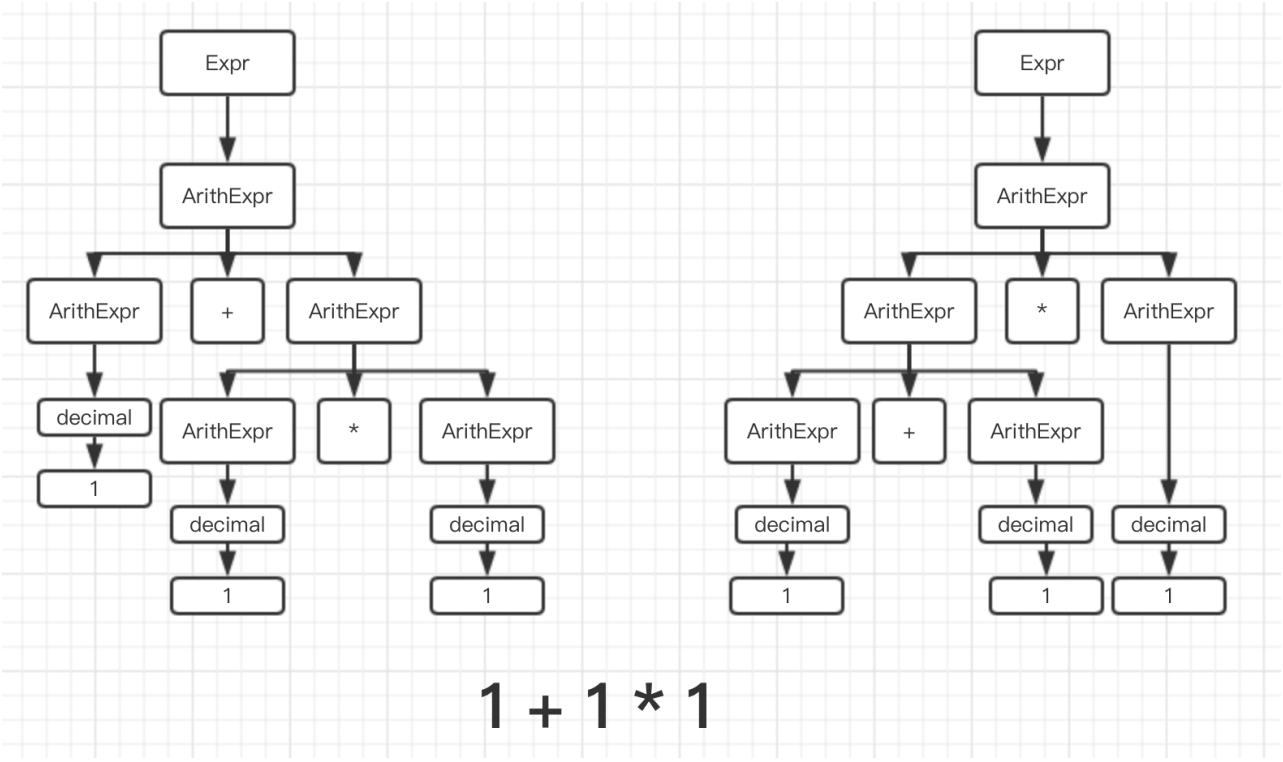
编译原理实验

基于表达式的计算器 ExprEval

15331260 邱兆丰

实验内容

1 讨论语法定义的二义性



答：如图所示，显然这个语法存在二义性，因为对于 `1+1*1` 可以构造出两棵parsing tree。ExprEval可以使用OOP运算符优先文法处理表达式。

2 设计并实现词法分析程序

2.1 Token分析及单词分类

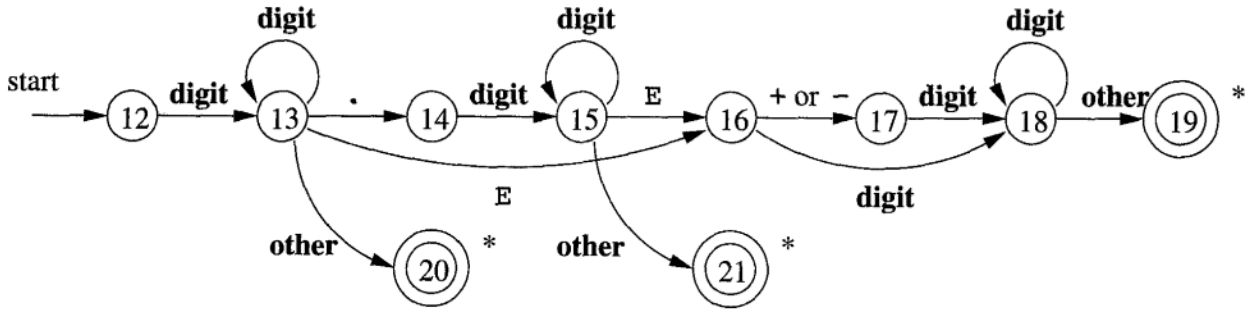
本程序的用到的 `Token` 有：

1. 数值类型的常量
   Number := [0-9]+(.[0-9]+)?(e(+|-)?[0-9]+)?
