**个人感悟与总结**

2015302580121 刘晓宇

1. **在项目中承担的任务**

在解释器这个项目中，我担任我们组的组长。

在项目初期，我和组员们进行了项目需求分析，考虑到了解释器关键代码的前后相关性、功能复杂性、代码工作量这三个方面，经过综合评估，我们进行了任务分工。我负责解释器的主体代码：词法分析、语法分析、语义分析，并完成UI界面设计代码。这样就可以保证即使附加特色功能因为某些原因不能实现，仍然可以产出一个可使用的版本，同时其余两位组员集中全力进行特色功能开发。

在开发过程中，我决定借助git工具实行多人协作高效开发。事实证明，这种开发方式效率很高，代码整合的效率很高，代码push记录也对组员的激励作用也比较大。

同时，在完成具有基本功能的解释器，并测试通过用例后，我又对界面进行了优化，加入了代码语法高亮及语法树展示等功能，并在最终版本整合测试完成后，撰写了项目说明文档。

在整个开发过程中，我还注重促进了组员间的交流，及时反馈自己负责部分的进度，及时交流代码的理解情况及需求的理解情况。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 项目需求 | 优先级 | 难度 | 代码量 | 负责人 |
| 词法分析 | 1 | 0 | 500 | 刘晓宇 |
| 语法分析 | 2 | 1 | 800 | 刘晓宇 |
| 语义分析 | 3 | 2 | 700 | 刘晓宇 |
| UI界面 | 4 | 2 | 1300 | 刘晓宇 |
| 语法特色 | 5 | 2 | 300 | 胡婷婷、刘晓宇 |
| 工具特色（语法高亮、语法树展示） | 6 | 3 | 300 | 刘晓宇 |

个人分工表

1. **个人实现**

**词法分析器**：从流中读取字符，分析，并产生一个含有词法单元的链表。

**语法分析器**：获得词法分析器产生的token列表，根据文法，采用自顶向下递归子程序法，构造语法树，返回语法树森林链表。在构造语法树的同时，会进行语法错误检查，并抛出语法错误异常。

**语义分析器**： 使用单例模式实现一个符号表，自顶向下递归遍历语法树结点，遇见变量声明或赋值等及时更新符号表，同时计算并保存结果到语法树结点中。在遍历语法树的同时，会进行语义错误检查，并抛出语义错误异常。语法树结点需要保存信息，这些信息包括：结点类型、结点字面值、结点数据值、结点数据类型、结点字符串字面值、结点布尔值、是否为中断结点（debug使用）以及结点的左中右子树，语义分析主要是围绕这些信息，在遍历时计算并保存。

