;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

/\*\*

\* Project name(项目名称)：java课程设计 Swing实现文本编辑器

\* Package(包名): UI

\* Class(类名): Color\_JtestAra

\* Author(作者）: mao

\* Author QQ：1296193245

\* GitHub：https://github.com/maomao124/

\* Date(创建日期)： 2021/12/8

\* Time(创建时间)： 20:01

\* Version(版本): 1.0

\* Description(描述)： 设置文本域颜色

\*/

public class Color\_JTextArea

{

private static JTextArea jTextArea;

public static void init\_Color\_JTextArea(JTextArea jTextArea, JMenuItem

font\_color, JMenuItem cursor\_color, JMenuItem background\_color, JMenuItem selected\_color, JMenuItem rendering\_color)

{

Color\_JTextArea.jTextArea = jTextArea;

font\_color.addActionListener(new ActionListener()

{ //设置监听器 字体颜色

@Override

public void actionPerformed(ActionEvent e)

{

change\_font\_color();

}

});

cursor\_color.addActionListener(new ActionListener()

{ //设置监听器 光标颜色

@Override

public void actionPerformed(ActionEvent e)

{

change\_cursor\_color();

}

});

background\_color.addActionListener(new ActionListener()

{ //设置监听器 背景颜色

@Override

public void actionPerformed(ActionEvent e)

{

change\_background\_color();

}

});

selected\_color.addActionListener(new ActionListener()

{ //设置监听器 选择颜色

@Override

public void actionPerformed(ActionEvent e)

{

change\_selected\_color();

}

});

rendering\_color.addActionListener(new ActionListener()

{ //设置监听器 渲染颜色

@Override

public void actionPerformed(ActionEvent e)

{

change\_rendering\_color();

}

});

}

private static void change\_font\_color() //设置此组件的前景色

{

Color color = null;

color = JColorChooser.showDialog(MainPanel.getjFrame(), "请选择字体颜色", Color.black);

if (color != null)

{

jTextArea.setForeground(color);

if (io.Configuration.config == null) //如果对象不存在就创建对象

{

io.Configuration.config = new data.Configuration();

Configuration.config\_is\_not\_null = true;

}

io.Configuration.config.setFont\_color\_r(color.getRed()); //写入配置

io.Configuration.config.setFont\_color\_g(color.getGreen());

io.Configuration.config.setFont\_color\_b(color.getBlue());

}

}

private static void change\_cursor\_color() //设置用于渲染插入符号的当前颜色

{

Color color = null;

color = JColorChooser.showDialog(MainPanel.getjFrame(), "请选择光标颜色", Color.black);

if (color != null)

{

jTextArea.setCaretColor(color);

if (io.Configuration.config == null) //如果对象不存在就创建对象

{

io.Configuration.config = new data.Configuration();

Configuration.config\_is\_not\_null = true;

}

io.Configuration.config.setCursor\_color\_r(color.getRed()); //写入配置

io.Configuration.config.setCursor\_color\_g(color.getGreen());

io.Configuration.config.setCursor\_color\_b(color.getBlue());

}

}

private static void change\_background\_color() //设置此组件的背景颜色。

// 背景颜色仅在组件不透明时使用，并且仅由JComponent或ComponentUI实现的子类使用

{

Color color = null;

color = JColorChooser.showDialog(MainPanel.getjFrame(), "请选择背景颜色", Color.black);

if (color != null)

{

jTextArea.setBackground(color);

if (io.Configuration.config == null) //如果对象不存在就创建对象

{

io.Configuration.config = new data.Configuration();

Configuration.config\_is\_not\_null = true;

}

io.Configuration.config.setBackground\_color\_r(color.getRed()); //写入配置

io.Configuration.config.setBackground\_color\_g(color.getGreen());

io.Configuration.config.setBackground\_color\_b(color.getBlue());

}

}

private static void change\_selected\_color() //设置用于呈现选定文本的当前颜色

{

Color color = null;

color = JColorChooser.showDialog(MainPanel.getjFrame(), "请选择选中颜色", Color.black);

if (color != null)

{

jTextArea.setSelectedTextColor(color);

if (io.Configuration.config == null) //如果对象不存在就创建对象

{

io.Configuration.config = new data.Configuration();

Configuration.config\_is\_not\_null = true;

}

io.Configuration.config.setSelected\_color\_r(color.getRed()); //写入配置

io.Configuration.config.setSelected\_color\_g(color.getGreen());

io.Configuration.config.setSelected\_color\_b(color.getBlue());

}

}

private static void change\_rendering\_color() //设置用于渲染选择的当前颜色

{

Color color = null;

color = JColorChooser.showDialog(MainPanel.getjFrame(), "请选择渲染颜色", Color.black);

if (color != null)

{

jTextArea.setSelectionColor(color);

if (io.Configuration.config == null) //如果对象不存在就创建对象

{

io.Configuration.config = new data.Configuration();

Configuration.config\_is\_not\_null = true;

}

io.Configuration.config.setRendering\_color\_r(color.getRed()); //写入配置

io.Configuration.config.setRendering\_color\_g(color.getGreen());

io.Configuration.config.setRendering\_color\_b(color.getBlue());

}

}

}

### 6.4.3. ErrorLog

package UI;

import javax.swing.\*;

import javax.swing.border.EmptyBorder;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

/\*\*

\* Project name(项目名称)：java课程设计 Swing实现文本编辑器

\* Package(包名): UI

\* Class(类名): ErrorLog

\* Author(作者）: mao

\* Author QQ：1296193245

\* GitHub：https://github.com/maomao124/

\* Date(创建日期)： 2021/12/8

\* Time(创建时间)： 9:55

\* Version(版本): 1.0

\* Description(描述)： 错误日志类的GUI界面

\*/

public class ErrorLog

{

private static JTextArea jTextArea\_ErrorLog;

private static JScrollPane jScrollPane;

private static JButton button\_back;

private static JButton button\_back\_pop;

public static JTextArea getjTextArea\_ErrorLog()

{

return jTextArea\_ErrorLog;

}

public static void init\_error\_log\_jPanel() //初始化错误日志面板

{

jTextArea\_ErrorLog = new JTextArea(15, 55);

jTextArea\_ErrorLog.setLineWrap(false);

jTextArea\_ErrorLog.setEditable(false);

Font font = new Font("宋体", Font.PLAIN, 18);

jTextArea\_ErrorLog.setFont(font);

jScrollPane = new JScrollPane(jTextArea\_ErrorLog);

JPanel jPanel = new JPanel();

MainPanel.setjPanel\_ErrorLog(jPanel);

jPanel.setLayout(new BorderLayout());

button\_back = new JButton("<-返回");

button\_back\_pop = new JButton("<-返回");

jScrollPane.setBorder(new EmptyBorder(20, 45, 50, 100));

jPanel.add(jScrollPane, BorderLayout.CENTER);

JPanel jPanel2 = new JPanel(); //底部面板

JPanel jPanel3 = new JPanel(); //顶部面板

jPanel2.setLayout(new FlowLayout());

jPanel2.add(button\_back);

jPanel3.setLayout(new FlowLayout(FlowLayout.LEFT, 0, 0));

jPanel3.add(button\_back\_pop);

jPanel.add(jPanel2, BorderLayout.SOUTH);

jPanel.add(jPanel3, BorderLayout.WEST);

button\_back.setBackground(Color.cyan); //设置颜色

button\_back\_pop.setBackground(Color.WHITE);

button\_back.addActionListener(new ActionListener()

{ //设置监听

@Override

public void actionPerformed(ActionEvent e)

{

back();

}

});

button\_back\_pop.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

back();

}

});

MainPanel.getErrorLog().addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

display();

}

});

jPanel.addMouseListener(new MouseAdapter()

{

public void mousePressed(MouseEvent e)

{

int mods = e.getModifiersEx();

if (mods == 16384)

{

back();

}

}

});

jTextArea\_ErrorLog.addMouseListener(new MouseAdapter()

{

public void mousePressed(MouseEvent e)

{

int mods = e.getModifiersEx();

if (mods == 16384)

{

back();

}

}

});

}

public static void display() //显示面板

{

jTextArea\_ErrorLog.setText(""); //清空

io.ErrorLog.read(); //读取日志

JFrame jFrame = MainPanel.getjFrame();

jFrame.remove(MainPanel.getjPanel());

jFrame.add(MainPanel.getjPanel\_ErrorLog());

MainPanel.getjPanel\_ErrorLog().updateUI();

jFrame.repaint();

}

private static void back() //返回到主面板

{

JFrame jFrame = MainPanel.getjFrame();

jFrame.remove(MainPanel.getjPanel\_ErrorLog());

jFrame.add(MainPanel.getjPanel());

jFrame.repaint();

}

}

### 6.4.4. FileInformation

package UI;

import javax.swing.\*;

import javax.swing.border.EmptyBorder;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.io.File;

import java.text.DecimalFormat;

import java.text.SimpleDateFormat;

import java.util.Date;

/\*\*

\* Project name(项目名称)：java课程设计 Swing实现文本编辑器

\* Package(包名): UI

\* Class(类名): FileInformation

\* Author(作者）: mao

\* Author QQ：1296193245

\* GitHub：https://github.com/maomao124/

\* Date(创建日期)： 2021/12/7

\* Time(创建时间)： 12:52

\* Version(版本): 1.0

\* Description(描述)： 文件信息面板

\*/

public class FileInformation

{

private static JTextArea jTextArea\_FileInformation;

private static JScrollPane jScrollPane;

public static void init() //初始化

{

jTextArea\_FileInformation = new JTextArea(15, 55);

jTextArea\_FileInformation.setLineWrap(true);

jTextArea\_FileInformation.setEditable(false);

Font font = new Font("宋体", Font.PLAIN, 22);

jTextArea\_FileInformation.setFont(font);

jScrollPane = new JScrollPane(jTextArea\_FileInformation);

JPanel jPanel = new JPanel();

MainPanel.setjPanel\_FileInformation(jPanel);

jPanel.setLayout(new BorderLayout());

JButton button = MainPanel.getButton\_Back();

JButton button\_back\_pop = new JButton("<-返回");

button\_back\_pop.setBackground(Color.white);

jScrollPane.setBorder(new EmptyBorder(20, 45, 50, 100));

//button.setBorder(new EmptyBorder(20, 20, 20, 20));

jPanel.add(jScrollPane, BorderLayout.CENTER);

JPanel jPanel2 = new JPanel(); //底层面板

JPanel jPanel3 = new JPanel(); //顶层面板

jPanel3.setLayout(new FlowLayout(FlowLayout.LEFT, 0, 0));

jPanel3.add(button\_back\_pop);

jPanel2.setLayout(new FlowLayout());

jPanel2.add(button);

jPanel.add(jPanel2, BorderLayout.SOUTH);

jPanel.add(jPanel3, BorderLayout.WEST);

button.setBackground(Color.cyan);

button.addActionListener(new ActionListener()

{ //设置监听

@Override

public void actionPerformed(ActionEvent e)

{

back();

}

});

button\_back\_pop.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

back();

}

});

MainPanel.getButton\_FileInformation().addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

display();

}

});

MainPanel.getFile\_information().addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

display();

}

});

jPanel.addMouseListener(new MouseAdapter()

{

public void mousePressed(MouseEvent e)

{

int mods = e.getModifiersEx();

if (mods == 16384)

{

back();

}

}

});

jTextArea\_FileInformation.addMouseListener(new MouseAdapter()

{

public void mousePressed(MouseEvent e)

{

int mods = e.getModifiersEx();

if (mods == 16384)

{

back();

}

}

});

}

public static void display() //显示面板

{

if (MainPanel.getFile() == null)

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"还未指定文件目录！！！", "提示", JOptionPane.ERROR\_MESSAGE);

return;

}

jTextArea\_FileInformation.setText("\t\t文件信息：\n\n"); //输出文件的信息

File file = MainPanel.getFile();

DecimalFormat decimalFormat = new DecimalFormat("###.##");

jTextArea\_FileInformation.append("\t文件名称：" + file.getName());

if (file.length() < 1048576)

{

jTextArea\_FileInformation.append("\n\t文件大小：" + file.length() + "字节 =" +

decimalFormat.format((double) file.length() / 1024) + "KB");

}

else

{

jTextArea\_FileInformation.append("\n\t文件大小：" + file.length() + "字节 =" +

decimalFormat.format((double) file.length() / 1024) + "KB =" +

decimalFormat.format((double) (file.length() / 1024 / 1024)) + "MB");

}

jTextArea\_FileInformation.append("\n\t文件相对路径：" + file.getPath());

jTextArea\_FileInformation.append("\n\t文件绝对路径：" + file.getAbsolutePath());

if (file.canRead())

{

jTextArea\_FileInformation.append("\n\t文件是否能读？：是");

}

else

{

jTextArea\_FileInformation.append("\n\t文件是否能读？：否");

}

if (file.canWrite())

{

jTextArea\_FileInformation.append("\n\t文件是否能写？：是");

}

else

{

jTextArea\_FileInformation.append("\n\t文件是否能写？：否");

}

Date date = new Date(file.lastModified());

SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy年 MM月 dd日 E HH点 mm分 ss秒");

jTextArea\_FileInformation.append("\n\t最后修改时间：" + simpleDateFormat.format(date));

JFrame jFrame = MainPanel.getjFrame();

jFrame.remove(MainPanel.getjPanel());

jFrame.add(MainPanel.getjPanel\_FileInformation());

MainPanel.getjPanel\_FileInformation().updateUI();

jFrame.repaint();

