**软件课程设计II**

**学院（系）： \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**专 业： \*\*\*\*\*\*\*\***

**学 生： \*\*\*\*\*\***

**学 号： \*\*\*\*\*\*\*\*\*\*\*\*\***

**完成日期： \*\*\*\*\*\*\*\*\*\*\*\***

**目 录**

[第一章 编译器系统需求分析 1](#_Toc71380399)

[1.导言 1](#_Toc71380400)

[2.系统定义 1](#_Toc71380401)

[3.应用环境 2](#_Toc71380402)

[4.功能规格 3](#_Toc71380403)

[4.3.1词法分析 4](#_Toc71380404)

[4.3.2语法分析 5](#_Toc71380405)

[4.3.3语义分析 5](#_Toc71380406)

[5.性能需求 5](#_Toc71380407)

[6.产品提交 6](#_Toc71380408)

[第二章 编译器系统概要设计 7](#_Toc71380409)

[1.系统分析 7](#_Toc71380410)

[2. 界面设计 7](#_Toc71380411)

[第三章 编译器系统详细设计 10](#_Toc71380412)

[1. 详细设计概述 10](#_Toc71380413)

[2. 辅助工具类 10](#_Toc71380414)

[3. 词法分析 11](#_Toc71380415)

[4. 语法分析与语义分析 12](#_Toc71380416)

[5. 用户界面（前端） 13](#_Toc71380417)

[第四章 编译器系统实现效果 15](#_Toc71380418)

[1.界面展示 15](#_Toc71380419)

[2.具体效果 17](#_Toc71380420)

[第五章 编译器系统总结体会 18](#_Toc71380421)

[1.总结和展望 18](#_Toc71380422)

[2.感想 18](#_Toc71380423)

# 第一章 编译器系统需求分析

## 1.导言

### 1.1 目的

该文档是关于用户对于编译器系统的功能和性能的要求，重点描述了编译器系统的功能需求，是概要设计阶段的重要输入。

本文档的预期读者是：

· 设计人员；

· 开发人员；

· 项目管理人员；

· 测试人员；

· 用户。

### 1.2 范围

该文档是借助于当前系统的逻辑模型导出目标系统的逻辑模型的，解决整个项目系统的“做什么”的问题。在这里，没有涉及开发技术，而主要是通过建立模型的方式来描述用户的需求，为客户、用户、开发方等不同参与方提供一个交流的平台。

### 1.3 编写说明

UML,Unified Modeling Language（统一建模语言）的缩写，是一个标准的建模语言。

## 2.系统定义

我们分别阐述一下项目的来源、背景，项目的用户特点和项目的目标。

### 2.1 项目来源及背景

本次课程设计要求学生掌握词法分析器、语法分析器、中间代码生成器的原理和实现技术。通过本实验使学生能够结合所学的数据结构、操作系统、软件设计等知识，编写一个词法分析器和一个语法分析器。

本软件课程设计综合了编程语言、程序设计基础、数据结构、编译原理等多门课程，通过实验教学使学生能充分了解编译程序的功能，设计有效的软件解决方案，实施软件开发。为学生提供了一个既动手又动脑，独立实践的机会，将课本上的程序设计基础、数据结构、编译原理理论知识和实际有机的结合起来，使学生在掌握计算机复杂软件设计理论和方法方面的基本知识的同时，也具有设计、实现、分析和维护编译器等复杂程序等方面的初步能力，提高学生适应实际，实践编程的能力。

### 2.2 项目目标

本项目设定的目标如下：

· 系统能够提供友好的用户界面，使操作人员的工作量最大限度的减少；

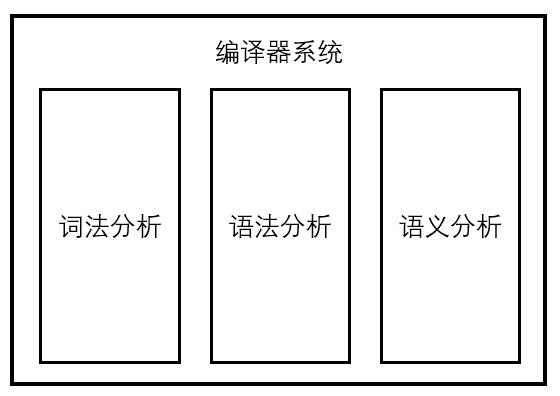
· 系统运行的效率一般，但是运算出结果所需的时间应该是用户在可以接受的范围内 （<=0.1秒）；

· 通过这个项目可以锻炼自己，提高开发能力和项目管理能力。

## 3.应用环境

根据课程要求，可以确定本项目的功能为词法分析、语法分析与语义分析。

它们的关系如图1-1所示。



**图1-1 编译器系统组成图**

### 3.1 系统运行的硬件环境

本系统的硬件环境如下：

· CPU：2.6 GHz以上

· 内存：4 GB以上

· 分辨率：推荐使用1024×768以上像素

### 3.2 系统运行软件环境

本系统的软件环境如下：

· 操作系统：Win 10或以上版本

· 开发工具包：JDK/IDEA

## 4.功能规格

我们采用面向对象分析作为主要的系统建模方法，使用UML（Unified Modeling Language）作为建模语言。UML为建模活动提供了从不同角度观察和展示系统的各种特征的方法。在UML中，从任何一个角度对系统所作的抽象都可能需要几种模型来描述，而这些来自不同角度的模型图最终组成了系统的映像。

用例描述角色（用户、外部系统以及系统处理）是如何与系统交互来完成工作的。用例模型提供了一个非常重要的方式来界定系统边界以及定义系统功能，同时，该模型将来可以派生出动态对象模型。

设计用例时，我们遵循下列步骤：

1）识别出系统的角色。角色可以是用户、外部系统，甚至是外部处理，通过某种途径与系统交互。重要的是着重从系统外部执行者的角度来描述系统需要提供哪些功能，并指明这些功能的执行者（角色）是谁。尽可能地确保所有角色都被完全识别出来。

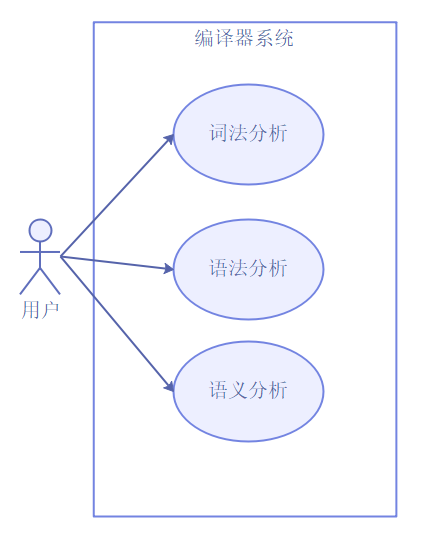
2）描述主要的用例。可以采取不断地问自已“这个角色究竟想过系统做什么？”来准确地描述用例。

