实验报告

1 项目结构

```
.
├── 201250190.pdf
├── Lab.iml
├── Makefile
├── Makefile.git
├── src
│   ├── Main.java
│   ├── myErrorListener.java
│   ├── SysYLexer.g4
│   ├── SysYParser.g4
│   └── Visitor.java
```

2 程序功能

实验二实现了将输入的代码段，在词法分析的基础上进行语法分析，从而判断代码段是否有语法上错误，最终得到正确代码段的语法结构树。

3 程序实现

首先依据 lab1 得到词法规则 SysYLexer，将代码段中的输入字符与词法规则一一匹配。再写一份语法规则 SysYParse，通过语法规则将输入的代码段进行结构上的分析，如果有出现语法规则的错误，则通过 myErrorListener 输出打印语法错误。如果代码段符合语法规则，则建立语法树 ParseTree，再通过 ANTLR 中的 Visitor 深度优先遍历 ParseTree，将对应的节点打印输出。

首先我们需要新建一个 Visitor 类，此类继承自 SysYParserBaseVisitor<Void>；其次，遍历节点的子节点调用 visitChildren(RuleNode node)函数，遍历终结符节点调用 visitTerminal(TerminalNode node)函数。这两个函数需要我们重写（override）。我们可以通过 node 分别获得对应的语法规则上下文(ruleContext)

和词法规则(Token)，接着便可以得到相应规则的索引和内容，进行内容的格式化之后，最后打印输出。特别的，我们需要对于终结符进行十进制的转换。

至于如何得到“语法规则上下文(ruleContext)和词法规则(Token)”，需要调用 node 中相应的 node.getRuleContext()和 node.getSymbol()函数；

在上一步得到的变量的的基础上，想要得到索引可以接着调用 ruleContext.getRuleIndex()和 token.getType()函数；

想要得到相应的规则名称，可以通过静态成员 SysYParser.ruleNames[]和 SysYLexer.ruleNames[]获得，其中终结符节点还需要额外获得叶节点中的原本内容 node.getText()

至此，我们已经获得全部输出内容，最后只需要按照格式打印输出即可。

4 精巧设计

在打印叶节点上，通过向上寻找父节点直至 null，得到叶节点的深度，而不是在父节点中判断下一个节点是否是叶节点。

5 印象深的地方

实验写的十分顺畅，除了阅读 API 方法以外基本上没有遇到较大的困难

一开始提交将缩进格式写成了 tab，后一次提交增加了进制转换