工作内容

原子结构

一般的，PA1-A的工作可以拆成如下几个原子单元：

1. 对于创建一个新关键词（比如"var"），需要：
    1. 在Tokens.java里添加并注册Token
    2. 在Decaf.jflex里添加关键词
    3. 在Decaf.jacc里添加关键词
    4. 在JaccParser.java里注册Token
    5. 在SemValue.java里注册Token
2. 对于修改一个已有方法以支持新特性（比如抽象类-ClassDef），需要：
    1. 在Decaf.jacc里添加语法
    2. 在Tree.java里修改构造函数，添加或修改一些变量或函数，以适应Decaf.jacc中的定义，需要参考其他类的实现
3. 对于添加一种新的抽象语法树的节点（一个新TreeNode类）以支持新特性（如函数类型-TLambda），需要：
    1. 在Decaf.jacc里添加语法
    2. 在Tree.java里添加Token
    3. 在Tree.java里添加新类以适应Decaf.jacc中的定义，需要参考其他类的实现
    4. 在Visitor.java里添加新方法，参考其他方法实现即可
4. 对于添加一种新的具体语法树的节点以支持新特性（如函数类型的typelist），需要：
    1. Decaf.jacc里添加语法
    2. 在AbstractParser.java里添加新方法SemValue AbstractParser.sv*，需要参考其他方法的实现
    3. 在SemValue.java里添加并注册Token

任务内容

对于本次任务，分为以下部分：

1. 抽象类：关键词"abstract"(1)、修改类ClassDef, MethodDef(2)
2. 局部类型推断：关键词"var"(1)、修改类LocalVarDef(2)
3. First-class Functions
    1. 函数类型
        1. 支持typeList的新具体语法树svTypes(4)、新增类TLambda(3)
    2. Lambda 表达式
        1. 关键词"=>"(1)、新增类Lambda(3)
        2. 在Decaf.jacc里修改优先级
    3. 函数调用
        1. 修改类Call(2)

回答问题

Q1. AST 结点间是有继承关系的。若结点 `A` 继承了 `B`，那么语法上会不会 `A` 和 `B` 有什么关系？限用 100 字符内一句话说明。

若A继承了B，则意味着A是一种B，即所有A的语法都会是B的语法。例如，ValSel继承了LValue，那么意味着所有满足变量的语法（例如a，是变量）都是LValue（左值）。但反过来一般不是，即一般并非所有B的语法都是A的语法。

Q2. 原有框架是如何解决空悬 else (dangling-else) 问题的？限用 100 字符内说明。

原框架下的Decaf.jacc含有这样一部分关于else的代码：

```
ElseClause      :   ELSE Stmt
                    {
                        $$ = $2;
                    }
                |   /* empty */             %prec EMPTY
                    {
                        $$ = svStmt(null);
                    }
                ;
```

这意味着else stmt比<empty>的优先级更高，因此先生成的ElseClause会优先绑定else stmt，也即else优先与最后出现的未绑定的if绑定。

Q3. PA1-A 在概念上，如下图所示：

```
作为输入的程序（字符串）
--> lexer --> 单词流 (token stream)
--> parser --> 具体语法树 (CST)
--> 一通操作 --> 抽象语法树 (AST)
```

输入程序 lex 完得到一个终结符序列，然后构建出具体语法树，最后从具体语法树构建抽象语法树。 这个概念模型与框架的实现有什么区别？我们的具体语法树在哪里？限用 120 字符内说明。

parser会在查看下一个单词时调用lex的接口，因此单词流与CST同步生成。

同样的，Decaf.jacc会在匹配节点后调用`$$=????`生成AST，因此AST也与CST同步生成。

以上两点是概念模型与框架的实现的区别。

具体语法树仅在匹配过程中暂时存在于parser，随即会在生成对应的AST后消除。