}

public static void back() //返回到原来的面板

{

JFrame jFrame = MainPanel.getjFrame();

jFrame.remove(MainPanel.getjPanel\_FileInformation());

jFrame.add(MainPanel.getjPanel());

jFrame.repaint();

}

}

### 6.4.5. FontSetting

package UI;

import io.Configuration;

import javax.swing.\*;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import java.awt.\*;

/\*\*

\* Project name(项目名称)：java课程设计 Swing实现文本编辑器

\* Package(包名): UI

\* Class(类名): Font

\* Author(作者）: mao

\* Author QQ：1296193245

\* GitHub：https://github.com/maomao124/

\* Date(创建日期)： 2021/12/8

\* Time(创建时间)： 19:37

\* Version(版本): 1.0

\* Description(描述)： 字体设置

\*/

public class FontSetting extends JFrame

{

private JTextArea textArea;

private JList<String> list1;

private JList<String> list2;

private JList<String> list3;

private DefaultListModel<String> defaultListModel1;

private DefaultListModel<String> defaultListModel2;

private DefaultListModel<String> defaultListModel3;

private JScrollPane jScrollPane1;

private JScrollPane jScrollPane2;

private JScrollPane jScrollPane3;

public FontSetting(JTextArea textArea)

{

this.textArea = textArea;

this.setTitle("字体设置");

this.setSize(350, 250);

this.setLocationRelativeTo(null);

this.setLayout(new GridLayout(1, 3));

this.addLists();

this.addListener();

}

private void addLists()

{

defaultListModel1 = new DefaultListModel<String>();

defaultListModel2 = new DefaultListModel<String>();

defaultListModel3 = new DefaultListModel<String>();

list1 = new JList<String>(defaultListModel1);

list2 = new JList<String>(defaultListModel2);

list3 = new JList<String>(defaultListModel3);

jScrollPane1 = new JScrollPane(list1);

jScrollPane2 = new JScrollPane(list2);

jScrollPane3 = new JScrollPane(list3);

defaultListModel1.addElement("12");

defaultListModel1.addElement("14");

defaultListModel1.addElement("16");

defaultListModel1.addElement("18");

defaultListModel1.addElement("20");

defaultListModel1.addElement("22");

defaultListModel1.addElement("24");

defaultListModel1.addElement("26");

defaultListModel1.addElement("28");

defaultListModel1.addElement("30");

defaultListModel1.addElement("32");

defaultListModel1.addElement("34");

defaultListModel1.addElement("36");

defaultListModel1.addElement("38");

defaultListModel1.addElement("40");

defaultListModel2.addElement("正常");

defaultListModel2.addElement("粗体");

defaultListModel3.addElement("宋体");

defaultListModel3.addElement("楷体");

defaultListModel3.addElement("黑体");

list1.setSelectedIndex(4);

list2.setSelectedIndex(0);

list3.setSelectedIndex(0);

this.add(jScrollPane1);

this.add(jScrollPane2);

this.add(jScrollPane3);

}

public void addListener()

{

list1.addListSelectionListener(new ListSelectionListener()

{

@Override

public void valueChanged(ListSelectionEvent e)

{

Font font = new java.awt.Font(defaultListModel3.get(list3.getSelectedIndex()),

list2.getSelectedIndex(), Integer.parseInt(defaultListModel1.get(list1.getSelectedIndex())));

textArea.setFont(font);

if (io.Configuration.config == null) //如果对象不存在就创建对象

{

io.Configuration.config = new data.Configuration();

Configuration.config\_is\_not\_null = true;

}

io.Configuration.config.setFontName(font.getName()); //写入配置

io.Configuration.config.setFontStyle(font.getStyle());

io.Configuration.config.setFontSize(font.getSize());

}

});

list2.addListSelectionListener(new ListSelectionListener()

{

@Override

public void valueChanged(ListSelectionEvent e)

{

Font font = new java.awt.Font(defaultListModel3.get(list3.getSelectedIndex()),

list2.getSelectedIndex(), Integer.parseInt(defaultListModel1.get(list1.getSelectedIndex())));

textArea.setFont(font);

if (io.Configuration.config == null) //如果对象不存在就创建对象

{

io.Configuration.config = new data.Configuration();

Configuration.config\_is\_not\_null = true;

}

io.Configuration.config.setFontName(font.getName()); //写入配置

io.Configuration.config.setFontStyle(font.getStyle());

io.Configuration.config.setFontSize(font.getSize());

}

});

list3.addListSelectionListener(new ListSelectionListener()

{

@Override

public void valueChanged(ListSelectionEvent e)

{

Font font = new java.awt.Font(defaultListModel3.get(list3.getSelectedIndex()),

list2.getSelectedIndex(), Integer.parseInt(defaultListModel1.get(list1.getSelectedIndex())));

textArea.setFont(font);

if (io.Configuration.config == null) //如果对象不存在就创建对象

{

io.Configuration.config = new data.Configuration();

Configuration.config\_is\_not\_null = true;

}

io.Configuration.config.setFontName(font.getName()); //写入配置

io.Configuration.config.setFontStyle(font.getStyle());

io.Configuration.config.setFontSize(font.getSize());

}

});

}

}

### 6.4.6. InstructionsForUse

package UI;

import javax.swing.\*;

import java.awt.\*;

/\*\*

\* Project name(项目名称)：java课程设计 Swing实现文本编辑器

\* Package(包名): UI

\* Class(类名): InstructionsForUse

\* Author(作者）: mao

\* Author QQ：1296193245

\* GitHub：https://github.com/maomao124/

\* Date(创建日期)： 2021/12/9

\* Time(创建时间)： 13:03

\* Version(版本): 1.0

\* Description(描述)： 使用说明

\*/

public class InstructionsForUse extends JFrame

{

public InstructionsForUse()

{

this.setTitle("使用说明");

this.setSize(950, 550);

JPanel jPanel = new JPanel();

JTextArea jTextArea = new JTextArea();

jTextArea.setEditable(false);

JScrollPane jScrollPane = new JScrollPane(jTextArea);

String str = """

使用说明：

第一次运行时首先需要单击你需要打开的文本文件，然后鼠标右键选择打开方式，选择使用此程序打开，

建议勾选始终使用此程序打开文本文件。如果勾选始终使用此程序打开后，第二次只需要双击文本文件

就可以使用此程序打开文本文件。

功能：

程序除了支持基本的文字编辑外，还支持自动保存、自动清理、文本边框设置、文件信息查看、设置文字大小、

设置字体、设置字体颜色、设置光标颜色、设置背景颜色、设置选择颜色、设置渲染颜色、可以设置换行策略、

设置编辑模式或者只读模式、支持查找和替换等。

替换功能使用说明：两种方式

第一种：先在文本域选中要替换的文字，再打开替换窗口，窗口的查找栏会自动显示你选中的文字，这时只要在

替换栏输入你要替换的文字再点击右边的替换按钮就行了。

第二种：先打开替换窗口，在搜索栏输入你想要要替换的文字，再点击右边的搜索按钮，搜索到你想要

替换的内容的正确位置后，再在下面的替换栏里输入你要替换的内容，输入完成后再点击右边的替换按钮。

字体颜色、光标颜色、背景颜色、选择颜色和渲染颜色都支持自定义色彩（8位色深 RGB）

自动清理功能每隔10秒清理一次软件内存,建议开启

软件会自动保存用户的个性化配置信息，

这些信息包括包括窗口大小、字体、各颜色信息、边框信息、换行策略和自动清理，

当用户重新打开软件时不需要再次配置。

支持自动识别编码和编码转换

快捷键说明：

当打开错误日志和文件信息界面时，按鼠标退回键快速返回到主面板；

F4：清理软件内存

ctrl+f4：是否自动清理内存

ctrl+f：打开搜索面板

ctrl+g：打开查找面板

ctrl+s：保存

ctrl+shift+s：另存为

ctrl+o：浏览

ctrl+i：打开文件信息面板

ctrl+e：打开错误日志面板

ctrl+a：全选

ctrl+c：复制

ctrl+v：粘贴

ctrl+x：剪切

ctrl+z：撤销

ctrl+w：重做

ctrl+r：编辑模式和只读模式的切换

f3：改变自动保存模式

""";

Font font = new Font("宋体", Font.BOLD, 18);

jTextArea.setFont(font);

jTextArea.setForeground(Color.PINK);

//jTextArea.setBackground(new Color(0,244,125));

jTextArea.setText(str);

jPanel.setLayout(new BorderLayout());

jPanel.add(jScrollPane, BorderLayout.CENTER);

this.add(jPanel);

}

}

### 6.4.7. JTextArea\_Border

package UI;

import io.Configuration;

import javax.swing.\*;

import javax.swing.border.EmptyBorder;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import java.awt.\*;

/\*\*

\* Project name(项目名称)：java课程设计 Swing实现文本编辑器

\* Package(包名): UI

\* Class(类名): Border

\* Author(作者）: mao

\* Author QQ：1296193245

\* GitHub：https://github.com/maomao124/

\* Date(创建日期)： 2021/12/10

\* Time(创建时间)： 20:49

\* Version(版本): 1.0

\* Description(描述)： 边框设置的界面

\*/

public class JTextArea\_Border extends JFrame

{

private JTextArea textArea;

private JScrollPane jScrollPane;

private JList<String> list1;

private JList<String> list2;

private JList<String> list3;

private JList<String> list4;

private DefaultListModel<String> defaultListModel1;

private DefaultListModel<String> defaultListModel2;

private DefaultListModel<String> defaultListModel3;

private DefaultListModel<String> defaultListModel4;

private JScrollPane jScrollPane1;

private JScrollPane jScrollPane2;

private JScrollPane jScrollPane3;

private JScrollPane jScrollPane4;

public JTextArea\_Border(JTextArea textArea, JScrollPane jScrollPane)

{

this.textArea = textArea;

this.jScrollPane = jScrollPane;

this.setTitle("边框设置-从左到右依次是左、右、上、下边框设置");

this.setSize(470, 300);

this.setLocationRelativeTo(null);

this.setLayout(new GridLayout(1, 4));

this.addLists();

this.addListener();

}

private void addLists()

{

defaultListModel1 = new DefaultListModel<String>();

defaultListModel2 = new DefaultListModel<String>();

defaultListModel3 = new DefaultListModel<String>();

defaultListModel4 = new DefaultListModel<String>();

list1 = new JList<String>(defaultListModel1);

list2 = new JList<String>(defaultListModel2);

list3 = new JList<String>(defaultListModel3);

list4 = new JList<String>(defaultListModel4);

jScrollPane1 = new JScrollPane(list1);

jScrollPane2 = new JScrollPane(list2);

jScrollPane3 = new JScrollPane(list3);

jScrollPane4 = new JScrollPane(list4);

defaultListModel1.addElement("0"); //左

defaultListModel1.addElement("5");

defaultListModel1.addElement("10");

defaultListModel1.addElement("15");

defaultListModel1.addElement("20");

defaultListModel1.addElement("25");

defaultListModel1.addElement("30");

defaultListModel1.addElement("35");

defaultListModel1.addElement("40");

defaultListModel1.addElement("45");

defaultListModel1.addElement("50");

defaultListModel1.addElement("55");

defaultListModel1.addElement("60");

defaultListModel1.addElement("65");

defaultListModel1.addElement("70");

defaultListModel1.addElement("75");

defaultListModel1.addElement("80");

defaultListModel1.addElement("85");

defaultListModel1.addElement("90");

defaultListModel1.addElement("95");

defaultListModel1.addElement("100");

defaultListModel1.addElement("110");

defaultListModel1.addElement("120");

defaultListModel1.addElement("130");

defaultListModel1.addElement("140");

defaultListModel1.addElement("150");

defaultListModel1.addElement("160");

defaultListModel1.addElement("170");

defaultListModel1.addElement("180");

defaultListModel1.addElement("190");

defaultListModel1.addElement("200");

defaultListModel1.addElement("220");

defaultListModel1.addElement("240");

defaultListModel1.addElement("260");

defaultListModel1.addElement("280");

defaultListModel1.addElement("300");

defaultListModel1.addElement("325");

defaultListModel1.addElement("350");

defaultListModel1.addElement("375");

defaultListModel1.addElement("400");

defaultListModel1.addElement("425");

defaultListModel1.addElement("450");

defaultListModel1.addElement("475");

defaultListModel1.addElement("500");

defaultListModel1.addElement("525");

defaultListModel1.addElement("550");

defaultListModel1.addElement("575");

defaultListModel1.addElement("600");

defaultListModel1.addElement("625");

defaultListModel1.addElement("650");

defaultListModel1.addElement("675");

defaultListModel1.addElement("700");

defaultListModel1.addElement("725");

defaultListModel1.addElement("750");

defaultListModel2.addElement("0"); //右

defaultListModel2.addElement("5");

defaultListModel2.addElement("10");

defaultListModel2.addElement("15");

defaultListModel2.addElement("20");

defaultListModel2.addElement("25");

defaultListModel2.addElement("30");

defaultListModel2.addElement("35");

defaultListModel2.addElement("40");

defaultListModel2.addElement("45");

defaultListModel2.addElement("50");

defaultListModel2.addElement("55");

defaultListModel2.addElement("60");

defaultListModel2.addElement("65");

defaultListModel2.addElement("70");

defaultListModel2.addElement("75");

defaultListModel2.addElement("80");

defaultListModel2.addElement("85");

defaultListModel2.addElement("90");

defaultListModel2.addElement("95");

defaultListModel2.addElement("100");

defaultListModel2.addElement("110");

defaultListModel2.addElement("120");

