编译原理 PA1-A 实验报告

2017011620 计 73 李家昊

2020 年 1 月 27 日

1 工作内容

1.1 abstract 关键字

在 Tokens.java 中添加

```
int ABSTRACT = 31;
```

在 JaccParser.java 中添加

```
case Tokens.ABSTRACT -> decaf.frontend.parsing.JaccTokens.ABSTRACT;
```

在 Decaf.jflex 中添加

```
"abstract"          { return keyword(Tokens.ABSTRACT);     }
```

在 Decaf.jacc 中添加

```
%token ABSTRACT
```

在 Decaf.jacc 的 ClassDef 中添加规则

```
|   ABSTRACT CLASS Id ExtendsClause '{' FieldList '}'
{
    $$ = svClass(new ClassDef(true, $3.id, Optional.ofNullable($4.id)
        , $6.fieldList, $2.pos));
}
```

在 MethodDef 中添加规则，由于虚函数不能有定义，因此这里必须以分号结尾，并将 Block 改成 Optional<Block>，以支持空语句块。

```
|   ABSTRACT Type Id '(' VarList ')' ';'
{
    $$ = svField(new MethodDef(true, false, $3.id, $2.type, $5.
        varList, Optional.empty(), $3.pos));
}
```

同时需要在 Tree.java 的 ClassDef 和 MethodDef 的 Modifier 中增加 isAbstract 属性。

1.2 Var 关键字

仿照实现 abstract 关键字的步骤，即可实现 var 关键字的识别。此外，为支持 var 对应的 none 类型，需要将 LocalVarDef 的 TypeLit 类型改为 Optional<TypeLit> 。

1.3 Lambda 表达式

新建一个 TLambda 类型，继承 TypeLit，支持 Lambda 表达式的类型，内有两个成员，分别为返回类型（TypeLit）和参数类型列表（List<TypeLit>）。

新建 LambdaExpr 类型，继承 Expr，支持形如 fun (int x) => x 的表达式类型，内有两个成员，分别为参数列表（List<LocalVarDef>）和表达式（Expr）。

新建 LambdaBlock 类型，继承 Expr，支持形如 fun (int x) { return x; } 的语句块类型，内有两个成员，分别为参数列表（List<LocalVarDef>）和语句块（Block）。

仿照前两问的实现方法，增加 fun 关键字，并在 Type 中添加语法规则

```
|   Type '(' TypeList ')'
{
    $$ = svType(new TLambda($1.type, $3.typeList, $1.pos));
}
```

其中 TypeList 可仿照 VarList 实现，然后在 Expr 中添加语法规则

```
|   FUN '(' VarList ')' LAMBDA_ARROW Expr
    {
        $$ = svExpr(new LambdaExpr($3.varList, $6.expr, $1.pos));
    }
|   FUN '(' VarList ')' Block
    {
        $$ = svExpr(new LambdaBlock($3.varList, $5.block, $1.pos));
    }
```

需要注意的是，=> 操作符优先级最低，且不可结合。此外，Call 节点需要支持任意表达式，因此需要将 Call 规则改为 Expr '(' ExprList ')'，同时在 Call 接口做相应修改即可。

2 遇到的困难及解决方案

本次 PA 非常简单，除了需要一些时间阅读代码之外，没有遇到其他困难。

3 PA1-A 相关问题

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。

A1. 会。若 A 继承 B，则语法上 B 能生成 A，即 `B ::= A`。

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。

A2. 通过在 `ElseClause` 中定义空分支的优先级为 `EMPTY`，以及在优先级表中规定 `ELSE` 的优先级比 `EMPTY` 高，使得 `ELSE` 始终与最邻近的 `IF` 匹配。

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。

A3. 框架的实现不是分成 lex 和 parse 两个阶段，而是按需解析单词的，当 parser 需要时，就调用 lexer 的接口获取下一个单词。我们的具体语法树并未生成，parser 直接通过 `jacc` 定义的规则生成抽象语法树。