**江苏科技大学**

**课程设计文档**

课 程： 应用密码学课程设计

课 题： Vigenere密码通

学 院： 计算机学院

组 员： 182210710119 陈四贵

182210710102 曹 萍

班 级： 1822107101

开发语言： Java

目 录

[一、 课程设计项目 1](#_Toc42181114)

[二、 小组成员及分工 1](#_Toc42181115)

[三、 实现过程 1](#_Toc42181116)

[四、 算法说明 3](#_Toc42181117)

[1.实现思路 4](#_Toc42181118)

[2.算法简述 4](#_Toc42181119)

[3.实现源代码 5](#_Toc42181120)

[①加密模块 5](#_Toc42181121)

[②解密模块 6](#_Toc42181122)

[五、 功能测试 6](#_Toc42181123)

[1.输入合法性测试 6](#_Toc42181124)

[2.加密算法测试 7](#_Toc42181125)

[3.解密算法测试 8](#_Toc42181126)

[六、 总结 8](#_Toc42181127)

# 课程设计项目

Vigenere密码通，一款致力于帮助初学者学习Vigenere密码的软件，包含Vigenere密码历史概述、Vigenere密码描述、解密方法、加解密Vigenere密码等功能。

# 小组成员及分工

182210710119 陈四贵：

1.Vigenere密码加、解密算法编程；

2.后端逻辑编程；

3.前端界面优化；

4.设计文档撰写。

182210710102 曹 萍：

1.搜集Vigenere密码相关信息；

2.前端界面设计及开发；

3.制作运行视频。

共同任务：软件定位、约定规范。

# 实现过程

本课程设计作品使用了java fx框架与著名的mvc设计模式，前端界面与后端逻辑分离，分层明显、结构清晰，便于开发。

考虑到作品实际，不需要model(m)文件，因而将整个作品设计文件分成四类，分别存到四个文件夹里，分类处理，分别是：view(存放视图文件，也即所谓的前端界面，这里是fxml文件)、controller(存放每个视图对应的控制器，也即所谓的后端逻辑，这里是java文件)、launch(存放整个项目的启动类，也即主类)、icon（存放整个作品中使用的图片等）。

view里存放的视图文件使用类似html的fxml标签编写，按类型主要有七个界面：开始界面、帮助界面、Vigenere密码历史概述界面、Vigenere密码描述界面、Vigenere密码解密方法界面、Vigenere密码加解密界面界面、深入学习密码界面。

整个项目设有一个启动类，主要作用是启动程序、加载第一个页面。存放于launch文件夹中。

每个视图文件都有其对应的控制器类(controller)，每个控制器类里都有对应的事件处理方法，共同的是“3个文件菜单项和各界面跳转菜单项”，所实现的功能基本上是加载新的页面（fxml文件）。

最核心的加密、解密算法置于Pane\_EnandDe.java中。

通用事件处理对应关系如表1所示，整个项目结构示意图如表2所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 菜单项 | 触发事件 | 控制器中的处理事件函数 | 备注 |
| 上一步 | 点击(onAction) | clickPrevious() | 返回上一步骤界面，各界面不一 |
| 下一步 | clickNext() | 加载下一步骤界面，各界面不一 |
| 退出 | clickExit() | 退出程序 |
| 主界面 | click\_main() | 加载对应界面 |
| Vigenere密码历史 | click\_History() |
| Vigenere密码描述 | click\_Dscribe() |
| Vigenere密码破译方法 | click\_CrackingMethod() |
| Vigenere密码加解密 | click\_EnandDe() |
| 使用说明 | click\_Instruction() |
| 密码深入学习 | click\_FurtherStudy() |