2. 布尔类型的常量
   Bool := true|false
3. 运算符
   Sign := [+-*/^()!&|?:>=<]
4. 预定义运算符
   Func := max|min|sin|cos
5. 标点符号
   Sign := [,]

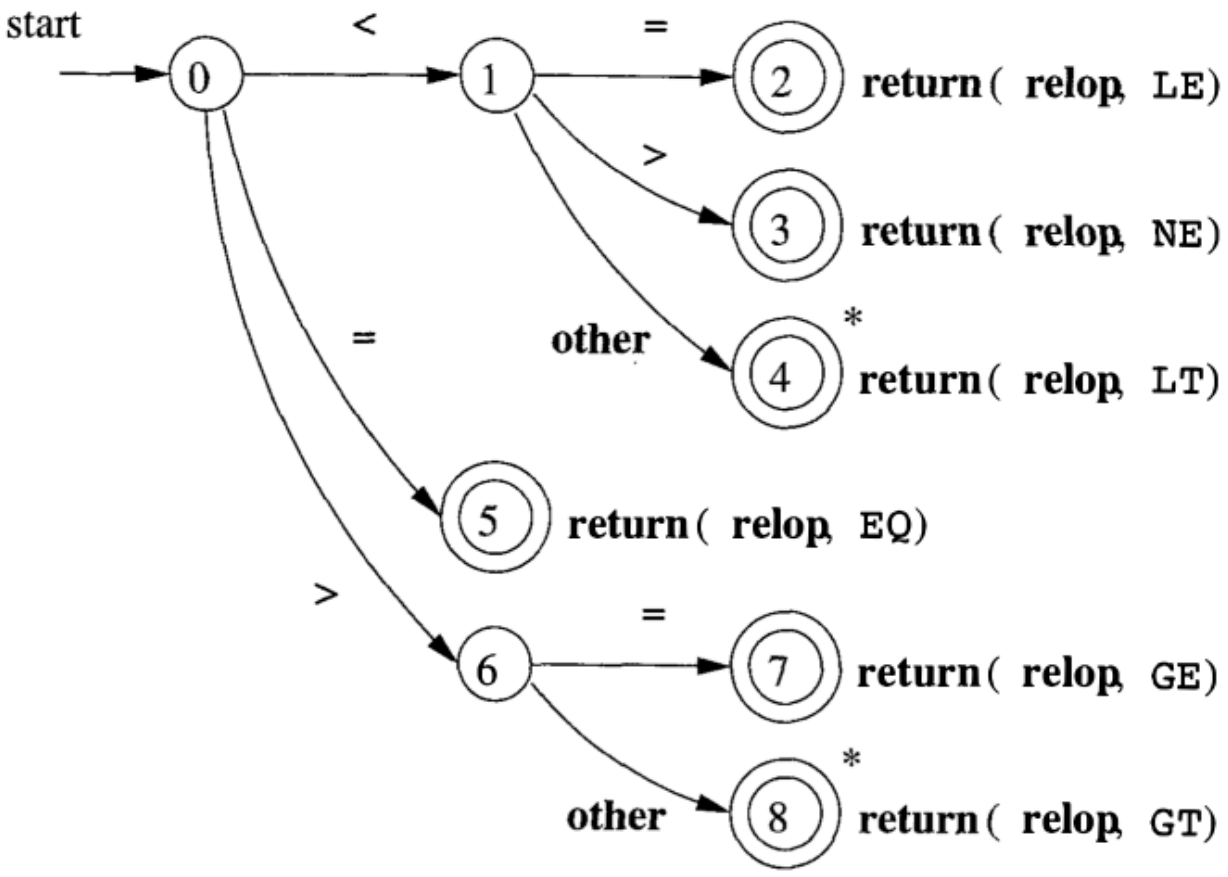
其中, 除了**运算符**和**预定义运算符**属于同一类外，其余符号各自为类，对于预定义函数名和布尔常量的识别问题，只需要按照本文2.2的状态转换图3，逐个判断其字符即可，而为了处理科学记数法表示的数值常量以及处理字符串的边界等，依然需要按照状态转换图去编程。

