

Informatik II

Postfix

Jan Lukas Deichmann

April 12, 2016

UML Diagramm

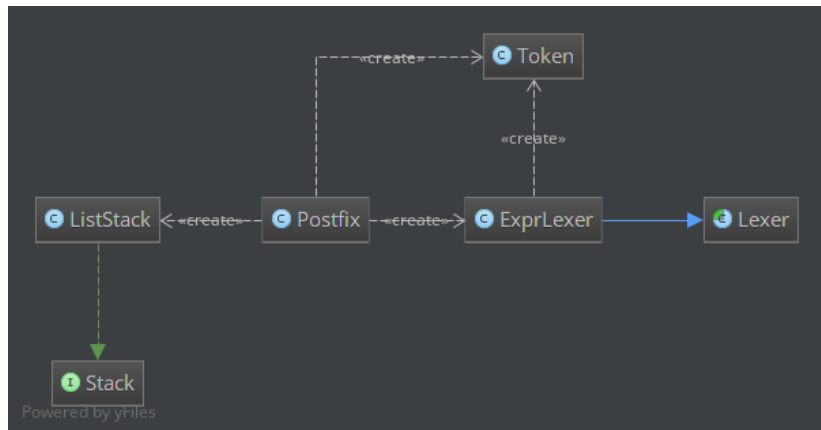
Code

ExprLexer

Postfix I

Postfix Test

UML Diagramm



ExprLexer

```
public static final int INT      = 2;  
public static final int PLUS    = 3;  
public static final int MINUS   = 4;  
public static final int UMINUS  = 5;  
public static final int MUL     = 6;  
public static final int DIV     = 7;  
public static final int POW     = 8;  
public static final int NL      = 9;
```

Postfix

```
public class Postfix {  
  
    public static int evalPostfix(String PostfixString) {  
  
        Stack<Token> tokenStack = new ListStack<Token>();  
        ExprLexer luthlexor = new ExprLexer(PostfixString);  
  
        Token currentToken = luthlexor.nextToken();  
        int v1, v2;  
        ...  
    }  
}
```

Postfix Binary

```
while (currentToken.getType() != ExprLexer.EOF_TYPE) {  
    switch ( currentToken.getType() ) {  
        case ExprLexer.PLUS:  
            v2 = Integer.parseInt(tokenStack.popTop().getText());  
            v1 = Integer.parseInt(tokenStack.popTop().getText());  
            tokenStack.push(new Token(ExprLexer.INT,  
                                     String.valueOf(v1 + v2)));  
            break;  
        ...  
    }  
    currentToken = luthlexor.nextToken();  
}
```

Postfix Binary

```
case ExprLexer.POW:  
    v2 = Integer.parseInt(tokenStack.popTop().getText());  
    v1 = Integer.parseInt(tokenStack.popTop().getText());  
    tokenStack.push(new Token(ExprLexer.INT, String.valueOf((int)  
    break;
```

Postfix Unary






```
case ExprLexer.UMINUS:  
    v1 = Integer.parseInt(tokenStack.popTop().getText());  
    tokenStack.push(new Token(ExprLexer.INT,  
                             String.valueOf(-v1)));  
    break;
```


Postfix Test

```
@org.junit.Test
public void testPostfix_test_add() throws Exception {
    assertTrue(evalPostfix("3 7 + \n") == 10);
}
```

```
@org.junit.Test
public void testPostfix_missing_newline() throws Exception {
    assertTrue(evalPostfix("3 # 3 8 4 / 2 ^ * 9 - +") == -1);
}
```

Postfix Coverage

Element	Class, %	Method, %	Line, %
 ExprLexer	100% (1/1)	85% (6/7)	91% (21/23)
 Lexer	100% (1/1)	66% (2/3)	75% (9/12)
 Postfix	100% (1/1)	100% (1/1)	97% (35/36)
 PostfixTest	100% (1/1)	100% (10/10)	96% (24/25)
 Token	100% (1/1)	75% (3/4)	75% (6/8)