

REPORT

파서 개발



주 제 : 파서 개발
교 수 : 이상범교수님
소속학과: 소프트웨어
학 번 : 32207825
이 름 : 정재호
마 감 : 2023-12-18

■ 목차

1. 프로그램 소스 코드 파서
 - A. Parser.java(3 pages)
 - B. Main.java(7 pages)
2. 프로그램 소스 코드 스캐너
 - A. CScanner.java(8 pages)
 - B. Token.java(11 pages)
 - C. Error.java(14 pages)
3. 입력 소스 코드(15 pages)
 - A. ProgramTest01.txt
4. 실행 결과(16 pages)
 - A. ProgramTest01.txt
5. 개발 후기(17pages)

● 프로그램 소스코드 파서

➤ Parser.java

```

1 package CScanner;
2 import java.util.Stack;
3
4
5 public class Parser {
6
7     public void InitiateParser(){
8         Stack<Integer> stateStack = new Stack(); //var, begin, end
9         Stack<Integer> flowStack = new Stack(); //상태
10        Stack<String> calStack = new Stack(); //연산 변수를 스택
11        Stack<String> calculStack = new Stack();
12        Stack<Float> IvarStack = new Stack();
13        Stack<Float> FvarStack = new Stack();
14
15        Object[][] intArray = new Object[5][2]; //int 변수를 테이블 어레이
16        Object[][] floatArray = new Object[5][2]; //float 변수를 테이블 어레이
17
18        String numberString = ""; //트큰번호 문자
19        String valueString = ""; //값 문자
20        String TOK = ""; //tok.toString()
21        String tokenName = ""; //트큰 명 변수
22        int code = 0; //트큰번호 숫자
23        float value = 0; //값 숫자
24        int NULLNUM = 777;
25        float tempVALUE = 0; //계산시 사용할 임시저장 변수
26
27        //변수 저장용 어레이 초기화
28        for(int i = 0; i < intArray.length; i++) {
29            for(int j = 0; j < 2; j++) {
30                intArray[i][j] = "NONE";
31            }
32        }
33        for(int i = 0; i < floatArray.length; i++) {
34            for(int j = 0; j < 2; j++) {
35                floatArray[i][j] = "NONE";
36            }
37        }
38
39        CScanner sc = new CScanner("D:\\ProgramTest01.txt"); // 스캐너 객체
40        Token tok = null; // 트큰을 저장하기 위한 변수
41        while ((tok = sc.getToken()).getSymbolOrdinal() != -1) { // 스캐너가 끝날때까지 트큰을 출력
42            // System.out.println(tok);
43            TOK = tok.toString();
44
45            //트큰 명
46            tokenName = TOK.substring(0, TOK.indexOf("@:"));
47
48            //트큰 넘버
49            numberString = TOK.substring(TOK.indexOf("@(") + 2, TOK.indexOf("@ "));
50            code = Integer.parseInt(numberString);
51
52            //트큰의 값
53            valueString = TOK.substring(TOK.indexOf("@ ") + 2, TOK.indexOf("@)"));
54            if(valueString.matches("[a-zA-Z]")){
55            }else {
56                value = Float.parseFloat(valueString);
57            }
58
59            //var, begin, end 상태 저장
60            if(code == 0) {
61                //System.out.println("state 0");
62                stateStack.push(code);
63            }else if(code == 1) {
64                for(int i = 0; i < stateStack.size(); i++) {
65                    stateStack.pop();
66                }
67                //System.out.println("state 1");
68                stateStack.push(code);
69            }else if(code == 2) {
70                for(int i = 0; i < stateStack.size(); i++) {
71                    stateStack.pop();
72                }
73                //System.out.println("state 2");
74                stateStack.push(code);
75            }
76