表1·通用事件处理对应关系

|  |  |  |
| --- | --- | --- |
| 文件夹及其类型 | 文件 | 描述/功能 |
| 视图文件夹view | Pane\_main.fxml | 开始界面 |
| Pane\_History.fxml | Vigenere密码历史界面 |
| Pane\_Describe.fxml | Vigenere密码描述界面 |
| Pane\_CrackingMethod.fxml | Vigenere密码破解方法描述界面 |
| Pane\_EnandDe.fxml | Vigenere密码加、解密界面（核心） |
| Pane\_Instruction.fxml | 使用指南/帮助界面 |
| Pane\_FurtherStudy.fxml | 密码深入学习界面 |
| 启动文件夹Launch | MainApp.java | 启动类，启动程序 |
| 控制器文件夹controller | Pane\_main\_Con.java | 开始界面控制器 |
| Pane\_History\_Con.java | Vigenere密码历史描述界面控制器 |
| Pane\_Describe\_Con.java | Vigenere密码描述界面控制器 |
| Pane\_CrackingMethod\_Con.java | Vigenere密码破解方法描述界面控制器 |
| Pane\_EnandDe\_Con.java | Vigenere密码加解密界面控制器 |
| Pane\_Instruction\_Con.java | 使用指南/帮助界面控制器 |
| Pane\_FurtherStudy\_Con.java | 密码深入学习界面控制器 |
| 资源文件夹icon | Confident\_Smile.png | 资源文件 |
| Country\_Icon.png |
| Emblem\_Icon.png |
| HEY.png |
| School\_Icon.jpg |
| Show\_1.png |
| Show\_2.png |
| Vigenere.png |

表2·项目结构示意表

# 算法说明

## 1.实现思路

本作品的核心功能在Pane\_EnandDe.fxml与Pane\_EnandDe\_Con.java里。Pane\_EnandDe里设置了文本域组件、文本区组件、按钮组件等。

其中，几个文本域组件用于接收用户键入的明文信息、密钥信息或密文信息，按钮组件用于控制加密与解密操作，文本区组件用于显示加解密过程信息。此外还附有合法性检查，利用正则表达式检查明文、密钥、密文是否是合法输入（全英文才是合法输入），如果有非法输入，则在状态打印栏打印错误信息。

一旦点击加密按钮，加密算法便开始工作，文本区组件同步打印加密信息；一旦点击解密按钮，解密算法便开始工作，文本区组件同步打印解密信息；一旦点击清空按钮，便清空明文、密钥、密文文本域信息。

## 2.算法简述

Vigenere算法可以看做若干个凯撒密码的组合——每个明文都拥有自己对应的密钥字符进行代替操作（密钥可重复）。

假设a是密钥，b是明文，c是密文，对于普通的凯撒密码来说，

凯撒密码如此，Vigenere密码也是如此，区别只是每一位明文所对应的密钥不一定相同。因此，我们分别使用几个字符串来记录密钥、明文或密文。对密钥的位数采取取模操作解决明文或密文长度比密钥长的问题，这也可以理解为所谓的分组。简要地说，如果密钥的长度是num，则每次取密钥时，取出的是第i%num位的密钥。

考虑到字母的大小写可能也是明文、密文的重要组成部分，改变大小写有可能导致密码破译上增加困难（本来大写的字母被转为小写混淆在一起，可能难以区别哪些字符和哪些字符是一个部分），我们并没有一刀切地将明文、密文统一转为大写或小写。又考虑到密钥输入的大小写对Vigenere密码加解密并没有什么影响（不管大写还是小写，都是改变同样的位置），为统一处理，将密钥统一转为小写字母，而根据明文或密文的大小写使用不同的处理方法。

考虑到计算机中字母对应ASCII码的实际，对上述公式作出了些许调整。在加解密过程中，我们并没有把明文或密文转为对应的0-25数字，而是只把密钥减去’a’对应的ASCII码值97，因为在加完后，数字还是要加上’a’或’A’对应的ASCII码值。换句话说，我们采用的是较直接的方法，直接看需要移动几位，明文或密文直接移动，而非明文、密文、密钥都完全转为对应数字后再转为对应字符。令a是密钥对应的ASCII码，b是明文对应的ASCII码，c是密文对应的ASCII码，调整公式如下所示：

但是这种方法可能产生越界——加密时得到的c可能大于52个英文字母（包括大小写）对应的ASCII码，解密时得到的b可能小于52个英文字母（包括大小写）对应的ASCII码。我们采取分类讨论的处理方法——加密时，若字母是大写，得到的值大于90（’Z’），就有问题，需要减去26，若字母是小写，得到的值大于122（’z’），也需要减去26；解密时，若字母是大写，得到的值小于65（’A’），就有问题，需要加上26，若字母是小写，得到的值小于97（’a’），也需要加上26.以此完善越界查漏机制。