defaultListModel2.addElement("130");

defaultListModel2.addElement("140");

defaultListModel2.addElement("150");

defaultListModel2.addElement("160");

defaultListModel2.addElement("170");

defaultListModel2.addElement("180");

defaultListModel2.addElement("190");

defaultListModel2.addElement("200");

defaultListModel2.addElement("220");

defaultListModel2.addElement("240");

defaultListModel2.addElement("260");

defaultListModel2.addElement("280");

defaultListModel2.addElement("300");

defaultListModel2.addElement("325");

defaultListModel2.addElement("350");

defaultListModel2.addElement("375");

defaultListModel2.addElement("400");

defaultListModel2.addElement("425");

defaultListModel2.addElement("450");

defaultListModel2.addElement("475");

defaultListModel2.addElement("500");

defaultListModel2.addElement("525");

defaultListModel2.addElement("550");

defaultListModel2.addElement("575");

defaultListModel2.addElement("600");

defaultListModel2.addElement("625");

defaultListModel2.addElement("650");

defaultListModel2.addElement("675");

defaultListModel2.addElement("700");

defaultListModel2.addElement("725");

defaultListModel2.addElement("750");

defaultListModel3.addElement("0"); //上

defaultListModel3.addElement("5");

defaultListModel3.addElement("10");

defaultListModel3.addElement("15");

defaultListModel3.addElement("20");

defaultListModel3.addElement("25");

defaultListModel3.addElement("30");

defaultListModel3.addElement("35");

defaultListModel3.addElement("40");

defaultListModel3.addElement("45");

defaultListModel3.addElement("50");

defaultListModel3.addElement("55");

defaultListModel3.addElement("60");

defaultListModel3.addElement("65");

defaultListModel3.addElement("70");

defaultListModel3.addElement("75");

defaultListModel3.addElement("80");

defaultListModel3.addElement("85");

defaultListModel3.addElement("90");

defaultListModel3.addElement("95");

defaultListModel3.addElement("100");

defaultListModel3.addElement("110");

defaultListModel3.addElement("120");

defaultListModel3.addElement("130");

defaultListModel3.addElement("140");

defaultListModel3.addElement("150");

defaultListModel3.addElement("160");

defaultListModel3.addElement("170");

defaultListModel3.addElement("180");

defaultListModel3.addElement("190");

defaultListModel3.addElement("200");

defaultListModel3.addElement("220");

defaultListModel3.addElement("240");

defaultListModel3.addElement("260");

defaultListModel3.addElement("280");

defaultListModel3.addElement("300");

defaultListModel3.addElement("325");

defaultListModel3.addElement("350");

defaultListModel3.addElement("375");

defaultListModel3.addElement("400");

defaultListModel3.addElement("425");

defaultListModel3.addElement("450");

defaultListModel3.addElement("475");

defaultListModel3.addElement("500");

defaultListModel4.addElement("0"); //下

defaultListModel4.addElement("5");

defaultListModel4.addElement("10");

defaultListModel4.addElement("15");

defaultListModel4.addElement("20");

defaultListModel4.addElement("25");

defaultListModel4.addElement("30");

defaultListModel4.addElement("35");

defaultListModel4.addElement("40");

defaultListModel4.addElement("45");

defaultListModel4.addElement("50");

defaultListModel4.addElement("55");

defaultListModel4.addElement("60");

defaultListModel4.addElement("65");

defaultListModel4.addElement("70");

defaultListModel4.addElement("75");

defaultListModel4.addElement("80");

defaultListModel4.addElement("85");

defaultListModel4.addElement("90");

defaultListModel4.addElement("95");

defaultListModel4.addElement("100");

defaultListModel4.addElement("110");

defaultListModel4.addElement("120");

defaultListModel4.addElement("130");

defaultListModel4.addElement("140");

defaultListModel4.addElement("150");

defaultListModel4.addElement("160");

defaultListModel4.addElement("170");

defaultListModel4.addElement("180");

defaultListModel4.addElement("190");

defaultListModel4.addElement("200");

defaultListModel4.addElement("220");

defaultListModel4.addElement("240");

defaultListModel4.addElement("260");

defaultListModel4.addElement("280");

defaultListModel4.addElement("300");

defaultListModel4.addElement("325");

defaultListModel4.addElement("350");

defaultListModel4.addElement("375");

defaultListModel4.addElement("400");

defaultListModel4.addElement("425");

defaultListModel4.addElement("450");

defaultListModel4.addElement("475");

defaultListModel4.addElement("500");

list1.setSelectedIndex(6);

list2.setSelectedIndex(6);

list3.setSelectedIndex(0);

list4.setSelectedIndex(3);

this.add(jScrollPane1);

this.add(jScrollPane2);

this.add(jScrollPane3);

this.add(jScrollPane4);

}

public void addListener()

{

list1.addListSelectionListener(new ListSelectionListener()

{

@Override

public void valueChanged(ListSelectionEvent e)

{

int Layout\_left = Integer.parseInt(defaultListModel1.get(list1.getSelectedIndex())); //获得int型的边框信息

int Layout\_right = Integer.parseInt(defaultListModel2.get(list2.getSelectedIndex()));

int Layout\_up = Integer.parseInt(defaultListModel3.get(list3.getSelectedIndex()));

int Layout\_down = Integer.parseInt(defaultListModel4.get(list4.getSelectedIndex()));

jScrollPane.setBorder(new EmptyBorder(Layout\_up, Layout\_left, Layout\_down, Layout\_right));

MainPanel.getjFrame().repaint(); //重新绘制此组件

if (io.Configuration.config == null) //如果对象不存在就创建对象

{

io.Configuration.config = new data.Configuration();

Configuration.config\_is\_not\_null = true;

}

Configuration.config.setLayout\_left(Layout\_left); //写入配置

Configuration.config.setLayout\_right(Layout\_right);

Configuration.config.setLayout\_up(Layout\_up);

Configuration.config.setLayout\_down(Layout\_down);

}

});

list2.addListSelectionListener(new ListSelectionListener()

{

@Override

public void valueChanged(ListSelectionEvent e)

{

int Layout\_left = Integer.parseInt(defaultListModel1.get(list1.getSelectedIndex())); //获得int型的边框信息

int Layout\_right = Integer.parseInt(defaultListModel2.get(list2.getSelectedIndex()));

int Layout\_up = Integer.parseInt(defaultListModel3.get(list3.getSelectedIndex()));

int Layout\_down = Integer.parseInt(defaultListModel4.get(list4.getSelectedIndex()));

jScrollPane.setBorder(new EmptyBorder(Layout\_up, Layout\_left, Layout\_down, Layout\_right));

MainPanel.getjFrame().repaint(); //重新绘制此组件

if (io.Configuration.config == null) //如果对象不存在就创建对象

{

io.Configuration.config = new data.Configuration();

Configuration.config\_is\_not\_null = true;

}

Configuration.config.setLayout\_left(Layout\_left); //写入配置

Configuration.config.setLayout\_right(Layout\_right);

Configuration.config.setLayout\_up(Layout\_up);

Configuration.config.setLayout\_down(Layout\_down);

}

});

list3.addListSelectionListener(new ListSelectionListener()

{

@Override

public void valueChanged(ListSelectionEvent e)

{

int Layout\_left = Integer.parseInt(defaultListModel1.get(list1.getSelectedIndex())); //获得int型的边框信息

int Layout\_right = Integer.parseInt(defaultListModel2.get(list2.getSelectedIndex()));

int Layout\_up = Integer.parseInt(defaultListModel3.get(list3.getSelectedIndex()));

int Layout\_down = Integer.parseInt(defaultListModel4.get(list4.getSelectedIndex()));

jScrollPane.setBorder(new EmptyBorder(Layout\_up, Layout\_left, Layout\_down, Layout\_right));

MainPanel.getjFrame().repaint(); //重新绘制此组件

if (io.Configuration.config == null) //如果对象不存在就创建对象

{

io.Configuration.config = new data.Configuration();

Configuration.config\_is\_not\_null = true;

}

Configuration.config.setLayout\_left(Layout\_left); //写入配置

Configuration.config.setLayout\_right(Layout\_right);

Configuration.config.setLayout\_up(Layout\_up);

Configuration.config.setLayout\_down(Layout\_down);

}

});

list4.addListSelectionListener(new ListSelectionListener()

{

@Override

public void valueChanged(ListSelectionEvent e)

{

int Layout\_left = Integer.parseInt(defaultListModel1.get(list1.getSelectedIndex())); //获得int型的边框信息

int Layout\_right = Integer.parseInt(defaultListModel2.get(list2.getSelectedIndex()));

int Layout\_up = Integer.parseInt(defaultListModel3.get(list3.getSelectedIndex()));

int Layout\_down = Integer.parseInt(defaultListModel4.get(list4.getSelectedIndex()));

jScrollPane.setBorder(new EmptyBorder(Layout\_up, Layout\_left, Layout\_down, Layout\_right));

MainPanel.getjFrame().repaint(); //重新绘制此组件

if (io.Configuration.config == null) //如果对象不存在就创建对象

{

io.Configuration.config = new data.Configuration();

Configuration.config\_is\_not\_null = true;

}

Configuration.config.setLayout\_left(Layout\_left); //写入配置

Configuration.config.setLayout\_right(Layout\_right);

Configuration.config.setLayout\_up(Layout\_up);

Configuration.config.setLayout\_down(Layout\_down);

}

});

}

}

### 6.4.8. MainPanel

package UI;

import io.Configuration;

import io.SHA.MD5;

import javax.swing.\*;

import javax.swing.border.EmptyBorder;

import javax.swing.event.CaretEvent;

import javax.swing.event.CaretListener;

import javax.swing.text.BadLocationException;

import javax.swing.undo.UndoManager;

import java.awt.\*;

import java.awt.event.\*;

import java.io.File;

import java.io.UnsupportedEncodingException;

/\*\*

\* Project name(项目名称)：java课程设计 Swing实现文本编辑器

\* Package(包名): UI

\* Class(类名): MainPanel

\* Author(作者）: mao

\* Author QQ：1296193245

\* GitHub：https://github.com/maomao124/

\* Date(创建日期)： 2021/12/7

\* Time(创建时间)： 12:42

\* Version(版本): 1.0

\* Description(描述)： 主面板类

\*/

public class MainPanel

{

private static JFrame jFrame; //顶层面板

private static JPanel jPanel; //主面板

private static JPanel jPanel\_FileInformation; //文件信息面板

private static JPanel jPanel\_ErrorLog; //错误日志面板

private JTextArea jTextArea; //文本域

private JScrollPane jScrollPane; //滚动面板

private final JLabel label\_FilePath = new JLabel("所选文件路径："); //路径显示

private final JTextField jTextField\_FilePath = new JTextField(35);

private final JButton button\_Open = new JButton("浏览"); //文件打开按钮

private final JButton button\_Save = new JButton("保存");//保存按钮

private final JButton button\_save\_file = new JButton("另存为");

private final JButton button\_EditMode = new JButton("编辑模式"); //编辑模式

private static JButton button\_FileInformation = new JButton("文件信息"); //文件信息按钮

private final JButton button\_autoSave = new JButton("不自动保存"); //自动保存按钮

private static JButton button\_Back = new JButton("<-返回"); //返回按钮

boolean isEditable = true; //文本域是否可以编辑

boolean isAutoClear = false; //是否自动清理软件内存

private static File file; //关联的文件

private final JLabel label\_Information = new JLabel("欢迎使用文件编辑器", JLabel.CENTER); //状态位

public static final JLabel label\_time\_and\_memory = new JLabel("", JLabel.RIGHT);

public static final JLabel label\_localTime = new JLabel("", JLabel.LEFT);

public static final JLabel label\_encoding = new JLabel("", JLabel.LEFT);

private final UI.FontSetting fontSetting;

private final UI.JTextArea\_Border jTextArea\_border;

private final UI.About about\_software;

private final InstructionsForUse instructionsForUse;

private Timer timer\_autoSave;

private Timer timer\_autoClear;

private int auto\_save\_mode = 0;

private UndoManager undoManager; //撤销

@SuppressWarnings("all")

private JMenuBar jMenuBar; //菜单栏

private JPopupMenu jPopupMenu; //弹出菜单

private JMenuItem copy\_pop;

private JMenuItem cut\_pop;

private JMenuItem paste\_pop;

private JMenuItem undo\_pop;

private JMenuItem redo\_pop;

private JMenuItem delete\_pop;

private JMenuItem deleteAll\_pop;

private JMenuItem selectAll\_pop;

@SuppressWarnings("all")

private JMenu menu\_file; //菜单

@SuppressWarnings("all")

private JMenu menu\_edit;

@SuppressWarnings("all")

private JMenu individualization;

@SuppressWarnings("all")

private JMenu format;

@SuppressWarnings("all")

private JMenu help;

private JMenuItem open; // 子菜单

private JMenuItem save;

private JMenuItem save\_as;

private JMenuItem auto\_save;

private static JMenuItem file\_information;

private JMenuItem auto\_clear;

private JMenuItem exit;

private JMenuItem selectAll;

private JMenuItem copy;

private JMenuItem cut;

private JMenuItem paste;

private JMenuItem undo;

private JMenuItem redo;

private JMenuItem delete;

private JMenuItem deleteAll;

private static JMenuItem search;

private static JMenuItem replace;

private JMenuItem edit\_mode;

private JMenuItem font\_setting;

private JMenuItem font\_color;

private JMenuItem cursor\_color;

private JMenuItem background\_color;

private JMenuItem selected\_color;

private JMenuItem rendering\_color;

private JMenuItem border;

private JMenuItem delete\_confirmation;

private JMenuItem wrap;

private JMenuItem encoding\_saveAs;

private JMenuItem overload\_UTF\_8;

private JMenuItem overload\_UTF\_16LE;

private JMenuItem overload\_UTF\_16BE;

private JMenuItem overload\_GBK;

private JMenuItem overload\_GB18030;

private JMenuItem overload\_GB2312;

private JMenuItem overload\_ISO\_8859\_1;

private JMenuItem overload\_US\_ASCII;

private JMenuItem overload\_user\_definition;

private static JMenuItem errorLog;

private JMenuItem instructions\_for\_use;

private JMenuItem about;