3）重新审视每个用例，为它们下个详尽的定义。

### 4.1 用户定义

用户指的是使用本编译器系统进行词法分析、语法分析与语义分析的学生及老师。通过本软件，用户可以便捷地了解到词法分析、语法分析与语义分析重点步骤的结果，加深对编译原理核心知识的掌握。

### 4.2 系统主用例图



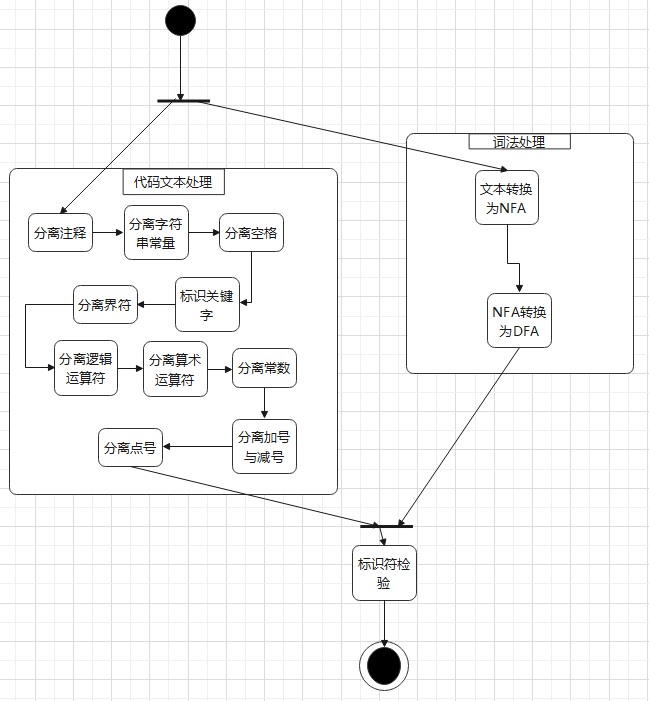
**图1-2 系统的主用例图**

### 4.3 用户端系统

用户端通过系统进行词法、语法与语义分析。

4.3.1词法分析

|  |  |
| --- | --- |
| 用例名称 | 词法分析 |
| 实现名称 | Lexer |
| 用例描述 | 用户通过此用例获得词法分析的相关信息 |
| 参与者 | 用户 |
| 前置条件 | 进入编译器平台 |
| 后置条件 | 无 |
| 主事件流 | 1.用户选择代码与词法文件  2.用户单击词法分析或NFA&DFA按钮 |
| 备选事件流 | 1.用户没有选择代码文件  1.弹窗显示用户没有输入代码  2. 用户没有选择词法文件  1.弹窗显示用户没有输入词法 |
| 业务规则 | 词法分析 |



4.3.2语法分析

|  |  |
| --- | --- |
| 用例名称 | 语法分析 |
| 实现名称 | Parse |
| 用例描述 | 用户通过此用例获得语法分析的相关信息 |
| 参与者 | 用户 |
| 前置条件 | 进入编译器平台，已完成词法分析 |
| 后置条件 | 无 |
| 主事件流 | 1.用户选择代码与语法文件  2.用户单击语法分析或展示Goto按钮 |
| 备选事件流 | 1.用户没有选择代码文件  1.弹窗显示用户没有输入代码  2. 用户没有选择语法文件  1.弹窗显示用户没有输入语法 |
| 业务规则 | 语法分析 |

4.3.3语义分析

|  |  |
| --- | --- |
| 用例名称 | 语义分析 |
| 实现名称 | Parse |
| 用例描述 | 用户通过此用例获得语义分析的相关信息 |
| 参与者 | 用户 |
| 前置条件 | 进入编译器平台，已完成词法分析 |
| 后置条件 | 无 |
| 主事件流 | 1.用户选择代码与语义文件  2.用户单击语义分析或展示Goto按钮 |
| 备选事件流 | 1.用户没有选择代码文件  1.弹窗显示用户没有输入代码  2. 用户没有选择语义文件  1.弹窗显示用户没有输入语义 |
| 业务规则 | 语义分析 |

## 5.性能需求

由于时间限制，本系统并未解决并发性问题。

### 5.1 界面需求

系统的界面要求如下。

1）页面内容：主题突出，站点定义、术语和行文格式统一、规范、明确、栏目、菜单设置和布局合理，传递的信息准确、及时。内容丰富，文字准确，语句通顺，专用术语规范，行文格式统一规范。