```

```

77 //44면 INT, 45면 FLOAT, 흐름스택에 저장
78 if(code == 44 && stateStack.peek() == 0) {
79     for(int i = 0; i < flowStack.size(); i++) {
80         flowStack.pop();
81     }
82     flowStack.push(44);
83 }else if(code == 45 && stateStack.peek() == 0) {
84     for(int i = 0; i < flowStack.size(); i++) {
85         flowStack.pop();
86     }
87     flowStack.push(45);
88 }else if(code == 37 && stateStack.peek() == 0) { //37 세미콜론을 만나면 흐름스택 비우기
89     for(int i = 0; i < flowStack.size(); i++) {
90         flowStack.pop();
91     }
92     flowStack.push(NULLNUM);
93 }
94
95 int count = 0; //변수 저장 어레이에 중복된 값이 있는지 확인하는 변수
96
97 //Int 변수명 저장
98 if(code == 3 && stateStack.peek() == 0 && flowStack.peek() == 44) {
99     for(int i = 0; i < intArray.length; i++) {
100         if (intArray[i][0] == valueString) {
101             count = 1;
102             break;
103         }
104     }
105     //중복된 값이 어레이에 없으면 변수명 저장
106     if (count != 1){
107         for(int j = 0; j < intArray.length; j++) {
108             if(intArray[j][0] == "NONE") {
109                 intArray[j][0] = valueString;
110                 count = 0;
111                 break;
112             }
113         }
114     }else {
115         System.out.println("duplicate ERROR");
116     }
117 }
118
119 //Float 변수명 저장
120 if(code == 3 && stateStack.peek() == 0 && flowStack.peek() == 45) {
121     for(int i = 0; i < floatArray.length; i++) {
122         if (floatArray[i][0] == valueString) {
123             count = 1;
124             break;
125         }
126     }
127     //중복된 값이 어레이에 없으면 변수명 저장
128     if (count != 1){
129         for(int j = 0; j < floatArray.length; j++) {
130             if(floatArray[j][0] == "NONE") {
131                 floatArray[j][0] = valueString;
132                 count = 0;
133                 break;
134             }
135         }
136     }else {
137         System.out.println("duplicate ERROR");
138     }
139 }
140

```

```

141         //begin 파트
142
143         // 변수명 에 값을 넣을 경우
144         if(code == 3 && stateStack.peek() == 1 && flowStack.peek() == NULLNUM) {
145             calStack.push(valueString);
146             flowStack.push(code);
147         }
148         // 변수명에 값을 넣는데, =를 만난경우
149         if(code == 15 && stateStack.peek() == 1 && flowStack.peek() == 3) {
150             flowStack.push(code);
151         }
152         // =인 상태에서 값을 만난경우
153         if(code == 5 && stateStack.peek() == 1 && flowStack.peek() == 15) {
154             IvarStack.push(value);
155         }
156         if(code == 3 && stateStack.peek() == 1 && flowStack.peek() == 15) {
157             calStack.push(valueString);
158         }
159         if(code == 10 && stateStack.peek() == 1 && flowStack.peek() == 15) {
160             flowStack.push(code);
161             calculStack.push(tokenName);
162         }
163         if(code == 3 && stateStack.peek() == 1 && flowStack.peek() == 10) {
164             calStack.push(valueString);
165         }
166         if(code == 12 && stateStack.peek() == 1 && flowStack.peek() == 10) {
167             flowStack.push(code);
168             calculStack.push(tokenName);
169         }
170         if(code == 3 && stateStack.peek() == 1 && flowStack.peek() == 12) {
171             float a = 0;
172             float b = 0;
173
174             calStack.push(valueString);
175
176             String tempA = calStack.pop();
177             for(int i = 0; i < intArray.length; i++) {
178                 if(tempA.equals(intArray[i][0])) {
179                     a = (Float) intArray[i][1];
180                     break;
181                 }
182             }
183             String tempB = calStack.pop();
184             for(int i = 0; i < intArray.length; i++) {
185                 if(tempB.equals(intArray[i][0])) {
186                     b = (Float) intArray[i][1];
187                     break;
188                 }
189             }
190             //곱하기 연산
191             tempVALUE = a * b;
192
193             calculStack.pop();
194             flowStack.pop();
195         }
196
197         if(code == 37 && stateStack.peek() == 1 && flowStack.peek() == 10){
198             float a = 0;
199             float b = 0;
200
201             String tempA = calStack.pop();
202             for(int i = 0; i < intArray.length; i++) {
203                 if(tempA.equals(intArray[i][0])) {
204                     System.out.println("FOUND");
205                     a = (Float) intArray[i][1];
206
207                     break;
208                 }
209             }
210
211             tempVALUE = a + tempVALUE;
212
213             String cal = calStack.pop();
214
215             for(int i = 0; i < floatArray.length; i++) {
216                 //floatVar에 floatArray에 저장된 변수명을 넣고 비교
217                 String floatVar = (String) floatArray[i][0];
218                 if(cal.equals(floatVar)) {
219                     float vf = tempVALUE;
220                     floatArray[i][1] = vf;
221                     break;
222                 }
223             }
224         }