//各种get和set方法

public static JButton getButton\_FileInformation()

{

return button\_FileInformation;

}

public static void setButton\_FileInformation(JButton button\_FileInformation)

{

MainPanel.button\_FileInformation = button\_FileInformation;

}

public static JMenuItem getReplace()

{

return replace;

}

public static JPanel getjPanel\_ErrorLog()

{

return jPanel\_ErrorLog;

}

public static JMenuItem getErrorLog()

{

return errorLog;

}

public static void setErrorLog(JMenuItem errorLog)

{

MainPanel.errorLog = errorLog;

}

public static JMenuItem getFile\_information()

{

return file\_information;

}

public static void setFile\_information(JMenuItem file\_information)

{

MainPanel.file\_information = file\_information;

}

public static JMenuItem getSearch()

{

return search;

}

public static void setjPanel\_ErrorLog(JPanel jPanel\_ErrorLog)

{

MainPanel.jPanel\_ErrorLog = jPanel\_ErrorLog;

}

public static JFrame getjFrame()

{

return jFrame;

}

public static void setjFrame(JFrame jFrame)

{

MainPanel.jFrame = jFrame;

}

public static JPanel getjPanel()

{

return jPanel;

}

public static void setjPanel(JPanel jPanel)

{

MainPanel.jPanel = jPanel;

}

public static JPanel getjPanel\_FileInformation()

{

return jPanel\_FileInformation;

}

public static void setjPanel\_FileInformation(JPanel jPanel\_FileInformation)

{

MainPanel.jPanel\_FileInformation = jPanel\_FileInformation;

}

public static JButton getButton\_Back()

{

return button\_Back;

}

public static void setButton\_Back(JButton button\_Back)

{

MainPanel.button\_Back = button\_Back;

}

public static File getFile()

{

return file;

}

public static void setFile(File file)

{

MainPanel.file = file;

}

private void init\_mainPanel() //初始化主面板

{

jTextField\_FilePath.setEditable(false);

jPanel = new JPanel(); //初始化主面板

JPanel jPanel1 = new JPanel(); //上面的按钮

JPanel jPanel2 = new JPanel(); //下面的状态字体

JPanel jPanel\_left = new JPanel(); //左

JPanel jPanel\_center = new JPanel(); //中

JPanel jPanel\_right = new JPanel(); //右

Font font = new Font("宋体", Font.PLAIN, 20); //设置字体

jTextArea = new JTextArea(720 / 35, 1280 / 12);//初始化文本域

jTextArea.setLineWrap(true);

jTextArea.setFont(font);

jTextArea.setEditable(isEditable);

undoManager = new UndoManager(); //撤销功能

jTextArea.getDocument().addUndoableEditListener(undoManager);

jPanel.setLayout(new BorderLayout()); //设置布局

jPanel1.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));

jPanel2.setLayout(new GridLayout(1, 3));

jPanel\_left.setLayout(new FlowLayout(FlowLayout.LEFT, 30, 0));

jPanel\_center.setLayout(new FlowLayout(FlowLayout.CENTER));

jPanel\_right.setLayout(new FlowLayout(FlowLayout.RIGHT, 30, 0));

jScrollPane = new JScrollPane();

jScrollPane.setViewportView(jTextArea);

jScrollPane.setBorder(new EmptyBorder(0, 30, 15, 30));

//jScrollPane.setBorder(new BevelBorder(0, Color.cyan, Color.green, Color.cyan, Color.red));

//jScrollPane.setBorder(new LineBorder(Color.cyan,50,true));

button\_Open.setBackground(Color.cyan); //设置颜色

button\_Save.setBackground(Color.cyan);

button\_autoSave.setBackground(Color.cyan);

button\_EditMode.setBackground(Color.green);

button\_save\_file.setBackground(Color.cyan);

button\_FileInformation.setBackground(Color.cyan);

label\_Information.setForeground(Color.black);

//label\_Information.setPreferredSize(new Dimension(800, 30));

jPanel1.add(label\_FilePath); //加入到主面板中

jPanel1.add(jTextField\_FilePath);

jPanel1.add(button\_Open);

jPanel1.add(button\_Save);

jPanel1.add(button\_save\_file);

jPanel1.add(button\_EditMode);

jPanel1.add(button\_autoSave);

jPanel1.add(button\_FileInformation);

jPanel.add(jPanel1, BorderLayout.NORTH);

jPanel.add(jScrollPane, BorderLayout.CENTER);

jPanel\_left.add(label\_localTime);

jPanel\_left.add(label\_encoding);

jPanel\_center.add(label\_Information);

jPanel\_right.add(label\_time\_and\_memory);

jPanel2.add(jPanel\_left);

jPanel2.add(jPanel\_center);

jPanel2.add(jPanel\_right);

jPanel.add(jPanel2, BorderLayout.SOUTH);

}

private void init\_menu() //初始化菜单面板

{

// 菜单栏

jMenuBar = new JMenuBar();

// 弹出菜单

jPopupMenu = new JPopupMenu();

copy\_pop = new JMenuItem("复制");

cut\_pop = new JMenuItem("剪切");

paste\_pop = new JMenuItem("粘贴");

undo\_pop = new JMenuItem("撤销");

redo\_pop = new JMenuItem("重做");

delete\_pop = new JMenuItem("删除");

deleteAll\_pop = new JMenuItem("清空");

selectAll\_pop = new JMenuItem("全选");

copy\_pop.setBackground(Color.cyan);

cut\_pop.setBackground(Color.cyan);

paste\_pop.setBackground(Color.cyan);

undo\_pop.setBackground(Color.cyan);

redo\_pop.setBackground(Color.cyan);

delete\_pop.setBackground(Color.yellow);

deleteAll\_pop.setBackground(Color.red);

selectAll\_pop.setBackground(Color.cyan);

jPopupMenu.add(copy\_pop);

jPopupMenu.add(cut\_pop);

jPopupMenu.add(paste\_pop);

jPopupMenu.add(undo\_pop);

jPopupMenu.add(redo\_pop);

jPopupMenu.add(delete\_pop);

jPopupMenu.add(deleteAll\_pop);

jPopupMenu.add(selectAll\_pop);

// 菜单

menu\_file = new JMenu("文件");

menu\_edit = new JMenu("编辑");

individualization = new JMenu("个性化");

format = new JMenu("格式");

help = new JMenu("帮助");

// 子菜单

open = new JMenuItem("浏览");

save = new JMenuItem("保存");

save\_as = new JMenuItem("另存为");

auto\_save = new JMenuItem("不自动保存");

file\_information = new JMenuItem("文件信息");

auto\_clear = new JMenuItem("自动清理");

exit = new JMenuItem("退出");

open.setBackground(Color.cyan);

save.setBackground(Color.cyan);

save\_as.setBackground(Color.cyan);

auto\_save.setBackground(Color.cyan);

file\_information.setBackground(Color.cyan);

auto\_clear.setBackground(Color.green);

exit.setBackground(Color.red);

selectAll = new JMenuItem("全选");

copy = new JMenuItem("复制");

cut = new JMenuItem("剪切");

paste = new JMenuItem("粘贴");

undo = new JMenuItem("撤销");

redo = new JMenuItem("重做");

delete = new JMenuItem("删除");

deleteAll = new JMenuItem("清空");

search = new JMenuItem("查找");

replace = new JMenuItem("替换");

edit\_mode = new JMenuItem("编辑模式");

selectAll.setBackground(Color.cyan);

copy.setBackground(Color.cyan);

cut.setBackground(Color.cyan);

paste.setBackground(Color.cyan);

undo.setBackground(Color.cyan);

redo.setBackground(Color.cyan);

delete.setBackground(Color.yellow);

deleteAll.setBackground(Color.red);

search.setBackground(Color.cyan);

replace.setBackground(Color.cyan);

edit\_mode.setBackground(Color.green);

font\_setting = new JMenuItem("字体设置");

border = new JMenuItem("边框设置");

font\_color = new JMenuItem("字体颜色");

cursor\_color = new JMenuItem("光标颜色");

background\_color = new JMenuItem("背景颜色");

selected\_color = new JMenuItem("选中颜色");

rendering\_color = new JMenuItem("渲染颜色");

delete\_confirmation = new JMenuItem("清除配置");

font\_setting.setBackground(Color.green);

border.setBackground(Color.green);

font\_color.setBackground(Color.green);

cursor\_color.setBackground(Color.green);

background\_color.setBackground(Color.green);

selected\_color.setBackground(Color.green);

rendering\_color.setBackground(Color.green);

delete\_confirmation.setBackground(Color.red);

wrap = new JMenuItem("不自动换行");

encoding\_saveAs = new JMenuItem("使用用户指定的编码格式另存文件");

overload\_UTF\_8 = new JMenuItem("使用UTF-8编码格式重新加载文件");

overload\_UTF\_16LE = new JMenuItem("使用UTF-16LE编码格式重新加载文件");

overload\_UTF\_16BE = new JMenuItem("使用UTF-16BE编码格式重新加载文件");

overload\_GBK = new JMenuItem("使用GBK编码格式重新加载文件");

overload\_GB2312 = new JMenuItem("使用GB2312编码格式重新加载文件");

overload\_GB18030 = new JMenuItem("使用GB18030编码格式重新加载文件");

overload\_ISO\_8859\_1 = new JMenuItem("使用ISO-8859-1编码格式重新加载文件");

overload\_US\_ASCII = new JMenuItem("使用US-ASCII编码格式重新加载文件");

overload\_user\_definition = new JMenuItem("使用用户输入的编码格式重新加载文件");

wrap.setBackground(Color.cyan);

encoding\_saveAs.setBackground(Color.green);

overload\_UTF\_8.setBackground(Color.yellow);

overload\_UTF\_16LE.setBackground(Color.yellow);

overload\_UTF\_16BE.setBackground(Color.yellow);

overload\_GBK.setBackground(Color.yellow);

overload\_GB2312.setBackground(Color.yellow);

overload\_GB18030.setBackground(Color.yellow);

overload\_ISO\_8859\_1.setBackground(Color.yellow);

overload\_US\_ASCII.setBackground(Color.yellow);

overload\_user\_definition.setBackground(Color.yellow);

errorLog = new JMenuItem("错误日志");

instructions\_for\_use = new JMenuItem("使用说明");

about = new JMenuItem("关于");

errorLog.setBackground(Color.pink);

instructions\_for\_use.setBackground(Color.pink);

about.setBackground(Color.pink);

//文件

menu\_file.add(open);

menu\_file.add(save);

menu\_file.add(save\_as);

menu\_file.add(auto\_save);

menu\_file.add(file\_information);

menu\_file.add(auto\_clear);

menu\_file.add(exit);

//编辑

menu\_edit.add(selectAll);

menu\_edit.add(copy);

menu\_edit.add(cut);

menu\_edit.add(paste);

menu\_edit.add(undo);

menu\_edit.add(redo);

menu\_edit.add(delete);

menu\_edit.add(deleteAll);

menu\_edit.add(search);

menu\_edit.add(replace);

menu\_edit.add(edit\_mode);

//个性化

individualization.add(font\_setting);

individualization.add(border);

individualization.add(font\_color);

individualization.add(cursor\_color);

individualization.add(background\_color);

individualization.add(selected\_color);

individualization.add(rendering\_color);

individualization.add(delete\_confirmation);

//格式

format.add(wrap);

format.add(encoding\_saveAs);

format.add(overload\_UTF\_8);

format.add(overload\_UTF\_16LE);

format.add(overload\_UTF\_16BE);

format.add(overload\_GBK);

format.add(overload\_GB2312);

format.add(overload\_GB18030);

format.add(overload\_ISO\_8859\_1);

format.add(overload\_US\_ASCII);

format.add(overload\_user\_definition);

//帮助

help.add(errorLog);

help.add(instructions\_for\_use);

help.add(about);

// 将菜单和相应的子菜单添加到菜单栏

jMenuBar.add(menu\_file);

jMenuBar.add(menu\_edit);

jMenuBar.add(individualization);

jMenuBar.add(format);

jMenuBar.add(help);

// 添加菜单栏

jFrame.setJMenuBar(jMenuBar);

}

private void init\_timer\_auto\_save()

{

ActionListener taskPerformer = new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

//Toolkit.getDefaultToolkit().beep();

if (Configuration.config\_is\_not\_null) //保存配置文件

{

Configuration.config.setWidth(jFrame.getWidth());

Configuration.config.setHeight(jFrame.getHeight());

io.Configuration.write();

}

if (file != null)

{

MainPanel.this.save();

label\_Information.setText("已触发自动保存");

}

}

};

timer\_autoSave = new Timer(5000, taskPerformer);

}

private void init\_configuration()

{

if (Configuration.config\_is\_not\_null)

