**设计实现：随手记事本**

本设计实现基于面向对象的程序设计课程教授的内容，分功能分析与建模、核心流程设

计分析、高级设计意图三个部分，在本学期通过三个阶段的实践，利用 Java 语言进行面向

对象的程序设计，以练习和运用学习的面向对象的程序设计思想，加深对相关软件工程思想

的理解，并实现一个通过面向对象编程的程序。

## 功能分析与建模

1、软件功能描述

本项目计划实现一个有文本编辑、图片插入、表格、云端存储功能的记事本。

1. 需求分析：

1)支持文件的打开与保存，以及基本的文字操作

[用例名称]

文本文件的打开、编辑与保存

[场景]

who : 本地文件、TextArea、用户

where : 笔记软件的主界面

when : 用户打开文件时

[用例描述]

1.用户打开一个本地文件

2.文件内容被TextArea呈现出来

3.用户开始编辑TextArea

4.点击按钮，进行保存。

[用例价值]

    为用户提供记事本的基础文字功能

[约束和限制]

1. 用此种方式只能打开纯文字的内容。
2. 支持表格功能

[用例名称]

文本文件的打开、编辑与保存

[场景]

who : 本地文件、TextArea、用户

where : 表格栏

when : 用户打开文件时

[用例描述]

1.用户打开一个本地文件

2.一个表格栏被提供给用户

3.用户编辑表格内容

4.点击保存按钮，表格的内容会随着文本内容被保存

[用例价值]

    为用户提供表格功能

[约束和限制]

1.表格并不能自由的修改。

2.表格的内容被保存后，再次打开时，不能直接显示在表格内

1. 支持图片功能

[用例名称]

图片的插入，保存与再次打开

[场景]

who : 本地、TextArea、用户

where : Textarea

when : 用户使用图片相关功能

[用例描述]

1.用户打开一个本地文件

2.用户点击插入一个图片

3.用户编辑表格内容

4.点击保存按钮，表格的内容会随着文本内容被保存

[用例价值]

    使TextArea能够处理图片

[约束和限制]

1.图片只能上传一个。

1. 支持上传功能

[用例名称]

本地文件上传到云端

[场景]

who : 本地、TextArea、用户、云端

where : 工具栏

when : 用户使用上传功能

[用例描述]

1.用户打开文本

2.用户编辑文本内容

3.用户点击上传键

4.文件的内容被上传到了云端中。

[用例价值]

    使本地的文件能够上传到云端。

[约束和限制]

1. 只能上传到特定的云。

3.抽象出类

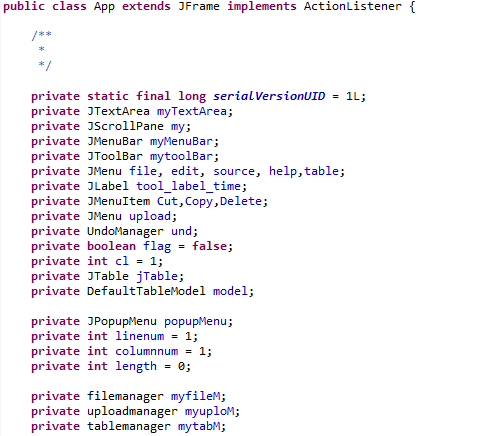
|  |  |  |
| --- | --- | --- |
| 类 | 属性 | 方法 |
| 主界面（文本框） | 大小，字数等 | 文字编辑与保存 |
| 表格 | 表格的行数与列数 | 表格文字的修改，行列数的增减 |
| 图片 | 图片位置等 | 图片的插入与保存 |
|  |  |  |