算法思想叙述完毕。

## 3.实现源代码

### ①加密模块

1. **public** String Vigenere\_Add() {//加密算法
2. **int**  num=a.length();//num是密钥的长度
3. **for** (**int** i = 0; i < b.length() ; i++) {//b.length是明文的长度
4. **int** Temp=a.charAt(i%num)-97;//Temp现在是密钥的对应值
5. Temp=b.charAt(i)+Temp%26;
6. **if**(65<=b.charAt(i)&&b.charAt(i)<=90) {
7. **if** (Temp > 90) Temp -= 26;
8. }
9. **else**{
10. **if**(Temp>122) Temp-=26;
11. }
12. Explain.setText(Explain.getText()+(i+1)+"：密钥第"+((a.charAt(i%num)))+"行，明文第"+(b.charAt(i))+"列，得出密文为"+(**char**)Temp+"\n");
13. c = c.concat(Character.toString((**char**)Temp));
14. }
15. **return** c;
16. }

### ②解密模块

1. **public** String Vigenere\_Decrease(){//解密算法
2. **int** num=a.length();
3. **for**(**int** i=0;i<c.length();i++){
4. **int** Temp=a.charAt(i%num)-97;//Temp现在是密钥的对应值
5. Temp=c.charAt(i)-Temp%26;
6. **if**(65<=c.charAt(i)&&c.charAt(i)<=90) {
7. **if** (Temp <65) Temp += 26;
8. }
9. **else**{
10. **if**(Temp<97) Temp+=26;
11. }
12. Explain.setText(Explain.getText() + (i + 1) + "：密文第" + ((c.charAt(i))) + "行，密钥第" + (a.charAt(i%num)) + "列，得出明文为" + (**char**) Temp + "\n");
13. b = b.concat(Character.toString((**char**)Temp));
14. }
15. **return** b;
16. }