```

```

225 //flowStack, calStack, calculStack 청소
226 for(int i = 0; i < flowStack.size(); i++) {
227     flowStack.pop();
228 }
229 for(int i = 0; i < calStack.size(); i++) {
230     calStack.pop();
231 }
232 for(int i = 0; i < calculStack.size(); i++) {
233     calculStack.pop();
234 }
235
236 calStack.push("NULL");
237 }
238
239
240 // 세미콜론이 flowStack = 과 만났을때
241 if(code == 37 && stateStack.peek() == 1 && flowStack.peek() == 15) {
242     String cal = calStack.pop();
243
244     for(int i = 0; i < intArray.length; i++) {
245         //intVar에 intArray에 저장된 변수명을 넣고 비교
246         String intVar = (String) intArray[i][0];
247         if (cal.equals(intVar)) {
248             float v = IvarStack.pop();
249             intArray[i][1] = v;
250             break;
251         }
252     }
253     for(int i = 0; i < floatArray.length; i++) {
254         //floatVar에 floatArray에 저장된 변수명을 넣고 비교
255         String floatVar = (String) floatArray[i][0];
256         if (cal.equals(floatVar)) {
257             float vf = FvarStack.pop();
258             floatArray[i][1] = vf;
259             break;
260         }
261     }
262
263     //flowStack, calStack 청소
264     for(int i = 0; i < flowStack.size(); i++) {
265         flowStack.pop();
266     }
267     for(int i = 0; i < calStack.size(); i++) {
268         calStack.pop();
269     }
270
271     calStack.push("NULL");
272 }
273
274 if(code == 7 && stateStack.peek() == 1) {
275     flowStack.push(code);
276 }
277 if(code == 3 && stateStack.peek() == 1 && flowStack.peek() == 7) {
278     for(int i = 0; i < intArray.length; i++) {
279         //intVar에 intArray에 저장된 변수명을 넣고 비교
280         String intVar = (String) intArray[i][0];
281         if (valueString.equals(intVar)) {
282             System.out.println("PRINT RESULT : " + intArray[i][1]);
283             break;
284         }
285     }
286     for(int i = 0; i < floatArray.length; i++) {
287         //floatVar에 floatArray에 저장된 변수명을 넣고 비교
288         String floatVar = (String) floatArray[i][0];
289         if (valueString.equals(floatVar)) {
290             System.out.println("PRINT RESULT : " + floatArray[i][1]);
291             break;
292         }
293     }
294 }
295 }
296 //테이블 출력
297 System.out.println("\n[INT TABLE]\n");
298 for (Object[] row : intArray) { //변수 테이블 확인을 출력은
299     for (Object element : row) {
300         System.out.print(element + "\t");
301     }
302     System.out.println();
303 }
304 System.out.println("\n[FLOAT TABLE]\n");
305 for (Object[] row : floatArray) { //변수 테이블 확인을 출력은
306     for (Object element : row) {
307         System.out.print(element + "\t");
308     }
309     System.out.println();
310 }
311 }
312 }

```

➤ Main.java

```
1 package CScanner;
2
3 public class Main {
4     public static void main(String[] args) {
5
6         Parser par = new Parser();
7         par.InitiateParser();
8     }
9 }
10 }
```

● 프로그램 소스 코드 스캐너(파란부분 : 원본코드와 차이가 있는부분)