## 核心流程设计分析

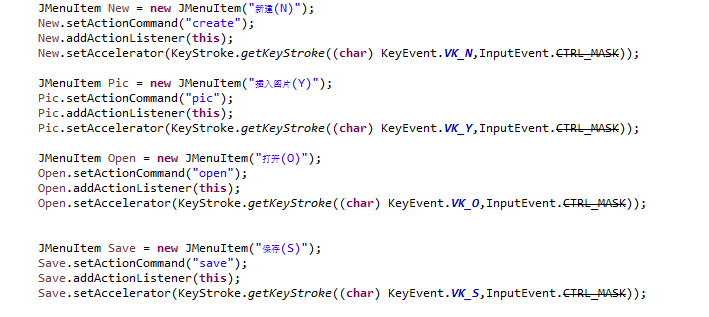
这一节中，我们将把所有类组合起来，共同实现软件功能。

1. 程序的运行

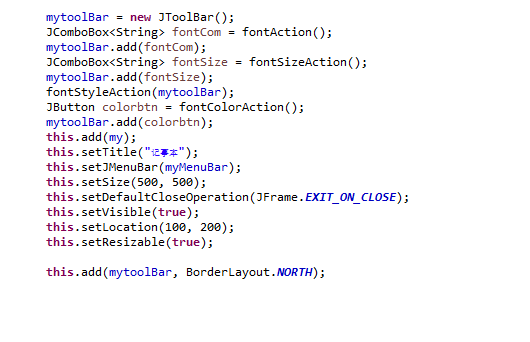
打开界面后，初始化各种类：



同时创建各种类的实例：

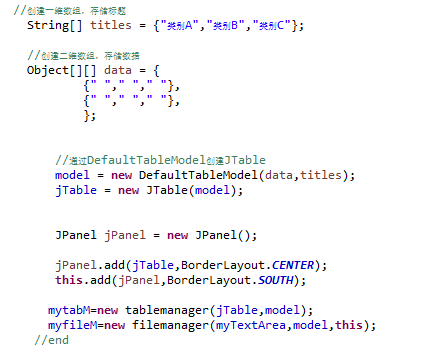


设置有关主界面的参数：

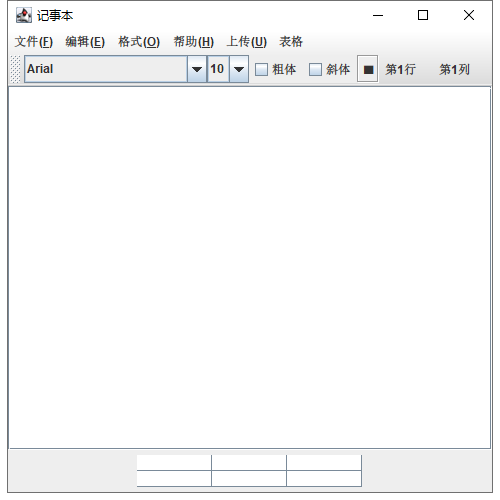


同时，为各种按键设置行为监听器，在相应的行为触发时执行相应的动作：

表格的实现：



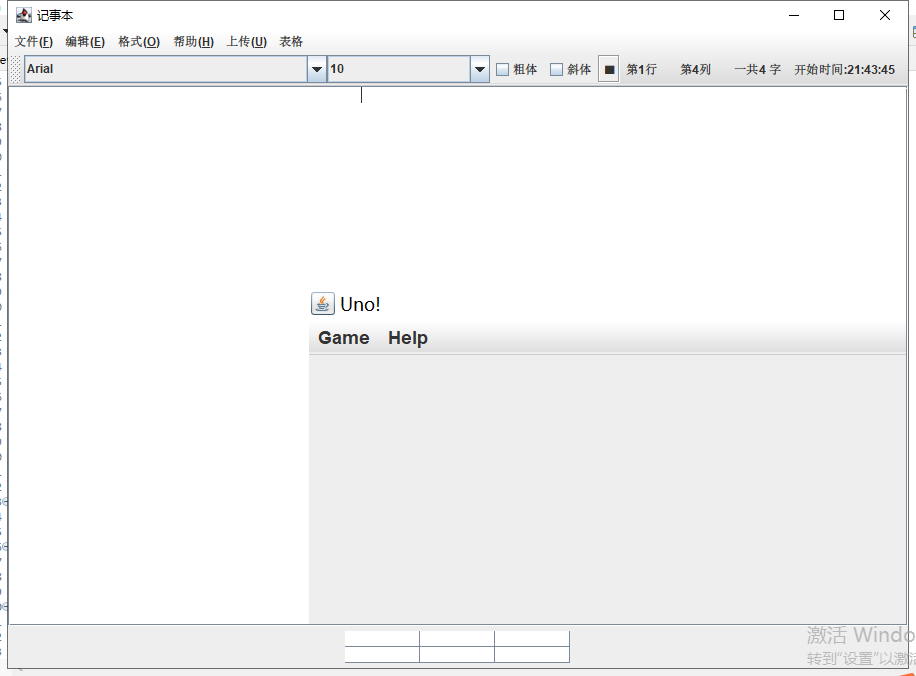
运行界面：



菜单：



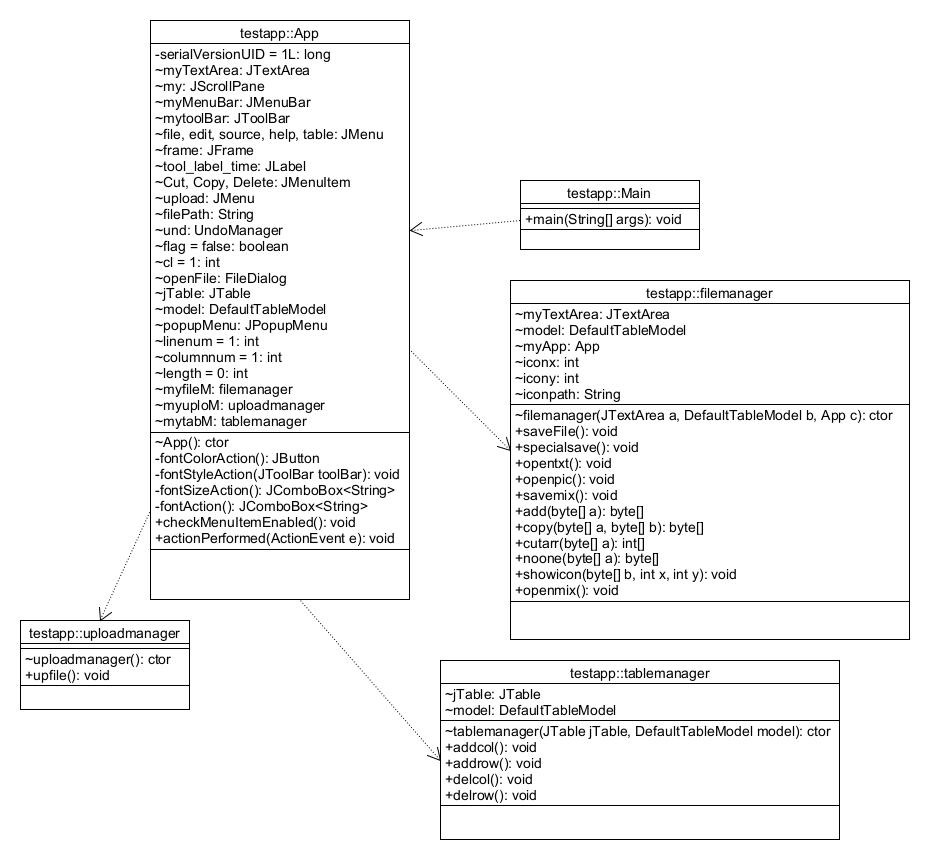
打开图片示例：



（全屏模式可以看到完整的图片）

1. 类图的分析

由Eclipse中的工具，得到如下类图：



可以看到最关键的类是App类，组合了TextArea、JToolBar、JMenuBar等等类以实现一个界面。此外，这个类依赖filemanager和Uploadmanager以及tablemanager这三个类来进行读取文件、保存文件和上传文件。使用了actionPerform方法去监听发生的动作。

## 高级设计意图

在流程的设计中，尽量符合了组合原则、单一职责原则和接口隔离原则。

**I.设计原则**

1.组合原则

​ 组合原则（又名合成复用原则）的特点是将已有的对象纳入新的对象，从核心流程设计中可以知道最关键的类是App类，它组合了 filemanager，Uploadmanager以及tablemanager这三个类来进行工作，它使用的方法有：

filemanager的 openText 和 openPic等 方法、

Uploadmanager 类的upfile方法

tablemanager 类的 addcol,addrow等方法

2.单一职责原则

​ 单一职责原则提出对象不应该承担太多职责，核心在于控制类的粒度大小，降低类的复杂度，提高系统的可维护性。在 设计中，这个设计原则有所体现：

应用场景的载入由 App 类实现

应用对文件的读写由 filemanager 类实现