{

jFrame.setSize(Configuration.config.getWidth(), Configuration.config.getHeight()); //设置大小

int screenWidth = ((int) java.awt.Toolkit.getDefaultToolkit().getScreenSize().width); //屏幕分辨率

int screenHeight = ((int) java.awt.Toolkit.getDefaultToolkit().getScreenSize().height);

int x = screenWidth / 2 - Configuration.config.getWidth() / 2; //位于中心

int y = screenHeight / 2 - Configuration.config.getHeight() / 2;

if (x < 0)

{

x = 0;

}

if (y < 0)

{

y = 0;

}

jFrame.setLocation(x, y);

Font font = new Font(Configuration.config.getFontName(), //设置字体和各颜色

Configuration.config.getFontStyle(), Configuration.config.getFontSize());

jTextArea.setFont(font);

Color font\_color = new Color(Configuration.config.getFont\_color\_r(),

Configuration.config.getFont\_color\_g(), Configuration.config.getFont\_color\_b());

jTextArea.setForeground(font\_color);

Color cursor\_color = new Color(Configuration.config.getCursor\_color\_r()

, Configuration.config.getCursor\_color\_g(), Configuration.config.getCursor\_color\_b());

jTextArea.setCaretColor(cursor\_color);

Color background\_color = new Color(Configuration.config.getBackground\_color\_r()

, Configuration.config.getBackground\_color\_g(), Configuration.config.getBackground\_color\_b());

jTextArea.setBackground(background\_color);

Color selected\_color = new Color(Configuration.config.getSelected\_color\_r()

, Configuration.config.getSelected\_color\_g(), Configuration.config.getSelected\_color\_b());

jTextArea.setSelectedTextColor(selected\_color);

Color rendering\_color = new Color(Configuration.config.getRendering\_color\_r()

, Configuration.config.getRendering\_color\_g(), Configuration.config.getRendering\_color\_b());

jTextArea.setSelectionColor(rendering\_color);

/\*

System.out.println(screenWidth);

System.out.println(screenHeight);

System.out.println(Configuration.config.getWidth());

System.out.println(Configuration.config.getHeight());

\*/

if (screenWidth <= Configuration.config.getWidth() + 100 && screenHeight <= Configuration.config.getHeight() + 100)

{ //任务栏会占用一部分屏幕空间

jFrame.setExtendedState(JFrame.MAXIMIZED\_BOTH); //设置窗口最大化

jFrame.setSize(1280, 720);

int x1 = screenWidth / 2 - 1280 / 2; //位于中心

int y1 = screenHeight / 2 - 720 / 2;

jFrame.setLocation(x1, y1);

}

if (!Configuration.config.isWrap())

{

wrap.setBackground(Color.yellow);

wrap.setText("自动换行");

jTextArea.setLineWrap(false);

}

int Layout\_left = 30; //边框大小

int Layout\_right = 300;

int Layout\_up = 0;

int Layout\_down = 15;

Layout\_left = Configuration.config.getLayout\_left();

Layout\_right = Configuration.config.getLayout\_right();

Layout\_up = Configuration.config.getLayout\_up();

Layout\_down = Configuration.config.getLayout\_down();

jScrollPane.setBorder(new EmptyBorder(Layout\_up, Layout\_left, Layout\_down, Layout\_right));

if (Configuration.config.isAutoClear()) //为真

{

isAutoClear = true;

timer\_autoClear.start(); //启动

auto\_clear.setBackground(Color.yellow);

auto\_clear.setText("不自动清理");

}

}

}

public MainPanel() //构造方法

{

io.Configuration.read(); //读配置文件

jFrame = new JFrame("文本编辑器"); //初始化顶层面板

jFrame.setSize(1280, 720);

int screenWidth = Toolkit.getDefaultToolkit().getScreenSize().width; //获取屏幕宽度

int screenHeight = Toolkit.getDefaultToolkit().getScreenSize().height; //获取屏幕高度

jFrame.setLocation(screenWidth / 2 - 640, screenHeight / 2 - 360); //位于屏幕中央

jFrame.setDefaultCloseOperation(WindowConstants.DO\_NOTHING\_ON\_CLOSE);

this.init\_mainPanel(); //初始化主面板

this.init\_menu(); //初始化菜单面板

FileInformation.init(); //初始化文件信息面板

UI.ErrorLog.init\_error\_log\_jPanel(); //初始化错误日志面板

UI.Search.init\_search(jTextArea, label\_Information); //初始化查找面板

UI.Replace.init\_replace(jTextArea, label\_Information); //初始化替换面板

about\_software = new UI.About(); //初始化关于面板

instructionsForUse = new InstructionsForUse();//初始化使用说明面板

jTextArea\_border = new JTextArea\_Border(jTextArea, jScrollPane); //初始化边框设置模板

this.init\_timer\_auto\_save(); //初始化自动保存

this.init\_auto\_clear(); //初始化自动清理

fontSetting = new UI.FontSetting(jTextArea); //初始化字体设置面板

this.init\_configuration(); //初始化配置

Color\_JTextArea.init\_Color\_JTextArea //初始化文本域颜色选择

(jTextArea, font\_color, cursor\_color, background\_color, selected\_color, rendering\_color);

jFrame.add(jPanel); //主面板加入到顶层面板

jFrame.setVisible(true); //设置可见

io.File.args\_read(file, jTextArea,

label\_Information, jTextField\_FilePath); //初始化参数

this.init\_Listener(); //初始化各种监听器

this.init\_menu\_Listener(); //初始化菜单监听器

}

private void init\_menu\_Listener() //初始化菜单监听器

{

// 鼠标监听,弹出右键菜单

jTextArea.addMouseListener(new MouseAdapter()

{

public void mousePressed(MouseEvent e)

{

int mods = e.getModifiersEx();

//System.out.println(mods);

// 鼠标右键

if (mods == 4096)

{

// 弹出菜单

jPopupMenu.show(e.getComponent(), e.getX(), e.getY());

}

}

});

open.addActionListener(new ActionListener()

{ //菜单按钮 浏览

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.open();

}

});

save.addActionListener(new ActionListener()

{ //菜单按钮 保存

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.save();

}

});

save\_as.addActionListener(new ActionListener()

{ //菜单按钮 另存为

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.saveAs();

}

});

exit.addActionListener(new ActionListener()

{ //菜单按钮 退出

@Override

public void actionPerformed(ActionEvent e)

{

if (Configuration.config\_is\_not\_null) //保存配置文件

{

Configuration.config.setWidth(jFrame.getWidth());

Configuration.config.setHeight(jFrame.getHeight());

io.Configuration.write();

}

MainPanel.this.close();

}

});

selectAll.addActionListener(new ActionListener()

{ //菜单按钮 全选

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.selectAll();

}

});

selectAll\_pop.addActionListener(new ActionListener()

{ //菜单按钮 全选

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.selectAll();

}

});

copy.addActionListener(new ActionListener()

{ //菜单按钮 复制

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.copy();

}

});

copy\_pop.addActionListener(new ActionListener()

{ //菜单按钮 复制

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.copy();

}

});

cut.addActionListener(new ActionListener()

{ //菜单按钮 剪切

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.cut();

}

});

cut\_pop.addActionListener(new ActionListener()

{ //菜单按钮 剪切

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.cut();

}

});

paste.addActionListener(new ActionListener()

{ //菜单按钮 粘贴

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.paste();

}

});

paste\_pop.addActionListener(new ActionListener()

{ //菜单按钮 粘贴

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.paste();

}

});

delete.addActionListener(new ActionListener()

{ //菜单按钮 删除

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.delete();

}

});

delete\_pop.addActionListener(new ActionListener()

{ //菜单按钮 删除

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.delete();

}

});

deleteAll.addActionListener(new ActionListener()

{ //菜单按钮 清空

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.deleteAll();

}

});

deleteAll\_pop.addActionListener(new ActionListener()

{ //菜单按钮 清空

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.deleteAll();

}

});

edit\_mode.addActionListener(new ActionListener()

{ //编辑模式和只读模式来回切换

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.EditMode();

}

});

font\_setting.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

int x = MainPanel.getjFrame().getX();

int y = MainPanel.getjFrame().getY();

int width = MainPanel.getjFrame().getWidth();

int height = MainPanel.getjFrame().getHeight();

int search\_x = x + width / 2 - fontSetting.getWidth() / 2;

int search\_y = y + height / 2 - fontSetting.getHeight() / 2;

fontSetting.setLocation(search\_x, search\_y);

fontSetting.setVisible(true);

}

});

wrap.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

boolean result = jTextArea.getLineWrap();

if (result) //是换行状态

{

wrap.setBackground(Color.yellow);

wrap.setText("自动换行");

jTextArea.setLineWrap(false);

label\_Information.setText("当前为不自动换行模式");

if (io.Configuration.config == null)

{

io.Configuration.config = new data.Configuration();

Configuration.config\_is\_not\_null = true;

}

io.Configuration.config.setWrap(false);

}

else //不是换行状态

{

wrap.setBackground(Color.cyan);

wrap.setText("不自动换行");

jTextArea.setLineWrap(true);

label\_Information.setText("当前为自动换行模式");

if (io.Configuration.config == null)

{

io.Configuration.config = new data.Configuration();

Configuration.config\_is\_not\_null = true;

}

io.Configuration.config.setWrap(true);

}

}

});

about.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

int x = MainPanel.getjFrame().getX();

int y = MainPanel.getjFrame().getY();

int width = MainPanel.getjFrame().getWidth();

int height = MainPanel.getjFrame().getHeight();

int search\_x = x + width / 2 - about\_software.getWidth() / 2;

int search\_y = y + height / 2 - about\_software.getHeight() / 2;

about\_software.setLocation(search\_x, search\_y);

about\_software.setVisible(true);

}

});

auto\_save.addActionListener(new ActionListener()

{ //自动保存按钮

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.change\_auto\_save\_mode();

}

});

delete\_confirmation.addActionListener(new ActionListener()

{ //清除配置文件

@Override

public void actionPerformed(ActionEvent e)

{

io.Configuration.delete();

}

});

border.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

int x = MainPanel.getjFrame().getX();

int y = MainPanel.getjFrame().getY();

int width = MainPanel.getjFrame().getWidth();

int height = MainPanel.getjFrame().getHeight();

int search\_x = x + width / 2 - jTextArea\_border.getWidth() / 2;

int search\_y = y + height / 2 - jTextArea\_border.getHeight() / 2;

jTextArea\_border.setLocation(search\_x, search\_y);

jTextArea\_border.setVisible(true);

}

});

instructions\_for\_use.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

int x = MainPanel.getjFrame().getX();

int y = MainPanel.getjFrame().getY();

int width = MainPanel.getjFrame().getWidth();

int height = MainPanel.getjFrame().getHeight();

int search\_x = x + width / 2 - instructionsForUse.getWidth() / 2;

int search\_y = y + height / 2 - instructionsForUse.getHeight() / 2;

instructionsForUse.setLocation(search\_x, search\_y);

instructionsForUse.setVisible(true);

}

});

undo.addActionListener(new ActionListener()

{ //撤销监听器

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.undo();

}

});

undo\_pop.addActionListener(new ActionListener()

{ //撤销监听器

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.undo();

}

});

redo.addActionListener(new ActionListener()

{ //重做监听器

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.redo();

}

});

redo\_pop.addActionListener(new ActionListener()

{ //重做监听器

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.redo();

}

});

auto\_clear.addActionListener(new ActionListener()

{ //自动清理监听器

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.change\_auto\_clear\_mode();

}

});

encoding\_saveAs.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

if (jTextArea.getText().length() == 0)

{

label\_Information.setText("文本域为空,没必要保存");

return;

}

JFileChooser jFileChooser = new JFileChooser(".");

int result = jFileChooser.showSaveDialog(null);

if (result == JFileChooser.APPROVE\_OPTION)

{

String str;

str = JOptionPane.showInputDialog(null,

"请输入文件编码：", "", JOptionPane.QUESTION\_MESSAGE);

if (str == null || str.equals(""))

{

//Toolkit.getDefaultToolkit().beep();

label\_Information.setText("已取消输入编码，或者输入的编码为空！");

return;

}

//System.out.println(str);

File file = jFileChooser.getSelectedFile();

io.File.write(file, jTextArea, label\_Information, str); //写入文件

}

else

{

Toolkit.getDefaultToolkit().beep();

label\_Information.setText("已取消！！！");

}

}

});

overload\_UTF\_8.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

if (file == null)

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"还未指定文件路径！", "提示", JOptionPane.ERROR\_MESSAGE);

return;

}

int result;

Toolkit.getDefaultToolkit().beep();

result = JOptionPane.showConfirmDialog(null,

"此操作将会刷新文本域里的内容，是否继续？", "数据丢失警告！",

JOptionPane.YES\_NO\_OPTION, JOptionPane.WARNING\_MESSAGE);

if (result == 0)

{

io.File.read(file, jTextArea, label\_Information, "UTF-8");

}

}

});

overload\_UTF\_16LE.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

if (file == null)

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"还未指定文件路径！", "提示", JOptionPane.ERROR\_MESSAGE);

return;

}

int result;

Toolkit.getDefaultToolkit().beep();

result = JOptionPane.showConfirmDialog(null,

"此操作将会刷新文本域里的内容，是否继续？", "数据丢失警告！",

JOptionPane.YES\_NO\_OPTION, JOptionPane.WARNING\_MESSAGE);

if (result == 0)

{

io.File.read(file, jTextArea, label\_Information, "UTF-16LE");

}

}

});

overload\_UTF\_16BE.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

if (file == null)

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"还未指定文件路径！", "提示", JOptionPane.ERROR\_MESSAGE);

return;