2）导航结构：页面具有明确的导航指示，且便于理解，方便用户使用。

3）技术环境：页面大小适当，能用各种常用浏览器以不同分辨率浏览，无错误链接和空链接；采用Swing处理前端，控制字体大小和版面布局。

4）艺术风格：界面、版面形象清晰悦目、布局合理，字号大小适宜、字体选择合理，前后一致，美观大方，动与静搭配恰当，动静效果好；色彩和谐自然，与主题内容相协调。

### 5.2 开放性需求

系统应具有较强的灵活性，以适应将来功能扩展的需求。

### 5.3 可扩展性需求

还有很多功能可以添加，如：

1代码优化

2展示语法树

以上功能均可实现，时间有限，尚未完成。

## 6.产品提交

提交产品为：

a）课程设计报告（电子版）。

b）源代码。

c）可执行文件。

d）代码说明文档。

e）使用实例。

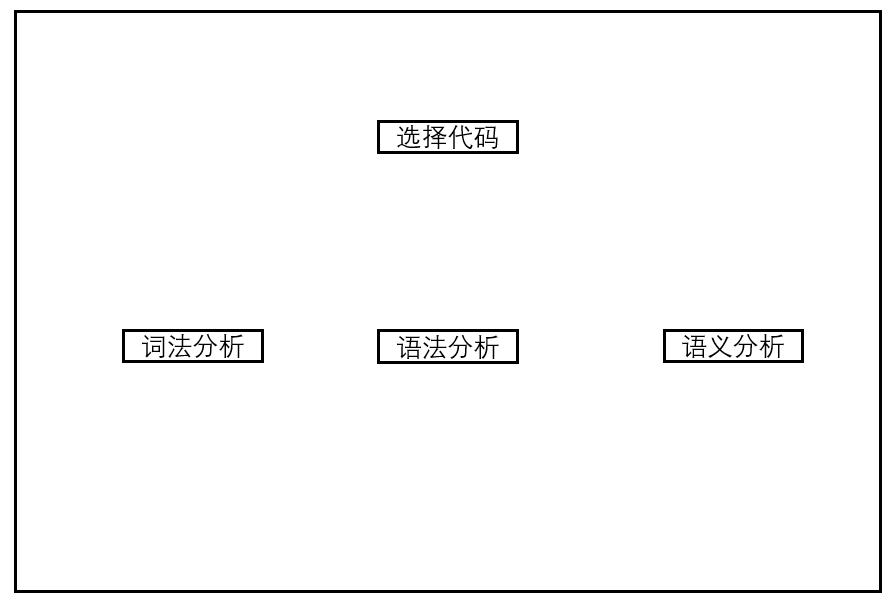
# 第二章 编译器系统概要设计

## 1.系统分析

本系统主要为用户提供词法、语法与语义分析的功能。用户可以自定义需要判断的代码、词法规则、语法规则与语义规则，并使用本系统进行判断，本系统也为用户提供了一些中间过程的展示界面，便于用户对编译原理的核心知识产生更深入的了解。

## 2. 界面设计

### 2.1 主界面设计



**图2-1 主界面**

主界面提供词法分析、语法分析与语义分析的导航功能。

主要界面设计如下：

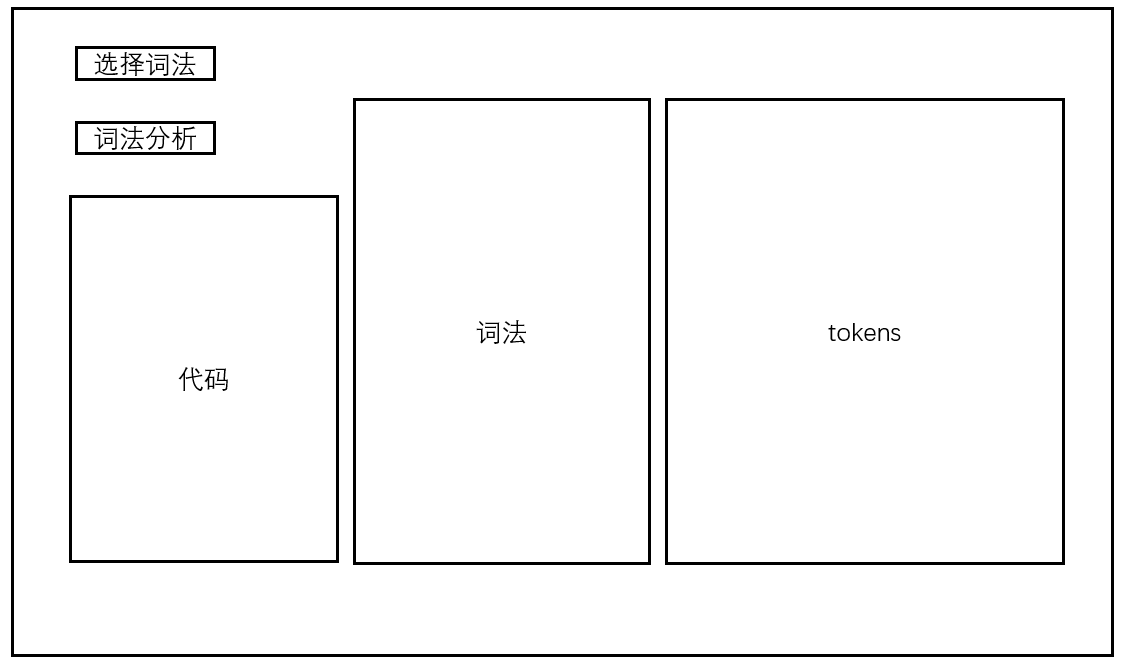
·选择代码按钮：单击此按钮，用户可以选择需要进行分析的代码文件，相应的界面中代码框会出现对应的文本内容；