2.3 状态转换图

2.3.1 无符号数字的状态转换图

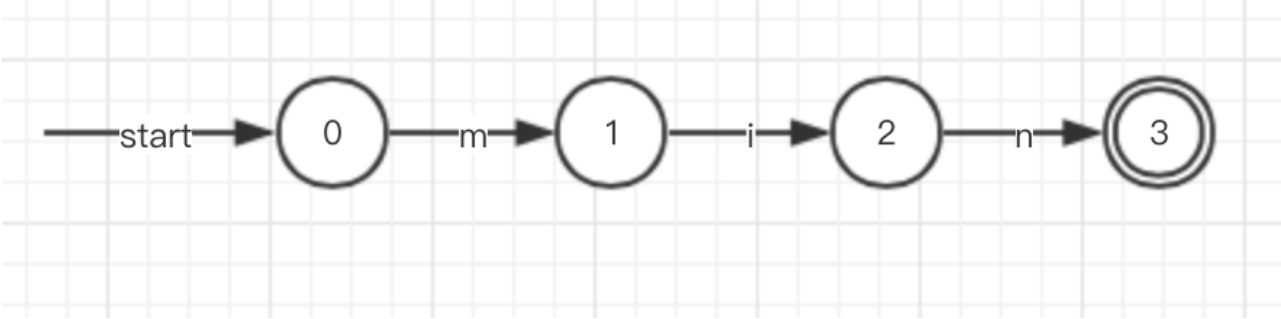


2.3.2 关系运算符的状态转换图



2.3.3 关键字状态转换图

其余关键字直接进行字符匹配，其状态转换图形式类比下图 `min` 的状态转换图。



其他直接进行字母归类

3 构造算符优先关系表

3.1 建表根据

在 ExprEval 支持的表达式中，各运算符的优先级与结合性质定义如下（其中未说明的结合性质默认为左结合）：

级别	描述	算符	结合性质
1	括号	( )	
2	预定义函数	sin cos max min	
3	取负运算（一元运算符）	–	右结合
4	求幂运算	^	右结合
5	乘除运算	* /	
6	加减运算	+ –	
7	关系运算	= <> < <= > >=	
8	非运算	!	右结合
9	与运算	&	
10	或运算	|	
11	选择运算（三元运算符）	? :	右结合