}

int result;

Toolkit.getDefaultToolkit().beep();

result = JOptionPane.showConfirmDialog(null,

"此操作将会刷新文本域里的内容，是否继续？", "数据丢失警告！",

JOptionPane.YES\_NO\_OPTION, JOptionPane.WARNING\_MESSAGE);

if (result == 0)

{

io.File.read(file, jTextArea, label\_Information, "UTF-16BE");

}

}

});

overload\_GBK.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

if (file == null)

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"还未指定文件路径！", "提示", JOptionPane.ERROR\_MESSAGE);

return;

}

int result;

Toolkit.getDefaultToolkit().beep();

result = JOptionPane.showConfirmDialog(null,

"此操作将会刷新文本域里的内容，是否继续？", "数据丢失警告！",

JOptionPane.YES\_NO\_OPTION, JOptionPane.WARNING\_MESSAGE);

if (result == 0)

{

io.File.read(file, jTextArea, label\_Information, "GBK");

}

}

});

overload\_GB2312.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

if (file == null)

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"还未指定文件路径！", "提示", JOptionPane.ERROR\_MESSAGE);

return;

}

int result;

Toolkit.getDefaultToolkit().beep();

result = JOptionPane.showConfirmDialog(null,

"此操作将会刷新文本域里的内容，是否继续？", "数据丢失警告！",

JOptionPane.YES\_NO\_OPTION, JOptionPane.WARNING\_MESSAGE);

if (result == 0)

{

io.File.read(file, jTextArea, label\_Information, "GB2312");

}

}

});

overload\_GB18030.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

if (file == null)

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"还未指定文件路径！", "提示", JOptionPane.ERROR\_MESSAGE);

return;

}

int result;

Toolkit.getDefaultToolkit().beep();

result = JOptionPane.showConfirmDialog(null,

"此操作将会刷新文本域里的内容，是否继续？", "数据丢失警告！",

JOptionPane.YES\_NO\_OPTION, JOptionPane.WARNING\_MESSAGE);

if (result == 0)

{

io.File.read(file, jTextArea, label\_Information, "GB18030");

}

}

});

overload\_ISO\_8859\_1.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

if (file == null)

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"还未指定文件路径！", "提示", JOptionPane.ERROR\_MESSAGE);

return;

}

int result;

Toolkit.getDefaultToolkit().beep();

result = JOptionPane.showConfirmDialog(null,

"此操作将会刷新文本域里的内容，是否继续？", "数据丢失警告！",

JOptionPane.YES\_NO\_OPTION, JOptionPane.WARNING\_MESSAGE);

if (result == 0)

{

io.File.read(file, jTextArea, label\_Information, "ISO-8859-1");

}

}

});

overload\_US\_ASCII.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

if (file == null)

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"还未指定文件路径！", "提示", JOptionPane.ERROR\_MESSAGE);

return;

}

int result;

Toolkit.getDefaultToolkit().beep();

result = JOptionPane.showConfirmDialog(null,

"此操作将会刷新文本域里的内容，是否继续？", "数据丢失警告！",

JOptionPane.YES\_NO\_OPTION, JOptionPane.WARNING\_MESSAGE);

if (result == 0)

{

io.File.read(file, jTextArea, label\_Information, "US-ASCII");

}

}

});

overload\_user\_definition.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

if (file == null)

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"还未指定文件路径！", "提示", JOptionPane.ERROR\_MESSAGE);

return;

}

int result;

Toolkit.getDefaultToolkit().beep();

result = JOptionPane.showConfirmDialog(null,

"此操作将会刷新文本域里的内容，是否继续？", "数据丢失警告！",

JOptionPane.YES\_NO\_OPTION, JOptionPane.WARNING\_MESSAGE);

if (result == 0)

{

String str;

str = JOptionPane.showInputDialog(null,

"请输入文件编码：", "", JOptionPane.QUESTION\_MESSAGE);

if (str == null || str.equals(""))

{

//Toolkit.getDefaultToolkit().beep();

label\_Information.setText("已取消输入编码，或者输入的编码为空！");

return;

}

try

{

String s = "123";

s.getBytes(str); //测试编码是否正确

}

catch (UnsupportedEncodingException e1)

{

Toolkit.getDefaultToolkit().beep();

System.out.println("编码\"" + str + "\"无法识别！");

JOptionPane.showMessageDialog(null,

"编码\"" + str + "\"无法识别！\n 编码输入错误，或者该编码不支持！", "编码错误", JOptionPane.ERROR\_MESSAGE);

return;

}

io.File.read(file, jTextArea, label\_Information, str);

}

}

});

}

private void init\_Listener() //初始化各种监听器

{

jFrame.addWindowListener(new WindowListener()

{

@Override

public void windowOpened(WindowEvent e)

{

}

@Override

public void windowClosing(WindowEvent e)

{

if (Configuration.config\_is\_not\_null) //保存配置文件

{

Configuration.config.setWidth(jFrame.getWidth());

Configuration.config.setHeight(jFrame.getHeight());

io.Configuration.write();

}

MainPanel.this.close();

}

@Override

public void windowClosed(WindowEvent e)

{

}

@Override

public void windowIconified(WindowEvent e)

{

}

@Override

public void windowDeiconified(WindowEvent e)

{

}

@Override

public void windowActivated(WindowEvent e)

{

}

@Override

public void windowDeactivated(WindowEvent e)

{

}

});

jTextArea.addCaretListener(new CaretListener()

{ //实时获取文本域指针位置

@Override

public void caretUpdate(CaretEvent e)

{

MainPanel.this.jTextArea\_CaretListener();

}

});

button\_save\_file.addActionListener(new ActionListener()

{ //另存为

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.saveAs();

}

});

button\_Save.addActionListener(new ActionListener()

{ //保存

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.save();

}

});

button\_EditMode.addActionListener(new ActionListener()

{ //只读模式或者编辑模式的来回切换

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.EditMode();

}

});

button\_Open.addActionListener(new ActionListener()

{ //浏览

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.open();

}

});

button\_autoSave.addActionListener(new ActionListener()

{ //保存按钮

@Override

public void actionPerformed(ActionEvent e)

{

MainPanel.this.change\_auto\_save\_mode();

}

});

jTextArea.addKeyListener(new KeyAdapter()

{

@Override

public void keyPressed(KeyEvent e)

{ //ctrl+f 查找

if ((e.getKeyCode() == KeyEvent.VK\_F) && (e.isControlDown()))

{

UI.Search.search(jTextArea, label\_Information);

}

else if ((e.getKeyCode() == KeyEvent.VK\_G) && (e.isControlDown()))

{ //ctrl+g 替换

UI.Replace.replace(jTextArea, label\_Information);

}

else if ((e.getKeyCode() == KeyEvent.VK\_S) && (e.isControlDown()))

{ //ctrl+s 保存

MainPanel.this.save();

}

else if ((e.getKeyCode() == KeyEvent.VK\_S) && (e.isControlDown()) && (e.isShiftDown()))

{ //ctrl+shift+s 另存为

MainPanel.this.saveAs();

}

else if ((e.getKeyCode() == KeyEvent.VK\_O) && (e.isControlDown()))

{ //ctrl+o 浏览

MainPanel.this.open();

}

else if ((e.getKeyCode() == KeyEvent.VK\_I) && (e.isControlDown()))

{ //ctrl+i 文件信息

UI.FileInformation.display();

}

else if ((e.getKeyCode() == KeyEvent.VK\_E) && (e.isControlDown()))

{ //ctrl+e 错误日志

UI.ErrorLog.display();

}

//以下快捷键不能设置操作，和操作系统快捷键起冲突，否则会得到双倍快乐

else if ((e.getKeyCode() == KeyEvent.VK\_A) && (e.isControlDown()))

{ //ctrl+a 全选

label\_Information.setText("全选成功");

}

else if ((e.getKeyCode() == KeyEvent.VK\_C) && (e.isControlDown()))

{ //ctrl+c 复制

label\_Information.setText("复制成功");

}

else if ((e.getKeyCode() == KeyEvent.VK\_V) && (e.isControlDown()))

{ //ctrl+v 粘贴

label\_Information.setText("粘贴成功");

}

else if ((e.getKeyCode() == KeyEvent.VK\_X) && (e.isControlDown()))

{ //ctrl+x 剪切

label\_Information.setText("剪切成功");

}

else if ((e.getKeyCode() == KeyEvent.VK\_R) && (e.isControlDown()))

{ //ctrl+r 模式切换

MainPanel.this.EditMode();

}

else if ((e.getKeyCode() == KeyEvent.VK\_F4) && (e.isControlDown()))

{ //ctrl+f4 更改清理内存模式

MainPanel.this.change\_auto\_clear\_mode();

}

else if ((e.getKeyCode() == KeyEvent.VK\_F4))

{ //f4 清理内存

System.gc();

label\_Information.setText("已清理软件内存");

}

else if ((e.getKeyCode() == KeyEvent.VK\_F3))

{ //f3 改变自动保存模式

MainPanel.this.change\_auto\_save\_mode();

}

else if ((e.getKeyCode() == KeyEvent.VK\_Z) && (e.isControlDown()))

{ //ctrl+z 撤销

MainPanel.this.undo();

}

else if ((e.getKeyCode() == KeyEvent.VK\_W) && (e.isControlDown()))

{ //ctrl+w 重做

MainPanel.this.redo();

}

}

});

}

private void close() //关闭程序

{

if (jTextArea.getText().length() == 0)

{

System.exit(1);

}

String fileMD5 = null;

String testAreaMD5 = null;

if (file != null)

{

label\_Information.setText("请稍后，正在计算MD5值...");

fileMD5 = MD5.getFileMD5(file.getAbsolutePath());

testAreaMD5 = MD5.getMD5API(jTextArea.getText());

label\_Information.setText("MD5值计算完成");

}

if (file == null)

{

int result;

Toolkit.getDefaultToolkit().beep();

result = JOptionPane.showConfirmDialog(null, "文本还未保存！ 是否退出？",

"退出提示", JOptionPane.YES\_NO\_OPTION, JOptionPane.ERROR\_MESSAGE);

if (result == JOptionPane.YES\_OPTION)

{

System.exit(1);

}

else if (result == 1)

{

label\_Information.setText("取消退出");

}

else

{

label\_Information.setText("关闭会话框，取消退出");

}

}

else if (fileMD5.equals(testAreaMD5)) //MD5值相同,直接退出

{

System.exit(1);

}

else if (fileMD5 == null || testAreaMD5 == null)

{

int result;

Toolkit.getDefaultToolkit().beep();

result = JOptionPane.showConfirmDialog(null,

"无法计算MD5值！ 是否退出？\n文件MD5：" + fileMD5 + "\n文本域MD5:" + testAreaMD5,

"退出提示", JOptionPane.YES\_NO\_OPTION, JOptionPane.ERROR\_MESSAGE);

if (result == JOptionPane.YES\_OPTION)

{

System.exit(1);

}

else if (result == 1)

{

label\_Information.setText("取消退出");

}

else

{

label\_Information.setText("关闭会话框，取消退出");

}

}

else

{

int result;

Toolkit.getDefaultToolkit().beep();

result = JOptionPane.showConfirmDialog(null,

"文本还有一部分未保存！ 是否退出？\n文件MD5：" + fileMD5 + "\n文本域MD5:" + testAreaMD5,

"退出提示", JOptionPane.YES\_NO\_OPTION, JOptionPane.ERROR\_MESSAGE);

if (result == JOptionPane.YES\_OPTION)

{

System.exit(1);

}

else if (result == 1)

{

label\_Information.setText("取消退出");

}

else

{

label\_Information.setText("关闭会话框，取消退出");

}

}

}

private void jTextArea\_CaretListener() //实时获取文本域指针位置

{

try

{

int pos = jTextArea.getCaretPosition();

//获取行数

int lineOfC = 0;

lineOfC = jTextArea.getLineOfOffset(pos) + 1;

//获取列数

int col = pos - jTextArea.getLineStartOffset(lineOfC - 1) + 1;

//System.out.println("当前光标位置:" + lineOfC + "行," + col + "列");

label\_Information.setText("当前光标位置：第" + lineOfC + "行,第" + col + "列");

}

catch (BadLocationException e1)

{

System.out.println("无法获取光标位置");

label\_Information.setText("无法获取光标位置");

//e1.printStackTrace();

}

}

private void saveAs() //另存为

{

if (jTextArea.getText().length() == 0)

{

label\_Information.setText("文本域为空,没必要保存");

return;

}

JFileChooser jFileChooser = new JFileChooser(".");

int result = jFileChooser.showSaveDialog(null);

if (result == JFileChooser.APPROVE\_OPTION)

{

File file = jFileChooser.getSelectedFile();

if (MainPanel.file == null)

{

MainPanel.file = file;

jTextField\_FilePath.setText(MainPanel.file.getAbsolutePath());

}

io.File.write(file, jTextArea, label\_Information); //写入文件

}

else

{

Toolkit.getDefaultToolkit().beep();

label\_Information.setText("未成功保存！！！");

}

}

private void save() //保存

{

if (jTextArea.getText().length() == 0)

{

label\_Information.setText("文本域为空,没必要保存");

return;

}

if (MainPanel.file == null)

{

JFileChooser jFileChooser = new JFileChooser(".");

int result = jFileChooser.showSaveDialog(null);

if (result == JFileChooser.APPROVE\_OPTION)

{

File file = jFileChooser.getSelectedFile();