·词法分析按钮：单击此按钮，用户可以跳转至词法分析界面；

·语法分析按钮：单击此按钮，用户可以跳转至语法分析界面；

·语义分析按钮：单击此按钮，用户可以跳转至语义分析界面。

### 2.2 词法分析界面设计



**图2-2 词法分析界面**

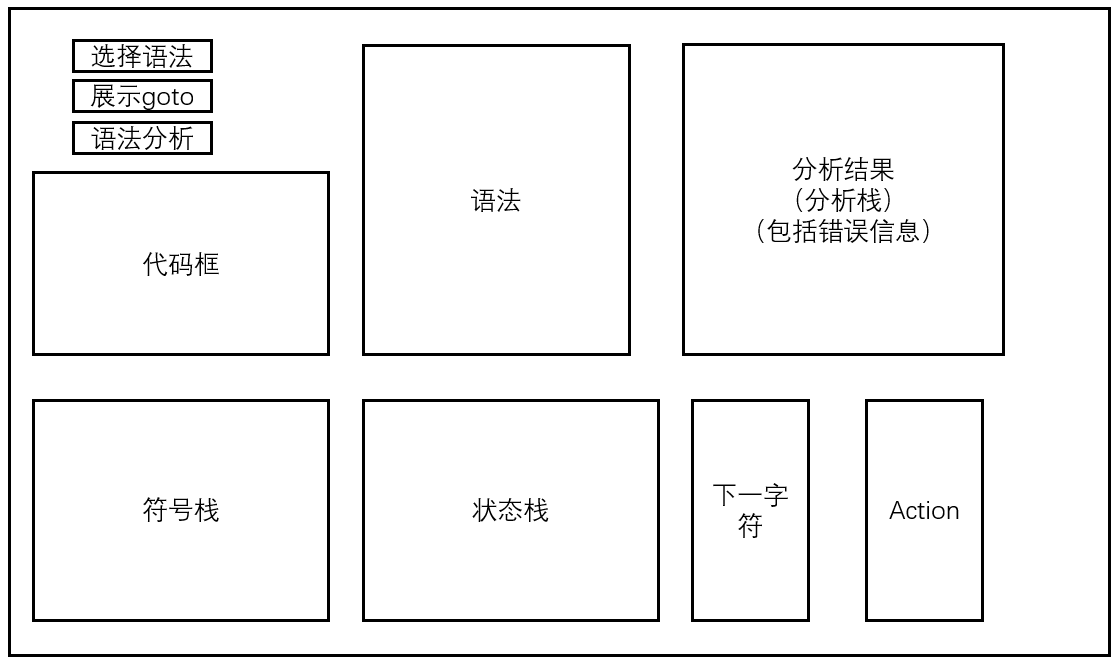
词法分析界面提供等词法分析的相关功能。

主要界面设计如下：

·选择词法按钮：单击此按钮，用户可以选择依赖的词法规则文件，词法框中会出现词法文件的文本内容；

·词法分析按钮：单击此按钮，用户可以进行词法分析，分析结果会显示在tokens框中。

### 2.3 语法分析界面设计



**图2-3 语法分析界面**

语法分析界面提供等语法分析的相关功能。

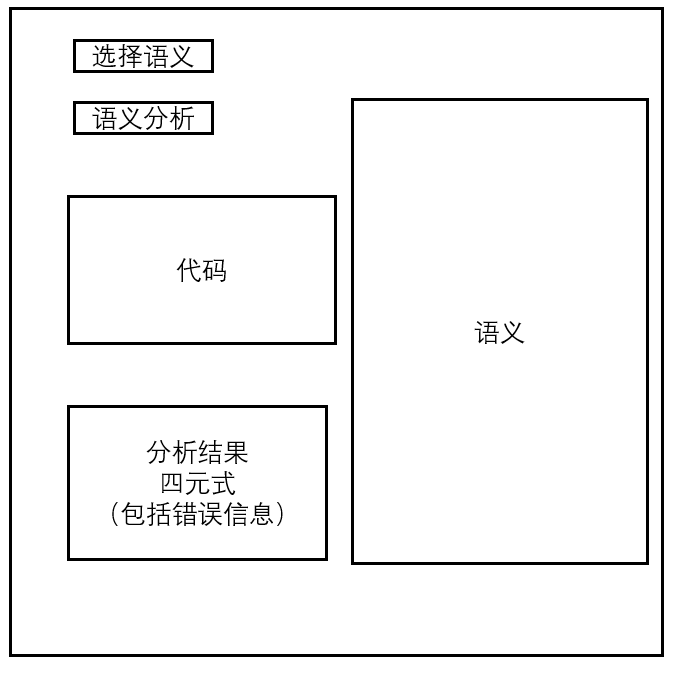
主要界面设计如下：

·选择语法按钮：单击此按钮，用户可以选择依赖的语法规则文件，语法框中会出现语法文件的文本内容；

·展示Goto按钮：单击此按钮，会显示依据用户选择的语法规则产生的Goto表界面；

·语法分析按钮：单击此按钮，用户可以进行语法分析，分析结果会显示在符号栈、状态栈、下一字符与Action框中。

### 2.4 语义分析界面设计



**图2-4 语义分析界面**

语义分析界面提供等语义分析的相关功能。

主要界面设计如下：

·选择语义按钮：单击此按钮，用户可以选择依赖的语义规则文件，语义框中会出现语义文件的文本内容；

·语义分析按钮：单击此按钮，用户可以进行语义分析，分析结果会显示在分析结果框中。

# 第三章 编译器系统详细设计

## 1. 详细设计概述

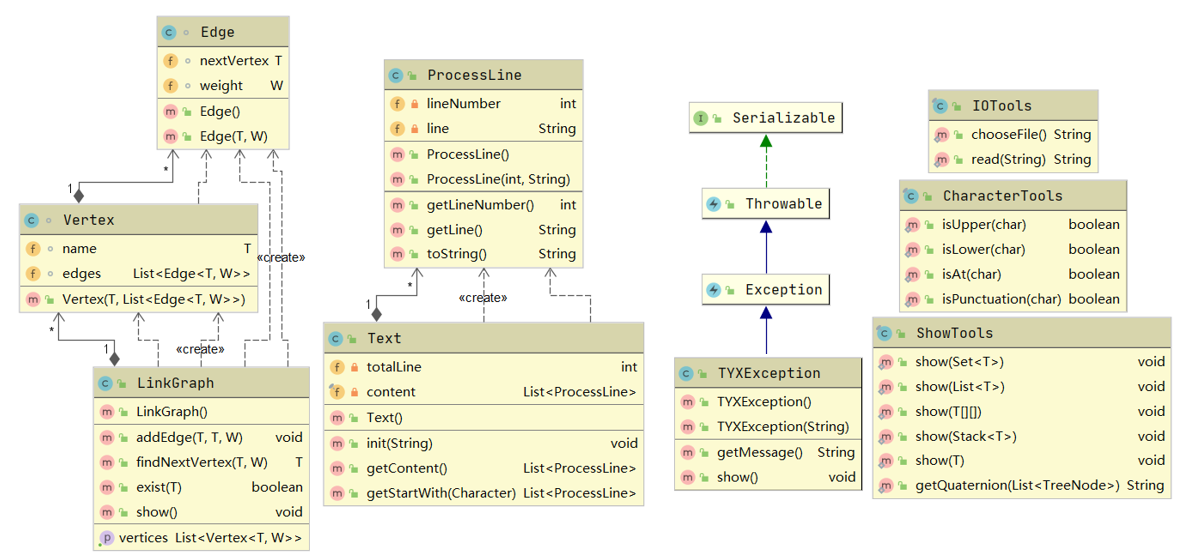
在本系统的开发过程中，由于考虑到代码复用与可持续性维护的问题，本人尽可能以软件设计模式的设计原则优化代码，如设计了图结构的泛型（在词法分析与语法分析中均有使用），也将部分不会变化的静态方法抽象成工具类，便于后续继续使用。

考虑到各类开发平台对Jar包的依赖路径问题，本项目以Maven为Jar包管理工具，增加了跨平台开发的可移植性。

考虑到各种操作系统对于前端界面的兼容性，本系统使用Swing作为用户界面开发工具包，屏蔽了操作系统的区别。

由于本文档的篇幅限制，本章节主要按照词法分析、语法分析与语义分析这三个功能，分别编制各模块的类图。

## 2. 辅助工具类



**图3-1 辅助工具类图**

### 2.1 图相关类

图相关类包括了Edge类、Vertex类与LinkGraph类。

Edge、Vertex、LinkGraph为以泛型为基础的图结构类，其中Edge为顶点，Vertex为边，LinkGraph为以邻接表构造的图结构。

### 2.2 文本处理类

文本处理类包括了ProcessingLine类与Text类。

ProcessingLine、Text为文本处理的辅助工具类，ProcessingLine封装了文本中某一行的内容、行号等信息，Text则封装了ProcessingLine的集合。