➤ CScanner.java

```
package CScanner;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

public class CScanner {
    static public final int ID_LENGTH = 16; // 명령의 길이 제한
    static public final char EOF = '\255'; // EOF는 파일의 끝을 의미합니다
    static public final String SPECIAL_CHARS = "!<=>"; // 두글자로 조합가능한 특수문자들

    private String src; // 소스코드를 String으로 저장해줄 변수
    private Integer cursor; // 소스코드를 읽을 때 커서가 될 변수

    // 어떤 토큰인지 인식하기 위해 만든 상태들
    private enum State {
        Initial, Dec, Oct, Hex, Real, Const, tidOrReserved, Op, Zero, PreHex, PreReal, PreConst, SingleOperator
    }

    // 생성자
    public CScanner(String filePath) {
        src = parseFile(filePath);
        cursor = 0;
    }

    // 소스코드를 String으로 읽어주는 메소드
    private String parseFile(String filePath) {
        String src = "", readedString = ""; // src: 소스코드를 저장해줄 String 변수, readedString: 소스코드의 한줄을 담아줄 String 변수
        FileReader fileReader = null; // 소스코드를 읽을 File Reader
        try {
            fileReader = new FileReader(new File(filePath)); // 파일 경로를 이용해 File Reader 생성
        } catch (IOException e) {
            // 파일을 읽을 수 없음
            System.err.print(Error.getMessage(Error.ErrorCode.CannotOpenFile));
            return "";
        }

        BufferedReader reader = new BufferedReader(fileReader); // BufferedReader 객체를 만들어 소스코드 파일을 읽음
        try {
            while ((readedString = reader.readLine()) != null) // 파일로부터 한줄 읽음
                src += readedString + "\n"; // 한줄 완 뒤에 개행문자 추가
            src += EOF; // 파일의 끝을 의미하는 EOF 문자 추가
            reader.close();
        } catch (IOException e) {
            // 파일을 읽을 수 없음
            System.err.print(Error.getMessage(Error.ErrorCode.CannotOpenFile));
            return "";
        }
        return src;
    }

    // EOF인지 검사하는 메소드
    private boolean isEOF(int idx) {
        return idx >= src.length();
    }

    // 주석 처리 메소드
    private boolean exceptComment() {
        char c;
        // 커서로부터 whitespace 문자들 모두 무시
        while(!isEOF(cursor) && Character.isWhitespace(src.charAt(cursor))){
            cursor++;
        }
        if(isEOF(cursor)) {
            return false; // 공백을 무시하고 EOF면 주석 제거 성공하면 false 반환
        }

        if (src.charAt(cursor) == '/') { // '/'가 나올 경우
            if(src.charAt(cursor+1) == '?') { // ?? 는 블록 주석
                cursor += 2; // "?/" 다음 문자로 커서 이동
                while(src.charAt(cursor) != '?' && src.charAt(cursor+1) != '/') { // ?/가 나올때 까지 반복
                    if(isEOF(cursor+1)) return true; // 찾지 못하면 실패, true 반환
                    cursor++;
                }
                cursor += 2; // "?/" 다음 문자로 커서 이동
            }
        }
        return false; // 성공적으로 제거하면 false 반환
    }
}
```



```
// 문자가 1글자 연산자인지 확인하는 메소드
private boolean isSingleSpecialToken(char c) {
    switch (c) {
        case '(': case ')': case ',': case ';': case '[': case ']': case '{': case '}':
        case '+': case '-': case '*': case '/': case '%': case EOF:
            return true;
        default:
            return false;
    }
}
```

```
// 문자가 1글자 연산자가 아닌 특수문자인지 확인하는 메소드
private boolean isSpecialChar(char c) {
    for (int i = 0; i < SPECIAL_CHARS.length(); ++i)
        if (SPECIAL_CHARS.charAt(i) == c)
            return true;
    return false;
}
```

```
private Token.SymType getSymbolType(State s) {
    switch (s) {
        case Dec: // 10, 8, 16, 0은 Digit 반환
        case Oct:
        case Hex:
        case Zero:
            return Token.SymType.Digit;
        case Real: // 실수인 경우 Real 반환
            return Token.SymType.Real;
        case Const: // 상수인 경우 Const 반환
            return Token.SymType.Const;
        case tidOrReserved: // 명칭이나 예약어인 경우 tidOrReserved 반환
            return Token.SymType.tidOrReserved;
        case Op: // 연산자인 경우 OP 반환
        case SingleOperator:
            return Token.SymType.Op;
        case Initial: // 종결상태가 아닌 경우에는 실패이므로 NULL을 반환
        case PreHex:
        default:
            return Token.SymType.NULL;
    }
}
```

```
public Token getToken() {
    Token token = new Token();
    Token.SymType symType = Token.SymType.NULL; // Symbol Type을 NULL로 설정
    String tokenString = "";

    State state = State.Initial;

    // 현재 커서로부터 Comment 제거
    if (exceptComment()) {
        // Comment를 지우는 도중에 ERROR가 발생했을 경우
        System.err.print(Error.getErrorMessage(Error.ErrorCode.InvalidComment));
        return token;
    }
}
```

```
while(!isEOF(cursor)) {
    char c = src.charAt(cursor++);

    if(Character.isWhitespace(c)) {
        if(state != State.Initial) break;
        else continue;
    }else if(state == State.Initial && c == '0') {
        state = State.Zero;
    }else if(Character.isDigit(c)) { //숫자를 만났을때 이전의 상태에 따라 현재상태 정의
        if(state == State.Initial) {
            state = State.Dec;
        }else if(state == State.Zero) {
            state = State.Oct;
        }else if(state == State.PreHex) {
            state = State.Hex;
        }else if(state == State.PreReal) {
            state = State.Real;
        }else if(state == State.PreConst) {
            state = State.Const;
        }else if(state == State.Op) { // 연산자가 나온 다음 숫자면 break
            --cursor;
            break;
        }
    }
}
```

```

} else if (state == State.Dec && c == '.') { // '숫자.' 을 인식한 경우
    state = State.PreReal;
} else if (state == State.Zero && c == '0') { // '0.' 을 인식한 경우
    state = State.PreReal;
} else if (state == State.Zero && c == 'x') { // '0x' 을 인식한 경우
    state = State.PreHex;
} else if (state == State.Dec && c == 'e') { // '숫자.e' 을 인식한 경우
    state = State.PreReal;
} else if (state == State.PreReal && c == '+') { // 'e+' 을 인식한 경우
    state = State.Real;
} else if (state == State.Initial && c == '\\') { // '\\' 을 인식한 경우
    state = State.PreConst;
} else if (state == State.Const && c == '\\') {
    state = State.Const;
} else if (state == State.PreConst) { // 상수
    state = State.Const;
} else if (isSingleSpecialToken(c)) { // '(', ')', '{', '}', ',', '[', ']', ';', EOF 을 인식한 경우
    if (state == State.Initial) {
        state = State.SingleOperator;
        tokenString = String.valueOf(c);
    } else --cursor;
    break;
} else if (isSpecialChar(c)) { // 문자 두개가 연산자가 될 수 있는 경우
    if (state != State.Initial && state != State.Op) {
        --cursor;
        break;
    }
    state = State.Op;
} else if (Character.isAlphabetic(c)) { // 알파벳을 인식한 경우
    if (state != State.Initial && state != State.tidOrReserved) { // ID나 예약어가 아니면 break
        --cursor;
        break;
    }
    state = State.tidOrReserved;
}
tokenString += String.valueOf(c); // 글자 String으로 추가
}
symType = getSymbolType(state);
if (symType == Token.SymType.tidOrReserved && tokenString.length() > ID_LENGTH) {
    // ID의 길이가 16을 넘으면 예외처리
    System.err.print(Error.getErrorMessage(Error.ErrorCode.AboveIDLlimit));
    return token;
}
token.setSymbol(tokenString, symType);
return token;
}
}