MainPanel.file = file;

jTextField\_FilePath.setText(MainPanel.file.getAbsolutePath());

io.File.write(file, jTextArea, label\_Information); //写入文件

}

else

{

Toolkit.getDefaultToolkit().beep();

label\_Information.setText("未成功保存！！！");

}

}

else

{

io.File.write(jTextArea, label\_Information); //写入文件

}

}

private void EditMode() //只读模式或者编辑模式的来回切换

{

if (isEditable)

{

button\_EditMode.setText("只读模式");

edit\_mode.setText("只读模式");

button\_EditMode.setBackground(Color.yellow);

edit\_mode.setBackground(Color.yellow);

isEditable = false;

jTextArea.setEditable(false);

label\_Information.setText("当前为只读模式");

}

else

{

button\_EditMode.setText("编辑模式");

edit\_mode.setText("编辑模式");

button\_EditMode.setBackground(Color.green);

edit\_mode.setBackground(Color.green);

isEditable = true;

jTextArea.setEditable(true);

label\_Information.setText("当前为编辑模式");

}

}

private void open() //打开或者浏览

{

if (jTextArea.getText().length() != 0)

{

String[] selection = {"文件数据插入到文本域的后面", "使用文件里的数据替换文本域里的数据"};

Toolkit.getDefaultToolkit().beep();

int result;

result = JOptionPane.showOptionDialog(null, "文本域数据不为空！请选择更新模式！"

, "警告", JOptionPane.YES\_NO\_OPTION, JOptionPane.ERROR\_MESSAGE, null, selection, 0);

if (result == 0)

{

label\_Information.setText("从第" + (jTextArea.getText().length() - 1) + "个位置插入文件数据");

}

else if (result == 1)

{

jTextArea.setText("");

label\_Information.setText("文本域原来的数据已丢失");

}

else //按到了关闭按钮

{

label\_Information.setText("取消操作");

return;

}

}

JFileChooser jFileChooser = new JFileChooser(".");

int result = jFileChooser.showOpenDialog(null);

if (result == JFileChooser.APPROVE\_OPTION)

{

jTextField\_FilePath.setText(jFileChooser.getSelectedFile().toString());

file = jFileChooser.getSelectedFile();

label\_Information.setText("正在加载...");

io.File.read(file, jTextArea, label\_Information);

}

else

{

Toolkit.getDefaultToolkit().beep();

label\_Information.setText("未选择文件！！！");

}

}

private void selectAll() //全选

{

if (jTextArea.getText().length() == 0)

{

Toolkit.getDefaultToolkit().beep();

label\_Information.setText("全选失败！ 文本域为空！");

}

else

{

jTextArea.selectAll();

int start = jTextArea.getSelectionStart();

int end = jTextArea.getSelectionEnd();

label\_Information.setText("全选成功, 选中位置为" + start + "到" + end + "的文本");

}

}

private void copy() //复制

{

if (jTextArea.getSelectedText() == null)

{

Toolkit.getDefaultToolkit().beep();

label\_Information.setText("复制失败！ 未选择文字");

}

else

{

jTextArea.copy();

int start = jTextArea.getSelectionStart();

int end = jTextArea.getSelectionEnd();

label\_Information.setText("复制成功, 复制选中位置为" + start + "到" + end + "的文本");

}

}

private void cut() //剪切

{

if (jTextArea.getSelectedText() == null)

{

Toolkit.getDefaultToolkit().beep();

label\_Information.setText("剪切失败！ 未选择文字");

}

else

{

int start = jTextArea.getSelectionStart();

int end = jTextArea.getSelectionEnd();

jTextArea.cut();

label\_Information.setText("剪切成功, 剪切选中位置为" + start + "到" + end + "的文本");

}

}

private void paste() //粘贴

{

jTextArea.paste();

label\_Information.setText("粘贴成功");

}

private void delete() //删除

{

if (jTextArea.getSelectedText() == null)

{

Toolkit.getDefaultToolkit().beep();

label\_Information.setText("删除失败！ 未选择如何文字！");

}

else

{

jTextArea.replaceSelection("");

}

}

private void deleteAll() //清空

{

if (jTextArea.getText().length() == 0)

{

label\_Information.setText("文本域已经清空 无法再清空");

}

else

{

int result;

Toolkit.getDefaultToolkit().beep();

result = JOptionPane.showConfirmDialog(null, "是否清空文本域的所有数据？"

, "数据丢失警告", JOptionPane.YES\_NO\_OPTION, JOptionPane.ERROR\_MESSAGE);

if (result == 0)

{

jTextArea.setText(""); //清空操作

label\_Information.setText("文本域已清空");

}

else if (result == 1)

{

label\_Information.setText("取消清空");

}

else

{

label\_Information.setText("关闭会话框，取消清空");

}

}

}

private void change\_auto\_save\_mode() //改变自动保存模式

{

if (auto\_save\_mode == 0) //当前为不自动保存

{ //改成600s

auto\_save\_mode = 1;

timer\_autoSave.setDelay(600000);

timer\_autoSave.start();

button\_autoSave.setText("自动保存:10min");

button\_autoSave.setBackground(Color.green);

auto\_save.setText("自动保存：10min");

auto\_save.setBackground(Color.green);

label\_Information.setText("自动保存设置成10分钟");

}

else if (auto\_save\_mode == 1) //600s

{

auto\_save\_mode = 2; //改成5分钟

timer\_autoSave.setDelay(300000);

timer\_autoSave.stop();

timer\_autoSave.start();

button\_autoSave.setText("自动保存:5min");

button\_autoSave.setBackground(Color.green);

auto\_save.setText("自动保存：5min");

auto\_save.setBackground(Color.green);

label\_Information.setText("自动保存设置成5分钟");

}

else if (auto\_save\_mode == 2) //300s

{

auto\_save\_mode = 3; //改成4分钟

timer\_autoSave.setDelay(240000);

timer\_autoSave.stop();

timer\_autoSave.start();

button\_autoSave.setText("自动保存:4min");

button\_autoSave.setBackground(Color.green);

auto\_save.setText("自动保存:4min");

auto\_save.setBackground(Color.green);

label\_Information.setText("自动保存设置成4分钟");

}

else if (auto\_save\_mode == 3) //240s

{

auto\_save\_mode = 4; //改成3分钟

timer\_autoSave.stop();

timer\_autoSave.setDelay(180000);

timer\_autoSave.start();

button\_autoSave.setText("自动保存:3min");

button\_autoSave.setBackground(Color.green);

auto\_save.setText("自动保存：3min");

auto\_save.setBackground(Color.green);

label\_Information.setText("自动保存设置成3分钟");

}

else if (auto\_save\_mode == 4) //180s

{

auto\_save\_mode = 5; //改成2分钟

timer\_autoSave.stop();

timer\_autoSave.setDelay(120000);

timer\_autoSave.start();

button\_autoSave.setText("自动保存:2min");

button\_autoSave.setBackground(Color.green);

auto\_save.setText("自动保存：2min");

auto\_save.setBackground(Color.green);

label\_Information.setText("自动保存设置成2分钟");

}

else if (auto\_save\_mode == 5) //120s

{

auto\_save\_mode = 6; //改成90s

timer\_autoSave.stop();

timer\_autoSave.setDelay(90000);

timer\_autoSave.start();

button\_autoSave.setText("自动保存：90s");

button\_autoSave.setBackground(Color.green);

auto\_save.setText("自动保存：90s");

auto\_save.setBackground(Color.green);

label\_Information.setText("自动保存设置成90秒");

}

else if (auto\_save\_mode == 6) //90s

{

auto\_save\_mode = 7; //改成60s

timer\_autoSave.stop();

timer\_autoSave.setDelay(60000);

timer\_autoSave.start();

button\_autoSave.setText("自动保存：60s");

button\_autoSave.setBackground(Color.green);

auto\_save.setText("自动保存：60s");

auto\_save.setBackground(Color.green);

label\_Information.setText("自动保存设置成60秒");

}

else if (auto\_save\_mode == 7) //60s

{

auto\_save\_mode = 8; //改成30s

timer\_autoSave.stop();

timer\_autoSave.setDelay(30000);

timer\_autoSave.start();

button\_autoSave.setText("自动保存：30s");

button\_autoSave.setBackground(Color.green);

auto\_save.setText("自动保存：30s");

auto\_save.setBackground(Color.green);

label\_Information.setText("自动保存设置成30秒");

}

else if (auto\_save\_mode == 8) //30s

{

auto\_save\_mode = 0; //关闭自动保存

button\_autoSave.setText("不自动保存");

timer\_autoSave.stop();

button\_autoSave.setBackground(Color.cyan);

auto\_save.setText("不自动保存");

auto\_save.setBackground(Color.cyan);

label\_Information.setText("已关闭自动保存");

}

}

private void undo() //撤销

{

if (undoManager.canUndo())

{

undoManager.undo();

label\_Information.setText("已撤销");

}

else

{

label\_Information.setText("撤销失败！");

}

}

private void redo() //重做

{

if (undoManager.canRedo())

{

undoManager.redo();

label\_Information.setText("已重做");

}

else

{

label\_Information.setText("重做失败");

}

}

private void init\_auto\_clear() //初始化自动清理

{

ActionListener taskPerformer = new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

System.gc();

}

};

timer\_autoClear = new Timer(10000, taskPerformer);

}

private void change\_auto\_clear\_mode() //改变自动清理模式

{

if (isAutoClear) //当前为自动清理模式

{

isAutoClear = false;

timer\_autoClear.stop(); //停止

if (io.Configuration.config == null) //如果对象不存在就创建对象

{

io.Configuration.config = new data.Configuration();

Configuration.config\_is\_not\_null = true;

}

Configuration.config.setAutoClear(false);

auto\_clear.setBackground(Color.green);

auto\_clear.setText("自动清理");

label\_Information.setText("已取消自动清理内存");

}

else //当前为不自动清理模式

{

isAutoClear = true;

timer\_autoClear.start(); //启动

if (io.Configuration.config == null) //如果对象不存在就创建对象

{

io.Configuration.config = new data.Configuration();

Configuration.config\_is\_not\_null = true;

}

Configuration.config.setAutoClear(true);

auto\_clear.setBackground(Color.yellow);

auto\_clear.setText("不自动清理");

label\_Information.setText("开始自动清理内存");

}

}

}

### 6.4.9. Replace

package UI;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

/\*\*

\* Project name(项目名称)：java课程设计 Swing实现文本编辑器

\* Package(包名): UI

\* Class(类名): Replace

\* Author(作者）: mao

\* Author QQ：1296193245

\* GitHub：https://github.com/maomao124/

\* Date(创建日期)： 2021/12/8

\* Time(创建时间)： 16:39

\* Version(版本): 1.0

\* Description(描述)： 替换

\*/

public class Replace

{

private static int start = 0; // 查找开始位置

private static int end = 0; // 查找结束位置

public static void init\_replace(JTextArea jTextArea, JLabel label\_information)

{

MainPanel.getReplace().addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

// 替换对话框

JDialog jDialog\_search = new JDialog(MainPanel.getjFrame(), "替换");

//JDialog:用于创建对话窗口的主类

jDialog\_search.setSize(500, 150);

int x = MainPanel.getjFrame().getX();

int y = MainPanel.getjFrame().getY();

int width = MainPanel.getjFrame().getWidth();

int height = MainPanel.getjFrame().getHeight();

int search\_x = x + width / 2 - 500 / 2;

int search\_y = y + height / 2 - 150 / 2;

jDialog\_search.setLocation(search\_x, search\_y); //位于中心

JLabel label\_search = new JLabel("查找的内容");

label\_search.setHorizontalAlignment(0);

JLabel label\_Replace = new JLabel("替换为");

label\_Replace.setHorizontalAlignment(0);

final JTextField textField\_search = new JTextField(20);

final JTextField textField\_Replace = new JTextField(20);

JButton buttonFind = new JButton("查找下一个");

JButton buttonReplace = new JButton("替换查找选中的内容");

JPanel panel = new JPanel(new GridLayout(2, 3));

buttonFind.setBackground(Color.cyan);

buttonReplace.setBackground(Color.cyan);

panel.add(label\_search);

panel.add(textField\_search);

panel.add(buttonFind);

panel.add(label\_Replace);

panel.add(textField\_Replace);

panel.add(buttonReplace);

jDialog\_search.add(panel);

if (jTextArea.getSelectedText() != null) //如果选中了文字

{

textField\_search.setText(jTextArea.getSelectedText());//从选中处开始

start = jTextArea.getSelectionStart();

end = jTextArea.getSelectionEnd();

}

jDialog\_search.setVisible(true);

buttonFind.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

String findText = textField\_search.getText(); // 查找的字符串

String textArea = jTextArea.getText(); // 当前文本框的内容

start = textArea.indexOf(findText, end);

end = start + findText.length();

if (start == -1) // 没有找到

{

//根据本机系统设置和硬件功能发出音频哔声

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"已经到达文档尾部", "提示", JOptionPane.WARNING\_MESSAGE);

//选择指定开始和结束位置之间的文本。

jTextArea.select(start, end);

start = 0;

end = 0;

}

else

{

jTextArea.select(start, end);

}

}

});

buttonReplace.addActionListener(new ActionListener()

{ //替换监听器

@Override

public void actionPerformed(ActionEvent e)

{

String ReplaceText = textField\_Replace.getText();

jTextArea.select(start, end);

jTextArea.replaceSelection(ReplaceText);

jTextArea.select(start, end);

label\_information.setText("替换成功 \"" + textField\_search.getText() + "\"替换成\"" + ReplaceText + "\"");

}

});

}

});

