目录

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</artifact
Id>
    <!--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)).ap
pend(":").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(StandardChar
sets.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：_____