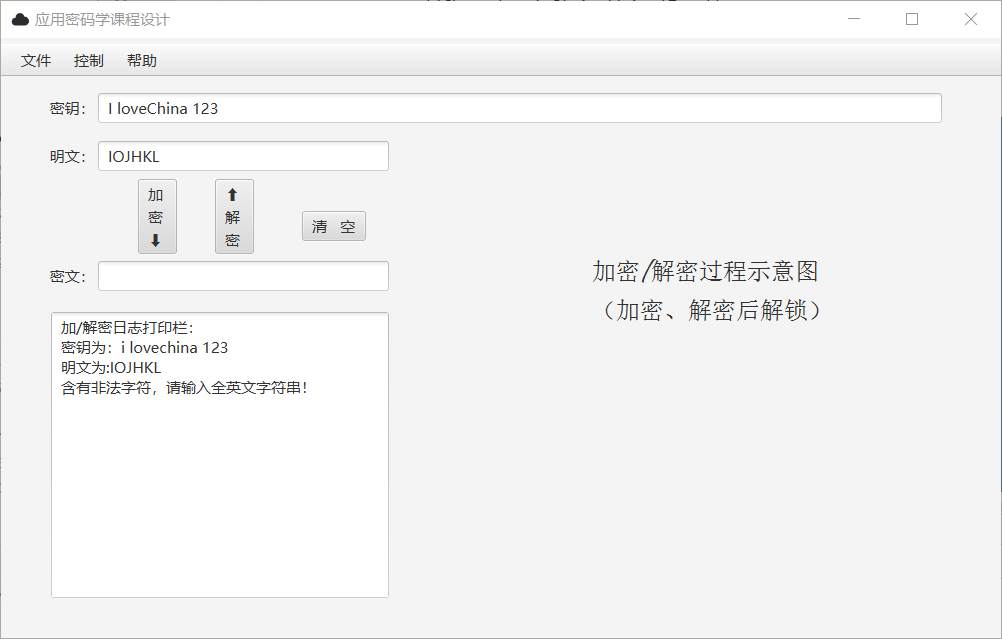
# 功能测试

## 1.输入合法性测试

输入明文：I loveChina 123

输入密钥：IOJHKL

测试结果：

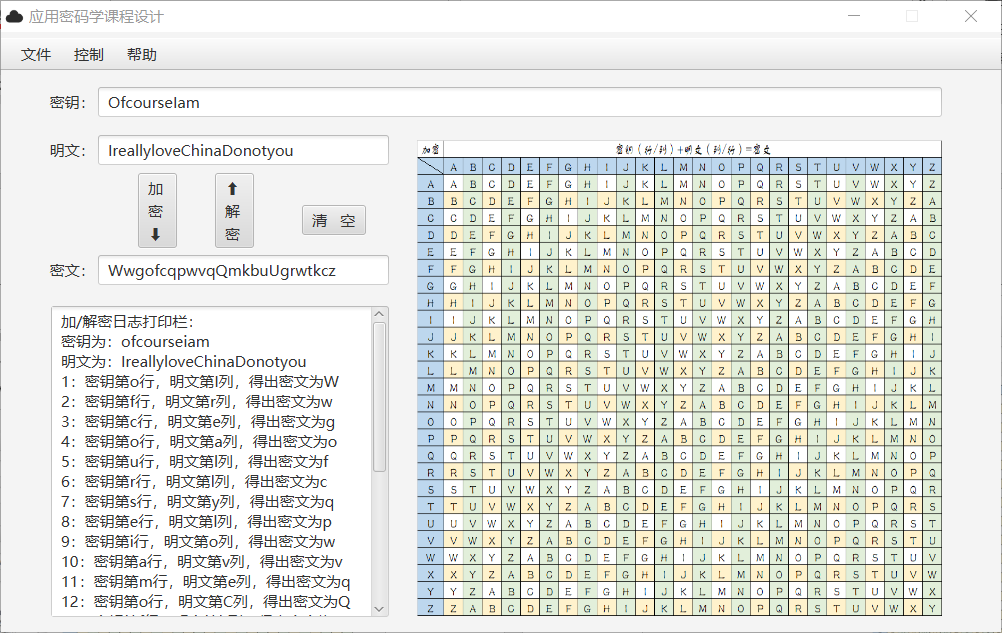


## 2.加密算法测试

输入明文：IreallyloveChinaDonotyou

输入密钥：OfcourseIam

测试结果（密钥）：WwgofcqpwvqQmkbuUgrwtkcz

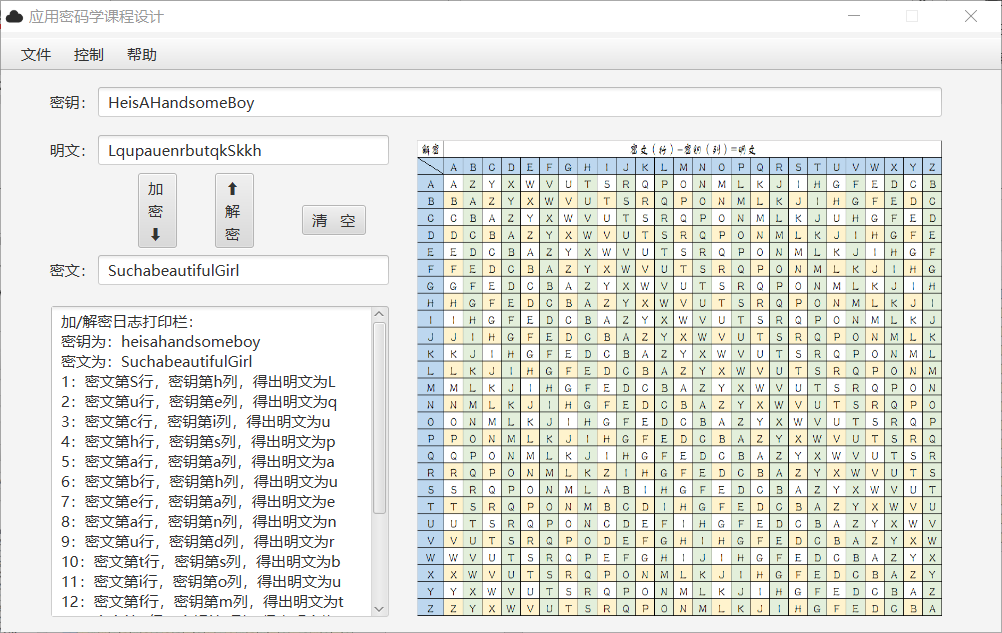


## 3.解密算法测试

输入密文：SuchabeautifulGirl

输入密钥：HeisAHandsomeBoy

测试结果（明文）：LqupauenrbutqkSkkh



# 总结

1.在编写加解密算法时，应结合计算机和数学实际考虑，只是机械地根据ASCII码值“找规律”不是很可行，只是一昧地生搬硬套数学公式也不行，应综合考量，这样才能把加解密算法做得既简便又高效。

2. Vigenere算法是一种多表代替密码，能制造混乱，使得确定消息和密钥是如何转换成密文的尝试变得困难。

3.虽然Vigenere密码算法简单，但是它“容易破译”主要体现在知道密钥的时候，如果不知道密钥，由于是多表代替密码，改变了每个字母出现的频率等基本信息，制造了混乱，破译难度很大。这种思想值得我们借鉴，多种简便算法组合也可组成破译难度高的密码。