### 2.3 异常处理类

由于本系统涉及的异常类型并不多，因此本人在考虑异常类设计时只建立了一个异常类，即TYXException，多种异常类型则以message字段区分。

### 2.4 工具类

工具类包括IOTools类、CharacterTools类与ShowTools类。

其中，IOTools实现文本读入的弹窗功能，CharacterTools实现对于字符的操作，如判断是否为大小写、是否为运算符等，ShowTools使用泛型实现输出的功能，为程序员的调试、测试提供便利。

## 3. 词法分析

**图3-2 词法分析类图**

### 3.1 token相关类

token相关类包括了Token类与Tokens类。

Token为单个token，封装了如单词类型、单词行号等信息，Tokens为Token的集合。由于软件设计原则中强调了针对对象的操作，因此Token也提供了一些对自身的分离、化简方法，如分离关键字、删除空格等函数。

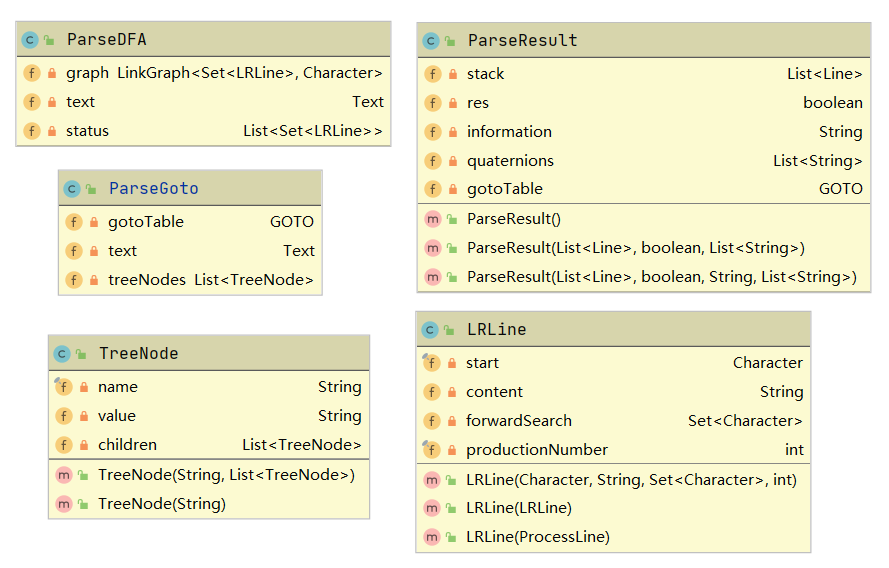
### 3.2 词法分析类

词法分析类包括了LexerNFA类、LexerDFA类与Lexer类。

LexerNFA为将词法文本转换成NFA的处理过程，其中的图结构是基于邻接表存储的；LexerDFA为将NFA转换成DFA的处理过程，其中的图结构是基于邻接矩阵存储的；Lexer为词法分析的封装类。