```

➤ Token.java

```
package CScanner;
```

```
public class Token {
    public enum SymType { // 쉽게 분류하기 위한 대분류들
        Op, tidOrReserved, Digit, Real, Const, NULL
    }

    public enum Symbols { // 토큰의 심볼들
        NULL,
        NULL0, NULL1, NULL2, tident, tconst, tint, treal, NULL7, NULL8, NULL9,
        Plus, Minus, Mul, Div, Mod, Assign, Not, And, Or, Equal,
        NotEqual, Less, Great, Lesser, Greater, NULL25, NULL26, NULL27, NULL28, NULL29,
        LBracket, RBracket, LBrace, RBrace, LParen, RParen, Comma, Semicolon, SingleQuote, NULL39,
        If, While, For, Const, Int, Float, Else, Return, Void, Break,
        Continue, Char, Then, EOF
    }
}
```

```
private Symbols symbol; //토큰이 가진 심볼 변수
private String value; // 값을 저장하는 변수
private String tokenString; // 토큰의 String
```

```
// 생성자
public Token() {
    symbol = Symbols.NULL;
    value = "0";
    tokenString = "NULL";
}
```

```
// 토큰이 가진 심볼과 value를 올바르게 설정하는 메소드
public void setSymbol(String token, SymType type) {
    tokenString = token;
    switch(type) {
        case tidOrReserved:
            symbol = GETtidentOrReservedWord(token);
            if(symbol == Symbols.tident) {
                value = token;
            }
            break;
        case Digit:
            symbol = Symbols.tint;
            value = Integer.toString(parseInt(token));
            break;
        case Real:
            symbol = Symbols.treal;
            value = token;
            break;
        case Const:
            symbol = Symbols.tconst;
            value = token;
            break;
        case Op:
            symbol = getOp(token);
            break;
        case NULL:
        default:
            break;
    }
}
```

