核心流程设计分析

在第一次作业已经提供过类图，这里先不提供了，这次主要在第一次的基础上，对核心流程进行分析，说明其实现方法。

首先是笔记的展示

专门使用一个类（note_frame）用来展示整个的笔记图形界面，我目前是使用了`java`的`swing`来实现。 目前设计展示如下图所示，在这个类中，加入了所有我需要的功能的监听器，具体如下：

对于需要实现的不同功能，直接在响应方法中调用需要的对象及其方法即可。

- 四个菜单：
  - 文件：

    > 包含新建，打开，保存，另存为，设置和关闭。

  - 编辑

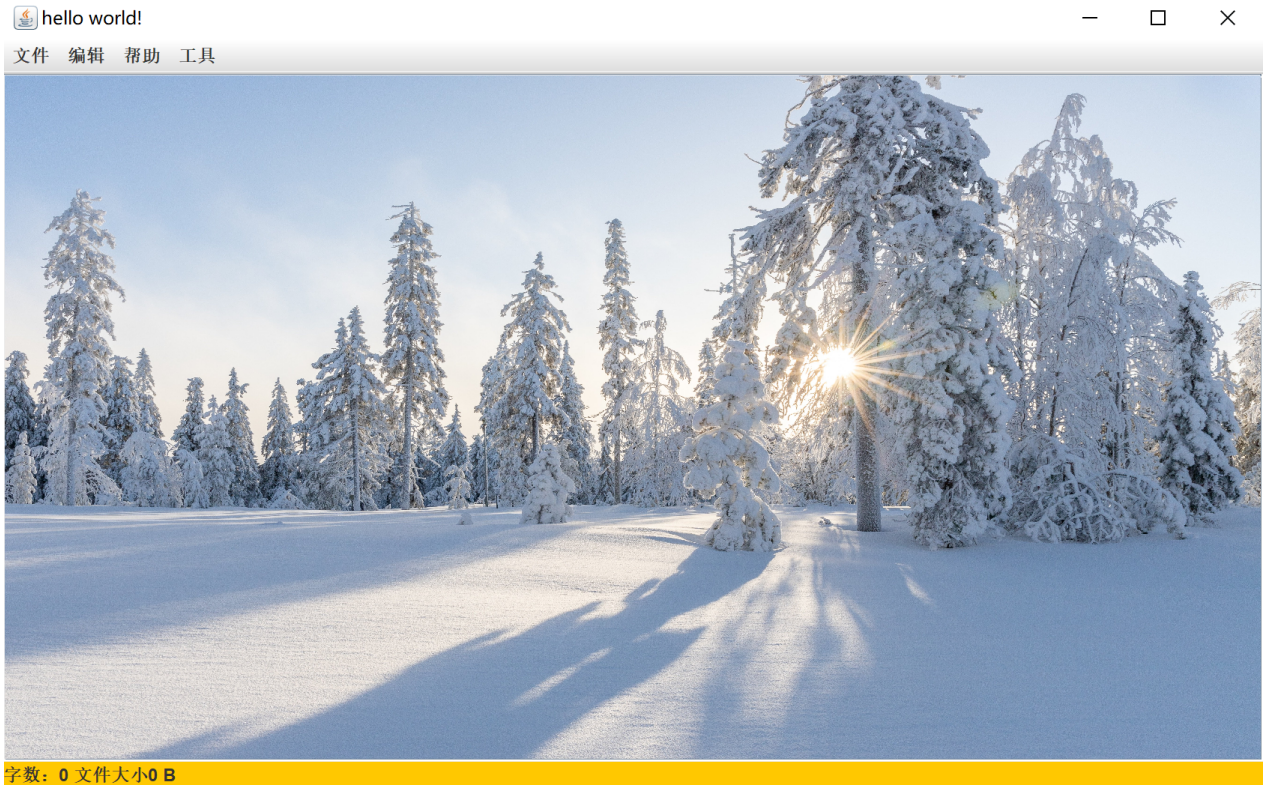
    > 撤销，复制，剪切，粘贴，查找和替换等
    >
    > 这里，对于简单的直接在响应方法中完成即可，无须令写类来封装。而对于查找，替换，插入图片等操作使用特定的一个类来封装。

  - 帮助

    > 较为简单，弹出一个frame即可。

  - 工具

    > 包含工具栏，点击可以显示出来（如下图）其中可以进行基本的排版设置（这里都可以直接在`listener`响应函数中简单完成）
    >
    > 注：这部分目前代码只写到这里，后期会进行完善。
    >
    > 
    >



对于笔记本 `设置` 功能的实现

使用XML文件保存默认设置，应用一开始会从此文件中读取所有的默认设置，并据此显示笔记。在修改设置的内容是会改变XML中记录的相关设置。

对于 `打开` 和 `保存` 功能的实现

简单的可以只保存一个笔记内容到一个文件中（格式不强求）。

而复杂的需要除了笔记文本内容外保存其他信息，如时间信息，此笔记的排版信息等。这个我准备使用json文件来保存，打开的时候读取相关的值即可。所以这里写一个 `页` 的类，用来保存所有我们需要保存的笔记的信息，这样每次 `保存` 和 `读取` 的时候对这个页进行相应的操作即可。

`快捷键` 功能的实现

实现并不复杂，专门写一个类来负责快捷键，除了默认的快捷键外用户也可以自行添加。

快捷键使用键盘监视器就可以做到，可以设置几个比较方便的快捷键。

这里想着可以做一套像Emacs的快捷键（还没实现）

`历史记录`功能的实现

每次在保存的时候将`页`的几个历史记录需要的信息保存的历史记录的文件中（放在一个单独的json文件中），不可缺少的信息有：文件序号和文件所在地址。

打开历史记录是，会调用类`history`的方法，读取文件并显示所有的记录供用户选择。

`模式切换`的实现（类似vim，可以设置3个模式，可实现，目前只是写了一部分）

1. 设置文本域不可编辑
2. 使用键盘响应为每个键设置一个新的动作，这里展示里最为简单和最为著名的vim的四个光标移动的实现，实现很成功。

```
        jta.addKeyListener(new KeyListener() {
            @Override
            public void keyPressed(KeyEvent e) {
                try {
                    rbt = new Robot();
                } catch (AWTException ex) {
                    ex.printStackTrace();
                }
                int offset = jta.getCaretPosition();
                int totalLineCount = jta.getLineCount();
                int row = 0;
                try {
                    row = jta.getLineOfOffset(offset);
                } catch (BadLocationException ex) {
                    ex.printStackTrace();
                }
                System.out.println(row);
                System.out.println(offset);

                if(e.getKeyCode() == KeyEvent.VK_0){
                    System.out.println("0");
                    jta.setEditable(false);
                    isM = true;
                }
                else if(isM && e.getKeyCode() == KeyEvent.VK_H){
                    jta.setCaretPosition(offset-1);
                }
                else if(isM && e.getKeyCode() == KeyEvent.VK_L){
                    jta.setCaretPosition(offset+1);
                }
                else if(isM && e.getKeyCode() == KeyEvent.VK_J){
                    rbt.keyPress(KeyEvent.VK_DOWN);
                }
                else if(isM && e.getKeyCode() == KeyEvent.VK_K){
                    rbt.keyPress(KeyEvent.VK_UP);
                }
```

```
            }
    }
```

备注：

- 时间有限，还没有太深入，争取接下来的时间进行完善，增加更多的功能。
- 这次作业中没有提到过多的关于面向对象的内容，主要是时间多给了写代码中，且在第一次作业中分析足够，这次就没有太多补充的，准备最后一起补充。