考虑到代码中只有标识符不固定，因此本人在编写词法时只针对标识符进行词法分析，其余的部分如常数、界符等依靠正则表达式及split函数分割。

## 4. 语法分析与语义分析



**图3-3 语法分析类图**

### 4.1 LRLine

由于LR分析中需要使用向前搜索符，因此单独设置LRLine类区别于ProcessingLine类。

### 4.2 ParseResult

为便于前端输出，特设置语法分析的结果封装类ParseResult，包括了语法分析栈的相关信息、分析结果、四元式、Goto表等信息。

### 4.3 TreeNode

TreeNode类为语法树的叶子结点，在语义分析的过程中被使用。

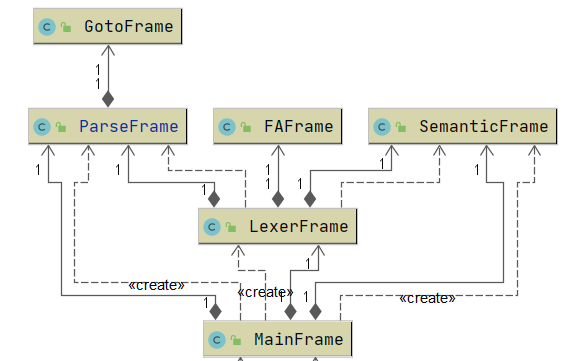
### 4.4 语法分析与语义分析类

语法分析与语义分析类包括了ParseDFA类与ParseGoto类。

其中，ParseDFA类主要处理语法文本到识别活前缀的DFA的转换，ParseGoto类主要处理Goto表的建立与代码的语法检查（语义分析包含在语法分析内）。

## 5. 用户界面（前端）

**图3-4 前端类图（无依赖）**



**图3-5 前端类图（含依赖）**

### 5.1 MainFrame

主界面：提供词法分析、语法分析与语义分析的导航功能。

### 5.2 LexerFrame

词法分析界面：提供等词法分析的相关功能。

### 5.3 FAFrame

NFA&DFA展示界面。