**UI界面**:

使用SWT GUI框架。

包含打开文件、词法分析、语法分析。语义分析、调试、退出等按钮，代码区、输入区、输出区、调试信息区等文本框这些基本组件。

**语法高亮特色功能模块**：

借鉴Java语法高亮库，继承LineStyleListener类。

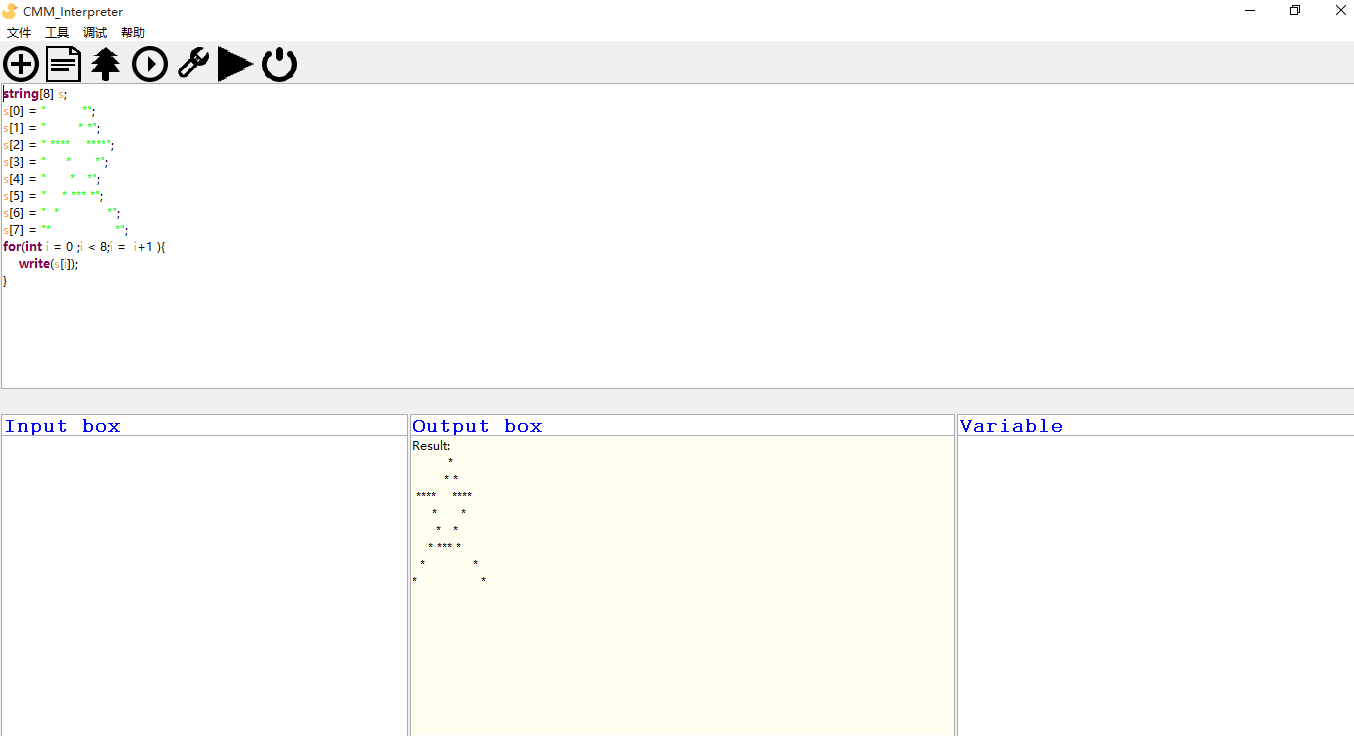
其基本思路为获取styledtext文本框的文本内容后，传入LineStyle类中处理，对于文本内容中的字符串、数字、标识符、关键字等不同token设置不同颜色与字体粗度。

**语法树图形化展示功能模块：**

此部分是为了方便语法分析及语义分析debug的工作，可以清晰看出语法树的结构。

实现思路为遍历语法树森林，自上而下递归遍历语法树、遇到结点时，使用画笔在Panel上画圆及结点类型或结点字面值，同时判断其是否存在子结点，若存在子结点，在父节点和子节点间连线，然后显示以该子节点为根节点的子树。其中值得注意的部分是，在不同层次的结点显示，需要不同的结点间距。

**个人实现示例：**

****

1. **感悟**

这个解释器课程项目是我比较喜欢的项目，不再仅仅停留在编程语言上的学习，而是更深一层地去了解编程世界里内在的工作，我觉得，让我的学习思维更加深入和开阔是我通过这门功课获取到的最大的收获。以前会纠结在各种不同语言的障碍上，很少有深层次的体会，例如，这次我们组的初始需求目标中是包含实现字节码生成和函数语法特色的，但是这两个任务的优先级不高，在我完成自己负责的部分后，我尝试实现这两个需求，于是去学习java虚拟机的知识，想通过了解java语言编译后的.class内容中的常量池等，得到启发并迁移到我们项目中去，但鉴于时间和精力有限，最终版本并没有加入这两个设计，但学习探究的过程我认为还是很有价值的。

其次，在项目的实战开发中，对数据结构的选取、对类与类的关系的设计、如何对报错信息和异常进行分析等过程，让我的编码能力和思考能力也得到明显的提升。这样一个规模并不算大的项目中，我也遇到了千奇百怪的bug，例如打开不同的文件运行时，会发生变量重名的错误，仔细一分析便会想到是使用了同一个符号表的原因，而使用同一个符号表又是因为我们选择的是为了安全统一使用单例模式获取符号表的原因，找到了缘由，解决便不再复杂，只需要在每次解释执行时先清空符号表即可。我们在实现debug功能时还遇到了解决了很长时间都没能解决的线程死锁问题，在尝试了多种不同的方案后，终于选择了另一种基于树的思路实现这个功能。总之，这一切经历给我的体会是，永远都不能眼高手低，设计的时候再完美，实际运行的时候总会出现各种自己没有想到的问题，这就是动手与动脑结合的重要性。

再者，令我感触也比较深的是组员之间的协调，在这种协同工作中，最不可缺少的便是责任心，我觉得我们组很幸运，每个组员都充满干劲和激情，同一个目标和具体的个人分工，便能够有条不紊地向目标靠近。我很享受集思广益并能够明确分工、互帮互助的开发氛围。

总之，这次实践项目让我更加深入地了解了编译或者解释一门编程语言的过程，为进一步的学习打下了坚实的基础，并通过协同合作的方式，让我切实体会到了合作的重要性和乐趣。