3.2 算符优先关系表

	num	bool	+-	*/	-	^	func	(	,	)	Re	!	&	|	?	:	$
num	E1	E1	>	>	>	>	>	E1	>	>	>	E5	>	>	>	>	>
bool	E1	E1	E5	E5	E5	E5	E1	E1	E5	>	E5	E1	>	>	>	>	>
+-	<	E5	>	<	<	<	<	<	>	>	>	E5	E5	E5	E5	>	>
*/	<	E5	>	>	<	<	<	<	>	>	>	E5	E5	E5	E5	>	>
-	<	E5	>	>	<	>	<	<	>	>	>	E5	E5	E5	E5	>	>
^	<	E5	>	>	<	<	<	<	>	>	>	E5	E5	E5	E5	>	>
func	E3	E3	E3	E3	E3	E3	E3	<	E3	>	E3	E3	E3	E3	E3	>	E3
(	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	<	E2
,	<	E5	<	<	<	<	<	<	<	<	E5	E5	E5	E5	<	<	E2
)	E1	E1	>	>	>	>	E1	E1	>	>	>	E1	>	>	>	>	>
Re	<	E5	<	<	<	<	<	<	E5	>	>	>	>	>	>	>	>
!	<	<	E5	E5	E5	E5	E5	<	E5	>	<	<	>	>	>	>	>
&	<	<	E5	E5	E5	E5	E5	<	E5	>	<	<	>	>	>	>	>
|	<	<	E5	E5	E5	E5	E5	<	E5	>	<	<	<	>	>	>	>
?	<	E7	<	<	<	<	<	<	E7	E6	<	E7	E7	E7	<	<	E7
:	<	E7	<	<	<	<	<	<	>	>	<	E7	E7	E7	>	>	>
$	<	<	<	<	<	<	<	<	E3	E3	<	<	<	<	<	E7	

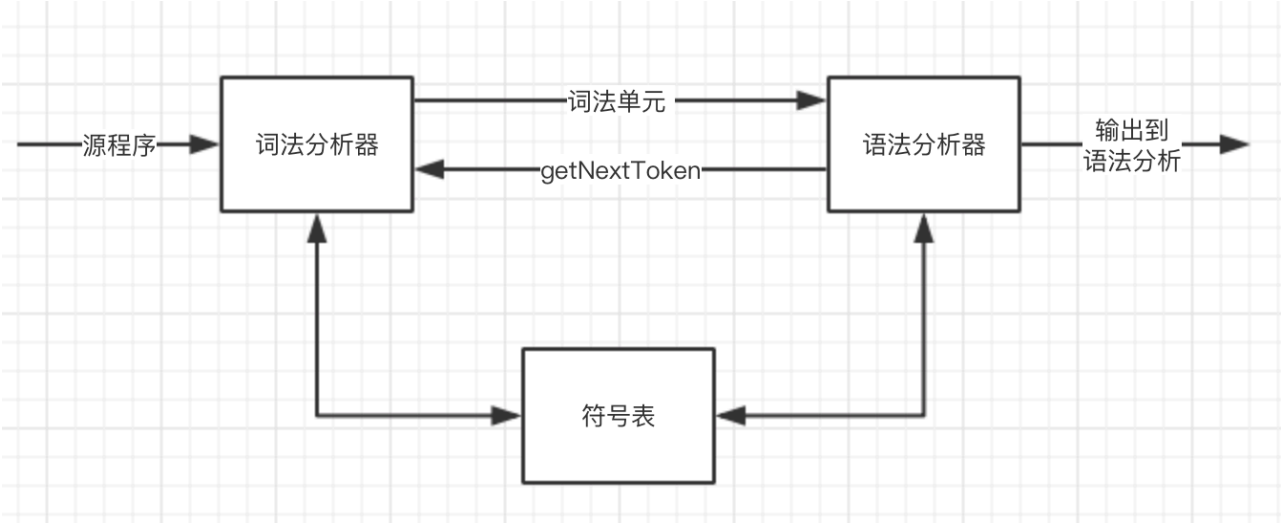
E1	MissingOperatorError
E2	MissingRightParenthesisException
E3	MissingLeftParenthesisException
E4	FunctionCallException
E5	TypeMismatchedException
E6	MissingOperandError
E7	TrinaryOperationError