### 5.4 ParseFrame

语法分析界面：提供等语法分析的相关功能。

### 5.5 GotoFrame

Goto表界面：展示Goto表。

### 5.6 SemanticFrame

语义分析界面：提供等语义分析的相关功能。

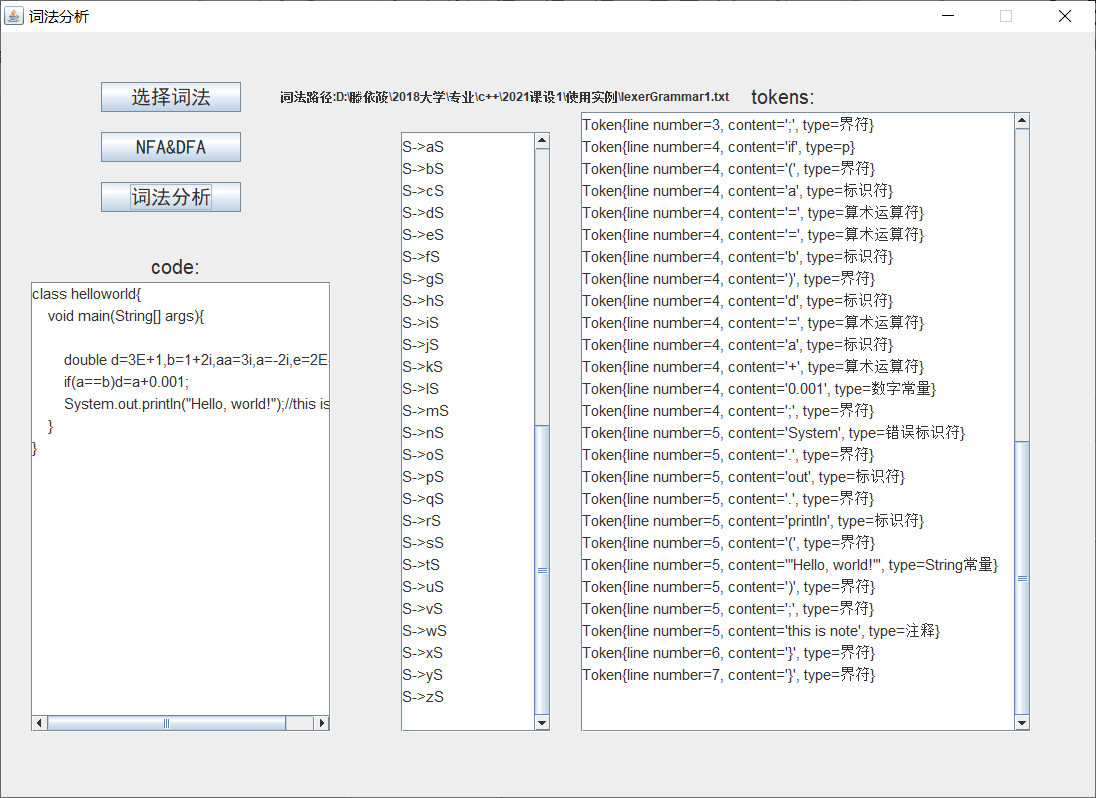
# 第四章 编译器系统实现效果

## 1.界面展示

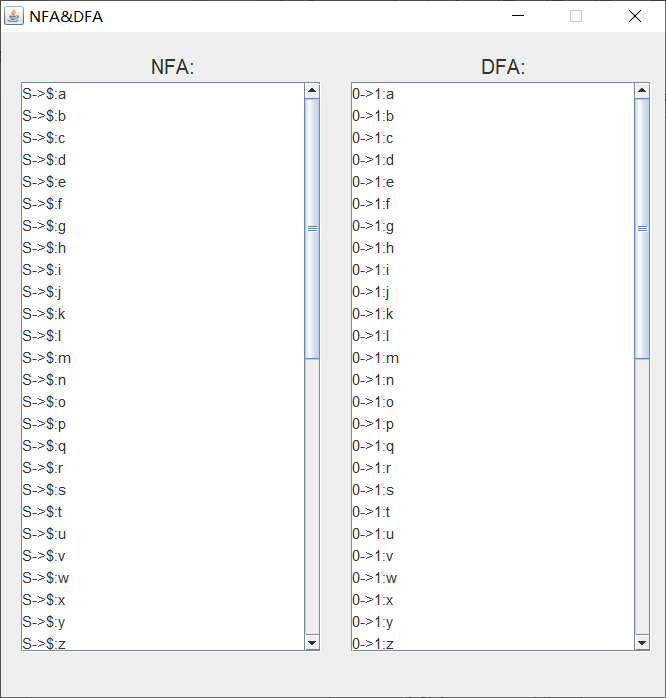
### 1.1 MainFrame



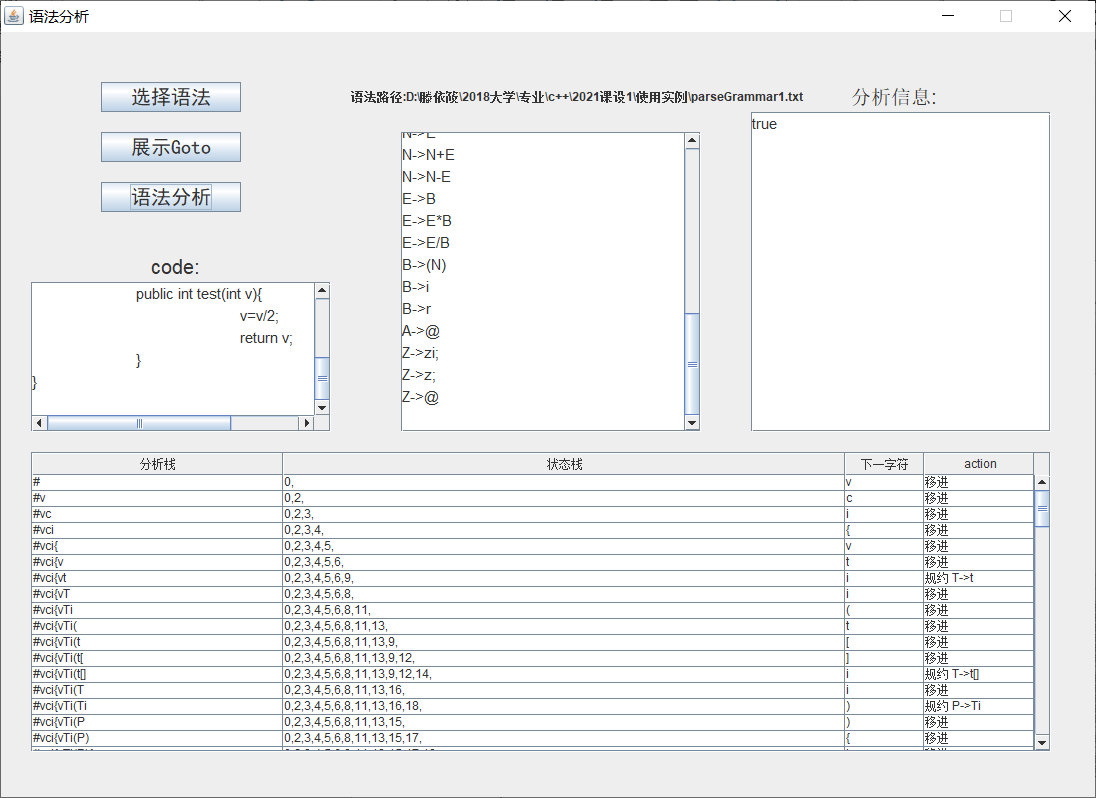
### 1.2 LexerFrame



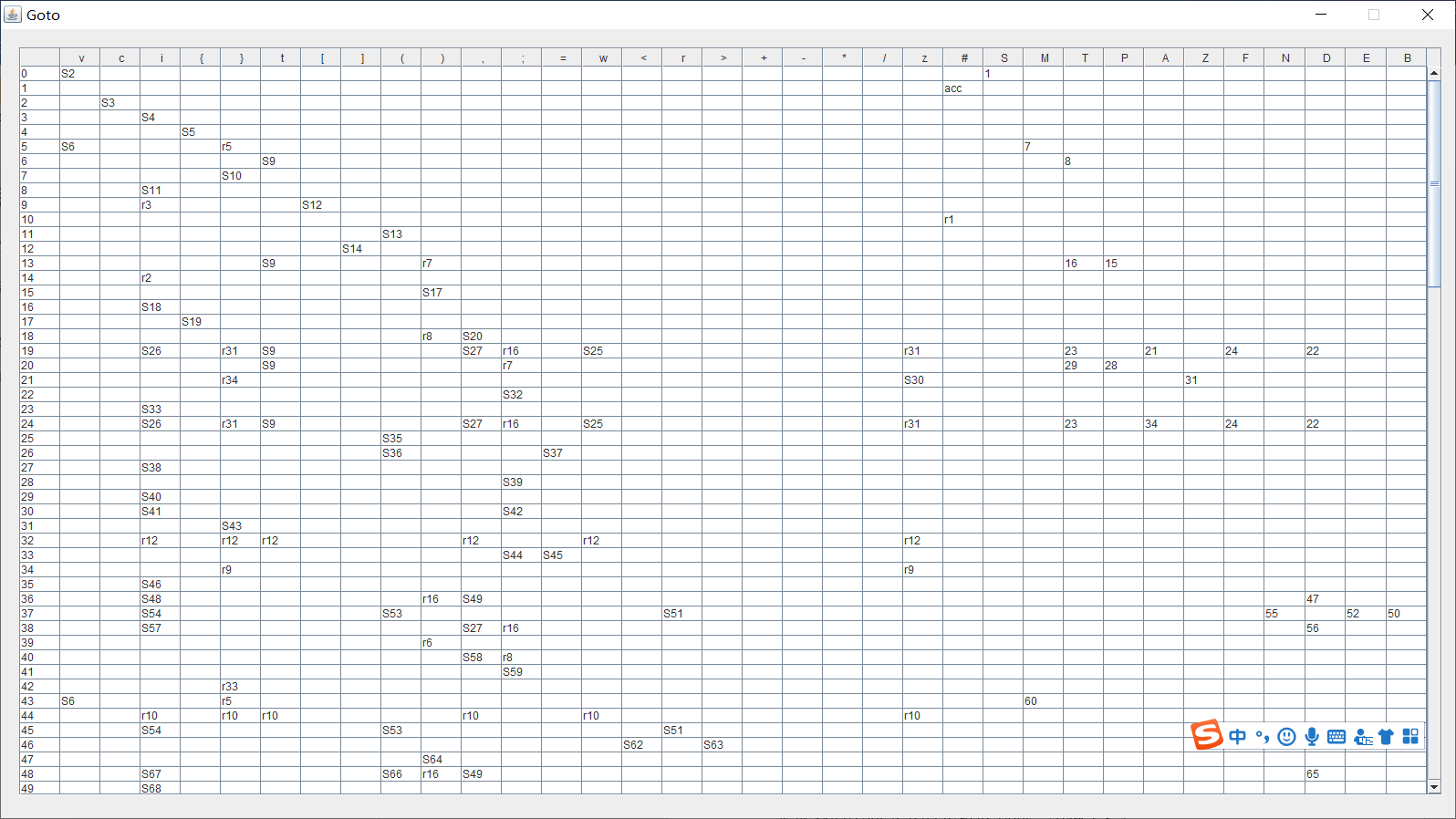
### 1.3 FAFrame



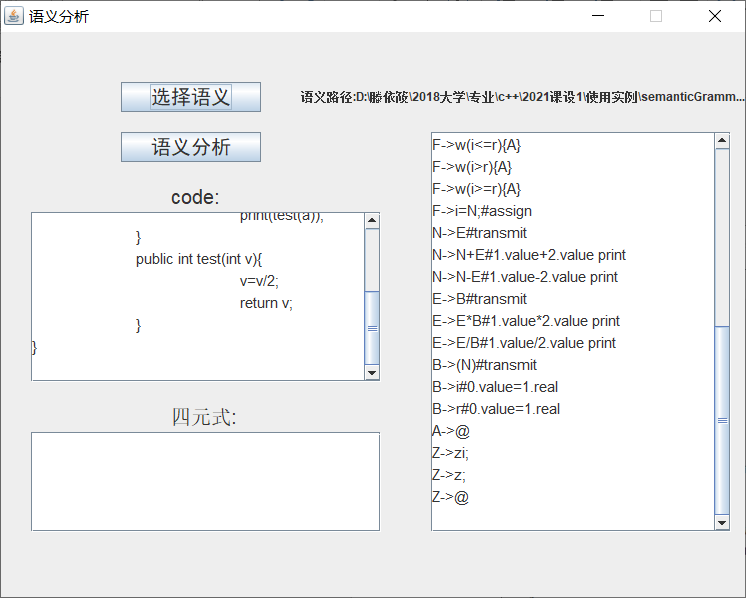
### 1.4 ParseFrame



### 1.5 GotoFrame



### 1.6 SemanticFrame



## 2.具体效果

具体效果详见使用实例。

# 第五章 编译器系统总结体会

## 1.总结和展望

本系统自本学期第一周开始设计与编写工作，第四周完成词法分析与语法分析，第七周完成语义分析，五一期间完成前端设计与编码。

技术方面，在系统核心代码的书写过程中，本人参考了许多软件设计模式中的设计原则，将一些可复用的静态方法分离为工具类，并且依靠泛型增强可复用性。在前端代码的书写过程中，本人使用了Swing技术以屏蔽底层操作系统的差异。

