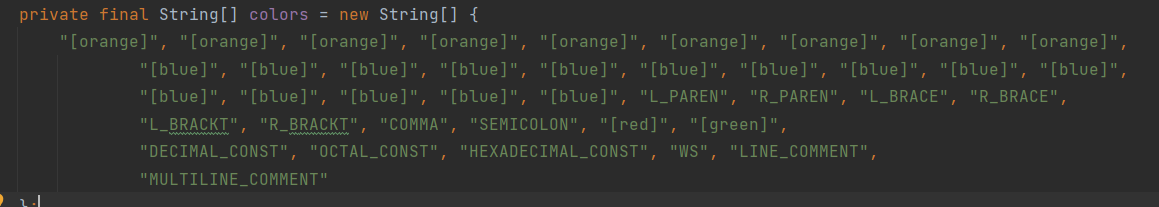
编译原理lab2

1. 具体实现
2. 首先按照Sysy语法规则以及助教实验手册编写SysyYParse.g4文件，之后为避免出现问题，虽然上次实验AC但使用的仍是助教实验手册中的SysyLexer.g4文件。生成tokens后将SysyLexer的tokens插入写好的SysyYParse.g4文件，并生成相应的类。
3. 之后在助教实验手册的提示下，编写Visitor继承自SysYParserBaseVisitor<Void>

成员变量：

private final String[] colors，此数组利用表驱动，在SysyLexer.rulesName[]表基础上，将有颜色的词对应的元素改为相应的颜色。



private String blanks来记录即将打印的要空几个空格。

四个函数：

Override了两个函数visitChildren，visitTerminal，创建了UpperFirstLetter和needSkip。

visitChildren：在方法源码基础上，在方法源码片段前，先打印(blanks+ruleName)，再将blanks加两个空格，在源码片段后减去两个空格，以此实现语法树的打印。

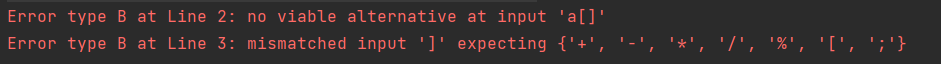
UpperFirstLetter：由于SysyYParse中的ruleNames均为首字母小写，所以写了此方法将字符串首字母变为大写。

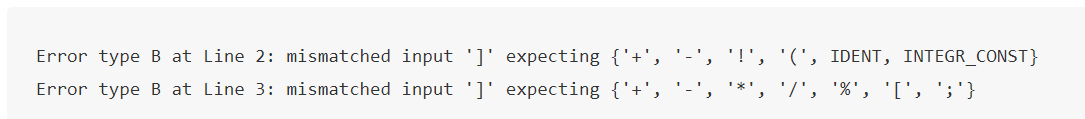
needSkip: 实验要求跳过如:, (, ), {, }, ;, 注释等非终结符，只打印终结符，故使用此方法。

本方法个人认为比较巧妙，因为终结符需高亮，即有颜色，所以判断一下符号是否有颜色即可判断是否为终结符。具体我是将上述成员变量中的colors[]中该元素对应的字符串是否和SysyLexer.rulesName[]表对应的字符串相同，若相同说明其没有颜色，即返回true，否则返回false。

visitTerminal: 除了上述needSkip判断是否为终结符外，此方法也判断了是否为十六进制或八进制，若是将其转换为十进制表示，具体方法和lab1中相同。之后将处理好的终结符带上colors[]表中查询到的颜色输出。返回值为super.visitTerminal()。

1. 错误处理和lab1相同，自己创建了一个MySysYParserErrorListener类，继承自 BaseErrorListener类，具体方法也是重写了syntaxError方法，并为判断是否有error新建了haserror成员变量。
2. Main函数则按照说明手册使用visitor深度优先遍历了语法树，但在创建visitor前多加了一个判断是否有报错，若有则直接return。
3. 遇到的问题以及解决方法：
4. 在写好代码后运行助教给的例子2发现结果和例子不同，如下图，但并不影响AC。





1. 起初我直接在antlr4工具生成好的类中进行代码修改，但在OJ上跑的时候发现，在编译时会重新生成该类，故要将改动放在新建的类中。
2. AC满分4300分，只得了4000分，发现要处理十六进制和八进制。