表格注释：

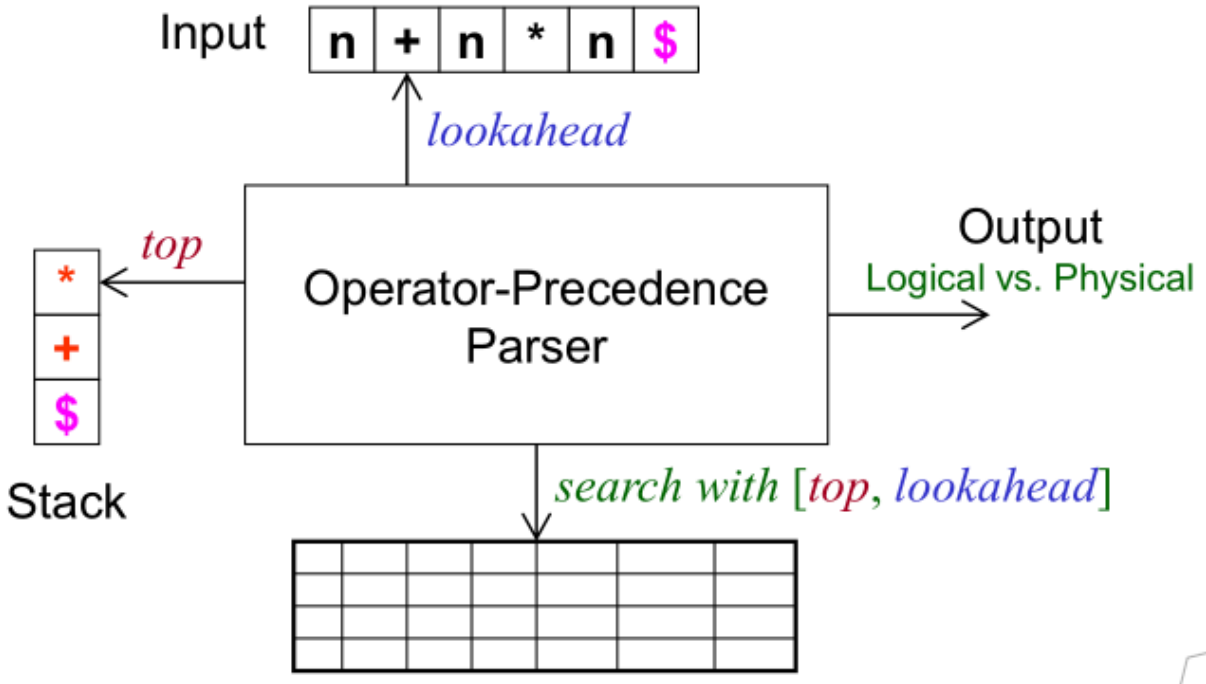
1. 右结合意味着该运算符遇到自身的时候是做shift操作。左结合则是reduce操作
2. `(` 的优先级最低，而 `)` 的优先级最高
3. `,` 也是一个Token，所以必须具有优先级。其优先级等同于 `(`
4. Re代表6个关系运算合集
5. 对于减号和负号，在scanner阶段获取token就进行判断，记录下前一个获取到token的类型，若前一个token为 `decimal` 或 `)`，则判定 `-` 为减号，否则判定为负号。