诚然，本系统也存在许多不足，例如词法分析在分离token的过程中，多次遍历文件文本以筛去会造成其他影响的单词，这降低了分析的效率。

这个系统的层次结构相对比较清晰，实体类与处理类、前端展示类分离，并且也设置了静态工具类。

## 2.感想

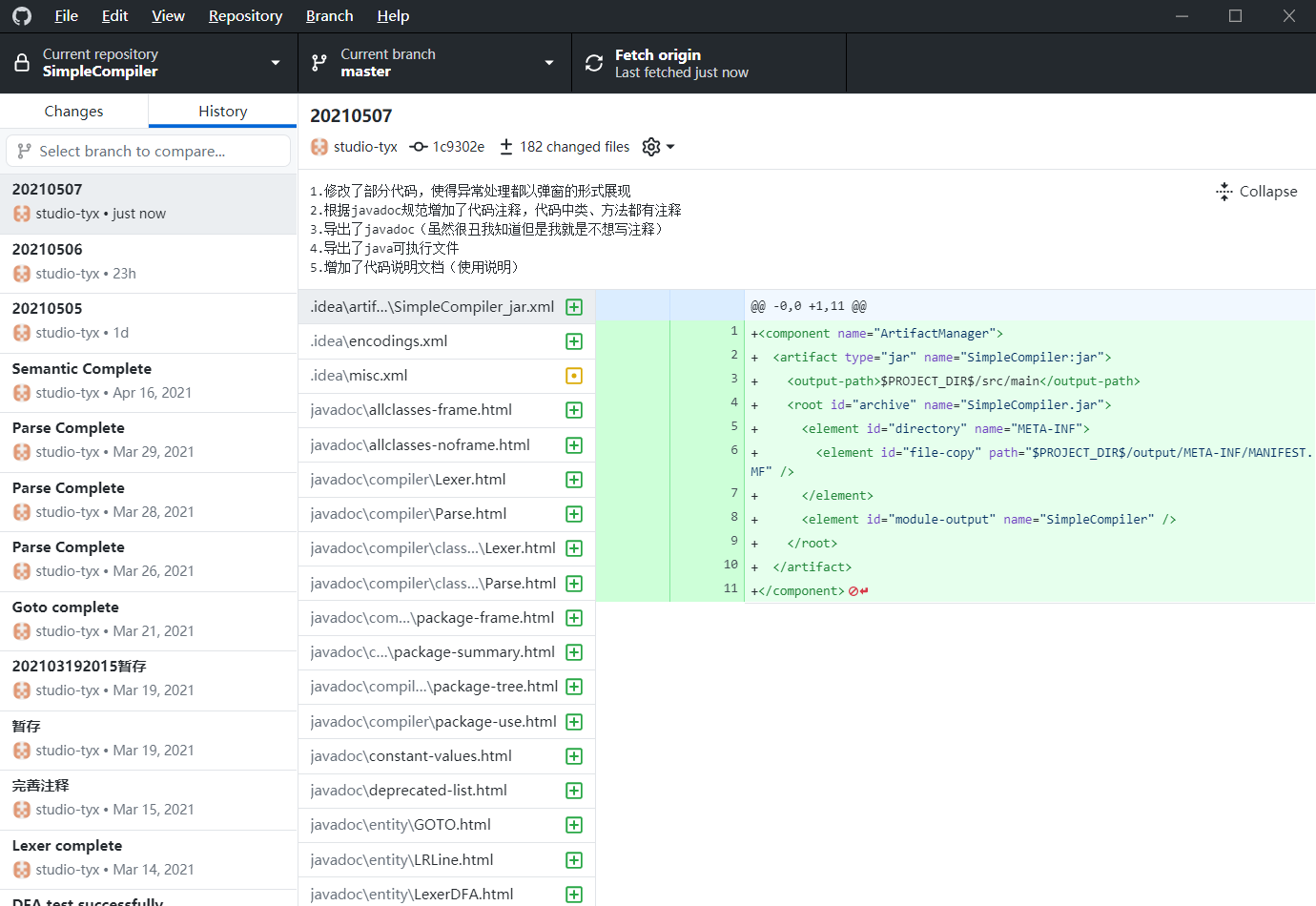
经过这次课程设计，我收获了许多。

第一，我对编译原理的核心知识更加了解了。犹记得在上学期的时候，编译原理那些“一支笔，一页纸，一道大题算一天”的作业题已经让我非常头疼了，没想到这学期还要把这些我用脑袋都不太弄的分明的东西“教”给电脑。我最初给自己的计划是在最空闲的前四周完成词法、语法和语义分析，然而现实是，我前四周只完成了词法和语法分析，在第五至七周课程稍多的时间段里，我完成了语义分析，在五一期间我才完成了前端的编码工作。显然，在这样理解-编码的过程中，我对词法分析、语法分析与语义分析的认识更加深入了。同时，我也更加能理解到编译器编写者的艰难之处，也许困难不在构造自动状态机，而是在构造语法，更何况那些中间优化功能。

第二，我对Java语言更加了解了。考虑到用户体验，我在这次项目的开发过程中，自学了Swing的相关知识，并且将其应用在这个项目中。在这次开发过程中，我也使用到了很多Java的封装类，从最基本的List到自学的Swing，我每一次使用这些封装类的方法，都会或多或少地翻到它们的源码，了解一些大佬的编码风范。

第三，我对软件设计的规范更加了解了。在学习过软件建模技术之后，我对对象的设计有了更深的认识，比如某个方法应该放在它改变的类中，而不是调用此方法的类。但是令我感到有些遗憾的是，我到项目结束才意识到自己需要撰写报告，因此我遗漏了在项目开发初期使用UML进行软件设计的机会。在项目初期，我几乎都是先写很多的注释或者开展非常多的调试，来确定自己的思路，在项目后期才逆向工程绘制类图流程图等。然而，非常幸运的是，在项目的开发过程中，我正好在学习软件设计模式这一课程，因此，每当设计模式老师讲解一个新模式的时候，我总会不自觉地去思考自己的项目是否应该应用此模式。也是在这样的原因下，我将我原先的代码做了一些优化调整，例如抽象出了基于邻接表的泛型图结构，也分理出了静态工具类。我也了解到javadoc的编写规范，javadoc对于方法、字段都需要增加注释。此时我更加敬佩那些开发源代码的程序员了，“编码一时爽，注释火葬场”，能写这么多注释真的是厉害。

第四，我对于IDEA工具的使用加强了，更重要的是了解到了IDE的重要性。在开展项目之前，我对开发工具是极为漠视的。但是在做课程设计的过程中，Maven包的架构、Jar包的导出、javadoc的导出，我都是借助IDEA完成的，我意识到了工具软件的重要作用。

第五，我深刻认识到了GitHub的妙处。这次的项目开发，我都是使用GitHub辅助完成的，每当我完成一个功能点，基本都会提交到GitHub上记录一下，以防下一次修改代码产生消极影响。看着GitHub上这两个月来满满当当的提交记录，我觉得很快乐。