表格的功能由 tablemanager 类实现

用户上传笔记本的操作由另一个类 Uploadmanager 实现

​ 可以看到设计中不同类别的功能由相应的类去实现，这样对于应用的可维护性和扩展性都有好处。

iii. 接口隔离原则

​ 接口隔离原则要求设计者将较为庞大的接口拆分成更具体的接口，它和单一职责原则一样都是为了提高类的内聚性、降低它们之间的耦合性，不同之处在于：单一职责原则注重的是职责，主要是约束类的行为，针对程序中的实现和细节；而接口隔离原则注重对接口依赖的隔离，主要是约束类的接口，针对抽象和程序整体的构建。这个设计原则有所体现：

应用对文件的读写过程由filemanager提供openText 和 openPic等 方法

用户上传笔记本的操作过程由 Uploadmanager 类提供upfile 方法

​ 可以看到，接口隔离原则通过定义合理的接口粒度大小，减少了代码冗余，提高了设计的灵活性和可拓展性。

**II.设计模式**

1.观察者模式

​ 观察者模式指多个对象间存在一对多的依赖关系，当一个对象的状态发生改变时，所有依赖它的对象都将得到通知并被自动更新，它的特点在于，降低了目标与观察者之间的耦合，在目标与观察者之间建立了一套触发机制。设计中，观察者模式体现为：

用户对 TextArea 进行编辑，与此同时 TextArea 的观察者读取到新的内容，进行文本的解析，将结果传递到撤销/剪切/复制使能按钮中

用户点击上传按钮，点击行为被观察到，文件被上传

​ 观察者可以是任意数量，一个按钮被点击了，视图并不会知道接下来会发生什么，但相应的过程会在观察者模式的机制下发生。

2. 组合模式

​ 组合模式使用户对单个对象和组合对象具有一致的访问性，它本身就是设计中对于组合原则的应用，正如之前对组合原则的应用进行介绍时所描述的，设计中包含了对组合模式和组合原则的体现，除了之前提到的之外，App类对于各种工具栏、菜单栏、按钮的组合也可算在内。

​ 组合模式的优点在于其使得应用更容易在组合体内加入新的对象，客户端不会因为加入了新的对象而需要修改代码。

## 总结

从对Java几乎一窍不通，到磕磕绊绊地搜查资料、一步一步地学习各种设计模式和面向对象思想，我最终在期末完成了这一个大作业。虽然还有诸多问题可以改进，但我还是很有成就感。非常感谢老师一个学期来的悉心教导和鼓励，这一门课让我受益匪浅。