4 设计并实现语法分析和语义处理程序

4.1 词法分析器与语法分析器之间的交互



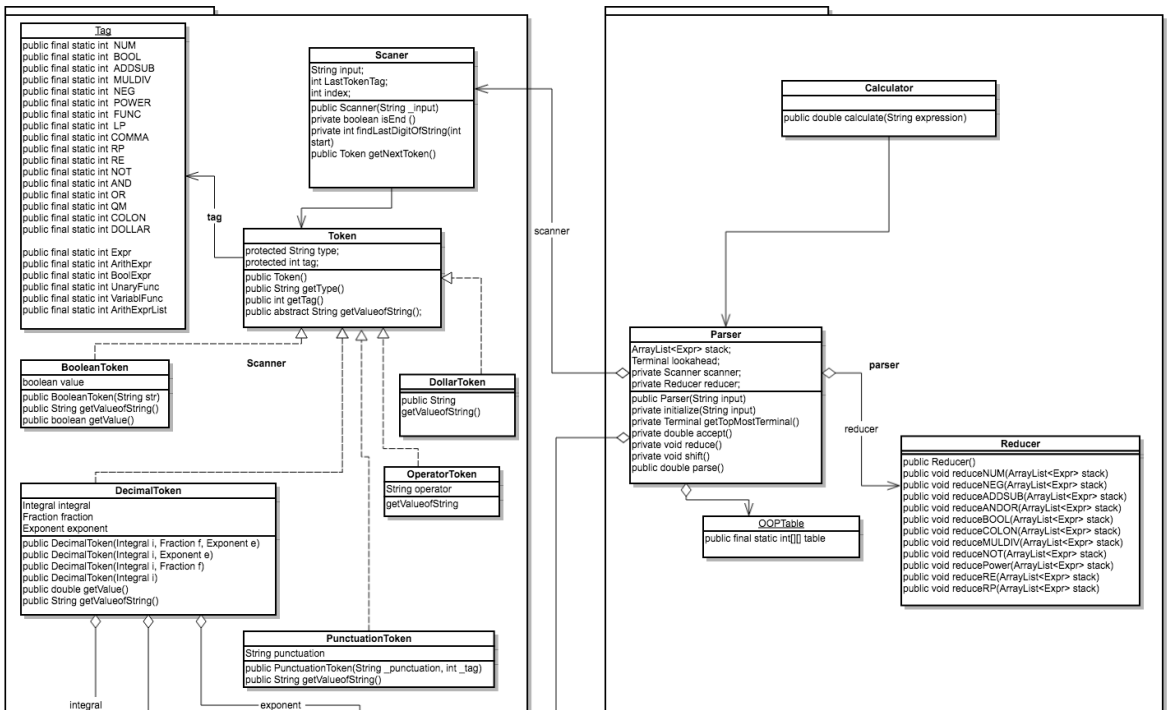
4.2 Operator Precedence Parser 工作原理图

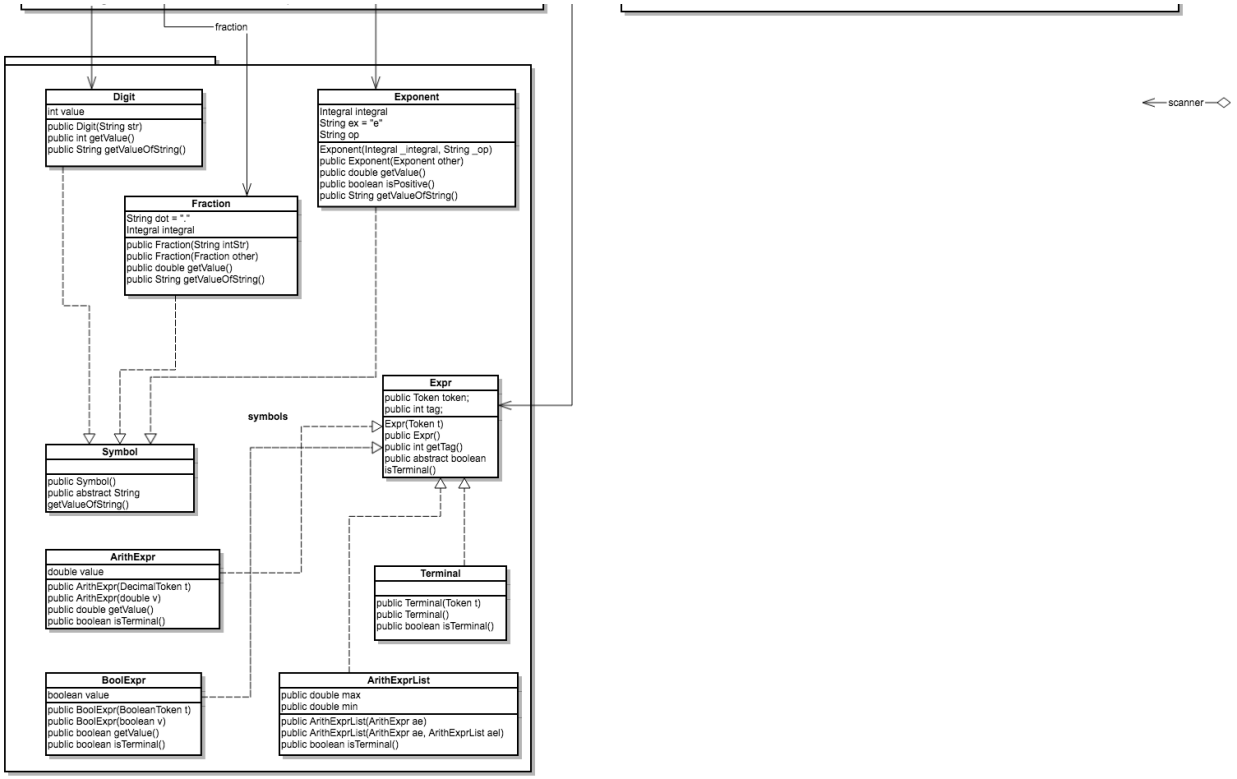


4.3 Parsing 算法参考

```
initialize();
for (;;) {
    if (top == $ && lookahead == $)  accept();
    topOp = stack.topMostTerminal();
    if (topOp ≺ lookahead || topOp ≡ lookahead) {  // shift
        stack.push(lookahead);
        lookahead = scanner.getNextToken();
    } else if (topOp ≻ lookahead) {  // reduce
        do {
            topOp = stack.pop();