}

public static void replace(JTextArea jTextArea, JLabel label\_information)

{

// 替换对话框

JDialog jDialog\_search = new JDialog(MainPanel.getjFrame(), "替换");

jDialog\_search.setSize(500, 150);

int x = MainPanel.getjFrame().getX();

int y = MainPanel.getjFrame().getY();

int width = MainPanel.getjFrame().getWidth();

int height = MainPanel.getjFrame().getHeight();

int search\_x = x + width / 2 - 500 / 2;

int search\_y = y + height / 2 - 150 / 2;

jDialog\_search.setLocation(search\_x, search\_y);

JLabel label\_search = new JLabel("查找的内容");

label\_search.setHorizontalAlignment(0);

JLabel label\_Replace = new JLabel("替换为");

label\_Replace.setHorizontalAlignment(0);

final JTextField textField\_search = new JTextField(20);

final JTextField textField\_Replace = new JTextField(20);

JButton buttonFind = new JButton("查找下一个");

JButton buttonReplace = new JButton("替换查找选中的内容");

JPanel panel = new JPanel(new GridLayout(2, 3));

buttonFind.setBackground(Color.cyan);

buttonReplace.setBackground(Color.cyan);

panel.add(label\_search);

panel.add(textField\_search);

panel.add(buttonFind);

panel.add(label\_Replace);

panel.add(textField\_Replace);

panel.add(buttonReplace);

jDialog\_search.add(panel);

if (jTextArea.getSelectedText() != null) //如果选中了文字

{

textField\_search.setText(jTextArea.getSelectedText()); //从选中处开始

start = jTextArea.getSelectionStart(); //返回选定文本的开始位置。 对于空文档返回 0，如果没有选择则返回 dot 的值。

end = jTextArea.getSelectionEnd(); //返回选定文本的结束位置。 如果文档为空，则返回 0，如果没有选择，则返回 dot 的值。

}

jDialog\_search.setVisible(true);

buttonFind.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

String findText = textField\_search.getText(); // 查找的字符串

String textArea = jTextArea.getText(); // 当前文本框的内容

start = textArea.indexOf(findText, end); //返回此字符串中第一次出现指定子字符串的索引，从指定索引开始。

end = start + findText.length();

if (start == -1) // 没有找到

{

//根据本机系统设置和硬件功能发出音频哔声

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"已经到达文档尾部", "提示", JOptionPane.WARNING\_MESSAGE);

//选择指定开始和结束位置之间的文本

jTextArea.select(start, end);

start = 0;

end = 0;

}

else

{

//选择指定开始和结束位置之间的文本

jTextArea.select(start, end);

}

}

});

buttonReplace.addActionListener(new ActionListener()

{ //替换监听器

@Override

public void actionPerformed(ActionEvent e)

{

String ReplaceText = textField\_Replace.getText();

//选择指定开始和结束位置之间的文本。

//此方法设置所选文本的开始和结束位置

jTextArea.select(start, end);

//用给定字符串表示的新内容替换当前选择的内容

jTextArea.replaceSelection(ReplaceText);

//选择指定开始和结束位置之间的文本

jTextArea.select(start, end);

label\_information.setText("替换成功 \"" + textField\_search.getText() + "\"替换成\"" + ReplaceText + "\"");

}

});

}

}

### 6.4.10. Search

package UI;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

/\*\*

\* Project name(项目名称)：java课程设计 Swing实现文本编辑器

\* Package(包名): UI

\* Class(类名): Search

\* Author(作者）: mao

\* Author QQ：1296193245

\* GitHub：https://github.com/maomao124/

\* Date(创建日期)： 2021/12/8

\* Time(创建时间)： 10:52

\* Version(版本): 1.0

\* Description(描述)： 查找面板

\*/

public class Search

{

private static int start = 0; // 查找开始位置

private static int end = 0; // 查找结束位置

public static void init\_search(JTextArea jTextArea, JLabel label\_information)

{

MainPanel.getSearch().addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

// 查找对话框

JDialog jDialog\_search = new JDialog(MainPanel.getjFrame(), "查找");

jDialog\_search.setSize(500, 150);

int x = MainPanel.getjFrame().getX();

int y = MainPanel.getjFrame().getY();

//System.out.println(x+" "+y);

int width = MainPanel.getjFrame().getWidth();

int height = MainPanel.getjFrame().getHeight();

int search\_x = x + width / 2 - 500 / 2;

int search\_y = y + height / 2 - 150 / 2;

jDialog\_search.setLocation(search\_x, search\_y);

JLabel label\_search = new JLabel("查找的内容");

label\_search.setFont(new Font("宋体", Font.BOLD, 15));

final JTextField textField\_search = new JTextField(10);

JButton buttonFind = new JButton("查找下一个");

buttonFind.setBackground(Color.cyan);

JPanel panel = new JPanel(new FlowLayout(FlowLayout.CENTER, 5, 5));

JPanel panel1 = new JPanel(new BorderLayout());

JPanel panel2 = new JPanel(new FlowLayout(FlowLayout.CENTER, 5, 5));

panel.add(label\_search);

panel2.add(buttonFind);

panel1.add(panel, BorderLayout.NORTH);

panel1.add(panel2, BorderLayout.SOUTH);

panel1.add(textField\_search, BorderLayout.CENTER);

jDialog\_search.add(panel1);

jDialog\_search.setVisible(true);

// 查找下一个按钮监听器

buttonFind.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

String findText = textField\_search.getText(); // 查找的字符串

String textArea = jTextArea.getText(); // 当前文本框的内容

start = textArea.indexOf(findText, end);

end = start + findText.length();

// 没有找到

if (start == -1)

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"已经到达文档尾部", "提示", JOptionPane.WARNING\_MESSAGE);

jTextArea.select(start, end);

start = 0;

end = 0;

}

else //找到了

{

jTextArea.select(start, end);

label\_information.setText("查找成功 当前位置：" + start + "-" + end);

}

}

});

}

});

}

public static void search(JTextArea jTextArea, JLabel label\_information)

{

// 查找对话框

JDialog jDialog\_search = new JDialog(MainPanel.getjFrame(), "查找");

jDialog\_search.setSize(500, 150);

int x = MainPanel.getjFrame().getX();

int y = MainPanel.getjFrame().getY();

//System.out.println(x+" "+y);

int width = MainPanel.getjFrame().getWidth();

int height = MainPanel.getjFrame().getHeight();

int search\_x = x + width / 2 - 500 / 2;

int search\_y = y + height / 2 - 150 / 2;

jDialog\_search.setLocation(search\_x, search\_y);

JLabel label\_search = new JLabel("查找的内容");

label\_search.setFont(new Font("宋体", Font.BOLD, 15));

final JTextField textField\_search = new JTextField(10);

JButton buttonFind = new JButton("查找下一个");

buttonFind.setBackground(Color.cyan);

JPanel panel = new JPanel(new FlowLayout(FlowLayout.CENTER, 5, 5));

JPanel panel1 = new JPanel(new BorderLayout());

JPanel panel2 = new JPanel(new FlowLayout(FlowLayout.CENTER, 5, 5));

panel.add(label\_search);

panel2.add(buttonFind);

panel1.add(panel, BorderLayout.NORTH);

panel1.add(panel2, BorderLayout.SOUTH);

panel1.add(textField\_search, BorderLayout.CENTER);

jDialog\_search.add(panel1);

jDialog\_search.setVisible(true);

// 查找下一个按钮监听器

buttonFind.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

String findText = textField\_search.getText(); // 查找的字符串

String textArea = jTextArea.getText(); // 当前文本框的内容

start = textArea.indexOf(findText, end);

end = start + findText.length();

// 没有找到

if (start == -1)

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"已经到达文档尾部", "提示", JOptionPane.WARNING\_MESSAGE);

//选择指定开始和结束位置之间的文本。

jTextArea.select(start, end);

start = 0; //清0

end = 0;

}

else //找到了

{

//选择指定开始和结束位置之间的文本。

jTextArea.select(start, end);

label\_information.setText("查找成功 当前位置：" + start + "-" + end);

}

}

});

}

}

## 6.5. Run

import UI.MainPanel;

import io.File;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.PrintWriter;

import java.io.StringWriter;

import java.io.Writer;

import java.text.DecimalFormat;

import java.text.SimpleDateFormat;

import java.util.Date;

/\*\*

\* Project name(项目名称)：java课程设计 Swing实现文本编辑器

\* Package(包名): PACKAGE\_NAME

\* Class(类名): Run

\* Author(作者）: mao

\* Author QQ：1296193245

\* GitHub：https://github.com/maomao124/

\* Date(创建日期)： 2021/12/7

\* Time(创建时间)： 12:58

\* Version(版本): 1.0

\* Description(描述)： 从这里启动整个程序

\*/

public class Run

{

private static long runTime = 0;

//public static boolean args\_filePath = false; //匿名包里其它包里的类无法调用匿名包里的

static DecimalFormat decimalFormat = new DecimalFormat("00"); //yyyy/MM/dd HH:mm:ss

private static final SimpleDateFormat simpleDateFormat = new SimpleDateFormat("HH:mm:ss");

private static void init\_MemoryComputing()

{

ActionListener taskPerformer = new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

Runtime r = Runtime.getRuntime();

float memory;

memory = r.totalMemory();

memory = memory / 1024 / 1024;

//System.out.printf("JVM总内存：%.3fMB\n", memory);

memory = r.freeMemory();

memory = memory / 1024 / 1024;

//System.out.printf(" 空闲内存：%.3fMB\n", memory);

memory = r.totalMemory() - r.freeMemory();

memory = memory / 1024 / 1024;

runTime = runTime + 1;

System.out.print("运行时长：" + runTime / 60 + "分" + decimalFormat.format(runTime % 60) + "秒 ");

System.out.printf("已使用的内存：%8.4fMB\n", memory);

MainPanel.label\_time\_and\_memory.setText("运行：" + runTime / 60 + "分" +

decimalFormat.format(runTime % 60) + "秒 " + String.format(" 内存：%8.3fMB", memory));

MainPanel.label\_localTime.setText("时间："+simpleDateFormat.format(new Date()));

/\*

if (Configuration.config\_is\_not\_null)

{

System.out.println(Configuration.config);

}

\*/

}

};

Timer timer = new Timer(1000, taskPerformer);

timer.start();

}

private static void init\_args(String[] args) //处理参数

{

if (args.length == 0)

{

return;

}

else if (args.length == 1) //第一个参数为操作系统传入的要打开的文件路径

{

java.io.File file = new java.io.File(args[0]);

if (!file.exists())

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"文件\"" + file.getName() + "\"不存在！", "参数传入错误", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!file.isFile())

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"传入的路径指向的不是一个文件！", "提示", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!file.canRead())

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"文件\"" + file.getName() + "\"不能读取！", "提示", JOptionPane.ERROR\_MESSAGE);

return;

}

if (file.length() > 1000000000)

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"传入的路径指向的文件过于庞大！", "提示", JOptionPane.QUESTION\_MESSAGE);

return;

}

//通过验证，开始处理

MainPanel.setFile(file);

//args\_filePath = true;

}

else if (args.length == 2) //有些情况第二个才是传入的要打开的文件路径

{ //第一个参数是操作系统传入的程序文件本身所在的的路径，第二个才是，c/c++就是这样的

java.io.File file = new java.io.File(args[1]);

if (!file.exists())

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"文件\"" + file.getName() + "\"不存在！", "参数传入错误", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!file.isFile())

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"传入的路径指向的不是一个文件！", "提示", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!file.canRead())

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"文件\"" + file.getName() + "\"不能读取！", "提示", JOptionPane.ERROR\_MESSAGE);

return;

}

if (file.length() > 1000000000)

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"传入的路径指向的文件过于庞大！", "提示", JOptionPane.QUESTION\_MESSAGE);

return;

}

//通过验证，开始处理

MainPanel.setFile(file);

//args\_filePath = true;

}

else

{

JOptionPane.showMessageDialog(null, "因为传入了多个参数，所以只处理第二个传入的参数");

java.io.File file = new java.io.File(args[1]);

if (!file.exists())

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"文件\"" + file.getName() + "\"不存在！", "参数传入错误", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!file.isFile())

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"传入的路径指向的不是一个文件！", "提示", JOptionPane.ERROR\_MESSAGE);

return;

}

if (!file.canRead())

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"文件\"" + file.getName() + "\"不能读取！", "提示", JOptionPane.ERROR\_MESSAGE);

return;

}

if (file.length() > 1000000000)

{

Toolkit.getDefaultToolkit().beep();

JOptionPane.showMessageDialog(null,

"传入的路径指向的文件过于庞大！", "提示", JOptionPane.QUESTION\_MESSAGE);

return;

}

//通过验证，开始处理

MainPanel.setFile(file);

//args\_filePath = true;

}

}

public static void main(String[] args)

{

try

{

init\_args(args);

init\_MemoryComputing();

new MainPanel();

}

catch (Exception e)

{

e.printStackTrace();

final Writer result = new StringWriter();

final PrintWriter printWriter = new PrintWriter(result);

e.printStackTrace(printWriter);

String stackTraceStr = result.toString();

io.ErrorLog.write(stackTraceStr);

}

}

}

# 计算机与通信学院课程设计评分表

课题名称：

|  |  |
| --- | --- |
| **项 目** | **评 价** |
| 设计方案的合理性与创造性 |  |
| 设计与调试结果 |  |
| 设计说明书的质量 |  |
| 答辩陈述与回答问题情况 |  |
| 课程设计周表现情况 |  |
| 综合成绩 |  |

教师签名：

日 期：