        } while (stack.topMostTerminal() ≻ || ≡ topOp);
    } else  error();
}
```

4.4 UML图





备注：大图请查看同文件夹的UML.png

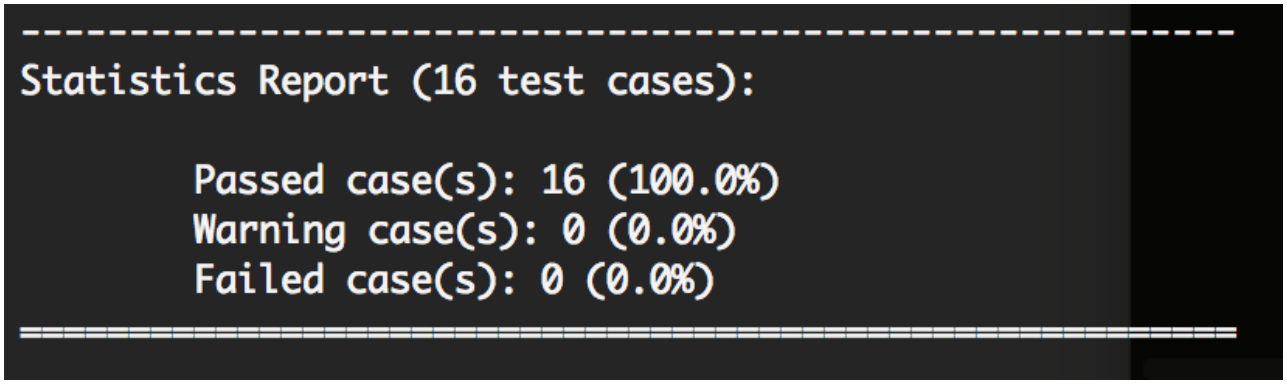
5 实验结果

simple测试：

```
--------------------------------------------------------------
Statistics Report (8 test cases):

        Passed case(s): 8 (100.0%)
        Warning case(s): 0 (0.0%)
        Failed case(s): 0 (0.0%)
==============================================================
```

standard测试


Statistics Report (16 test cases):
Passed case(s): 16 (100.0%)
Warning case(s): 0 (0.0%)
Failed case(s): 0 (0.0%)