// 정수의 진법 계산 메소드

```
private int parseInt(String s) {
    int radix = 10; // default 진법은 10진수
    if (s.startsWith("0x")){ // 16진수일 경우
        radix = 16; // 진법을 16진수로 설정
        s = s.substring(2); // prefix인 0x 제거
    } else if (s.startsWith("0") && s.length() > 1){ // 8진수일 경우
        radix = 8; // 진법을 8진수로 설정
    }
    return Integer.parseInt(s, radix); // 위에서 설정한 진법대로 진법 변환
}
```

// indent나 예약어를 받았을 때 구분해주는 메소드

```
private Symbols GETtidentOrReservedWord(String token) {
    switch(token) {
        case "If": return Symbols.If;
        case "While": return Symbols.While;
        case "For": return Symbols.For;
        case "Const": return Symbols.Const;
        case "Int": return Symbols.Int;
        case "Float": return Symbols.Float;
        case "Else": return Symbols.Else;
        case "Return": return Symbols.Return;
        case "Void": return Symbols.Void;
        case "Break": return Symbols.Break;
        case "Continue": return Symbols.Continue;
        case "Char": return Symbols.Char;
        case "Then": return Symbols.Then;

        default:
            return Symbols.tident;
    }
}
```

// 트윈이 연산자인 경우 구분해주는 메소드

```
private Symbols getOp(String token) {
    switch (token) {
        case "!": return Symbols.Not;
        case "!=": return Symbols.NotEqual;
        case "%": return Symbols.Mod;
        case "&&": return Symbols.And;
        case "(": return Symbols.LParen;
        case ")": return Symbols.RParen;
        case "*": return Symbols.Mul;
        case "+": return Symbols.Plus;
        case ",": return Symbols.Comma;
        case "-": return Symbols.Minus;
        case "/": return Symbols.Div;
        case ";": return Symbols.Semicolon;
        case "<": return Symbols.Less;
        case "<=": return Symbols.Lesser;
        case "=": return Symbols.Assign;
        case "==": return Symbols.Equal;
        case ">": return Symbols.Great;
        case ">=": return Symbols.Greater;
        case "[": return Symbols.LBracket;
        case "]": return Symbols.RBracket;
        case "\\255": return Symbols.EOF;
        case "{": return Symbols.LBrace;
        case "|": return Symbols.Or;
        case "}": return Symbols.RBrace;
        case "\"": return Symbols.SingleQuote;
        case "&": // &하나만 있으면 에러
            System.err.print(Error.getErrorMessage(Error.ErrorCode.SingleAmpersand));
            break;
        case "|": // |하나만 있으면 에러
            System.err.print(Error.getErrorMessage(Error.ErrorCode.SingleBar));
            break;
        default: // 그 외 연식을 못하면 에러
            System.err.print(Error.getErrorMessage(Error.ErrorCode.InvalidChar));
            break;
    }
    return Symbols.NULL;
}
```

```
// 토큰 심볼을 숫자로 리턴해주는 메소드
public int getSymbolOrdinal() {
    return symbol.ordinal()-1; // NULL이 -1이기 때문에 -1 해야한다.
}

// 토큰이 명칭이나 숫자인 경우에 토큰의 값을 얻는 메소드
public String getSymbolValue() {
    return value;
}

// 출력 편의를 위한 toString 메소드
public String toString() {
    return tokenString + "\t : (" + getSymbolOrdinal() + ", " + getSymbolValue() + ")";
}
}
```

➤ Error.java

```
package CScanner;

public class Error {
    public enum ErrorCode {
        CannotOpenFile, AboveIDLlimit, SingleAmpersand, SingleBar, InvalidChar, InvalidComment
    }

    public static String getErrorMessage(ErrorCode code) {
        String msg;
        msg = "Error occur(code: " + code.ordinal() + ")\n";
        switch (code){
            case CannotOpenFile:
                msg += "cannot open the file. please check the file path.";
                break;
            case AboveIDLlimit:
                msg += "an identifier length must be less than 12.";
                break;
            case SingleAmpersand:
                msg += "next character must be &.";
                break;
            case SingleBar:
                msg += "next character must be |.";
                break;
            case InvalidChar:
                msg += "invalid character!!!";
                break;
            case InvalidComment:
                msg += "invalid block comment!!!";
                break;
            default:
                msg += "Unknown Error";
                break;
        }
        return msg;
    }
}
```

- **입력 소스 코드**

ProgramTest01.txt

var

Int a, b, c;

Float x, y, z;

begin

a = 10; b = 20; c = 3;

x = a + b * c;

print x;

end

- 실행 결과

SampleCode01.txt

```
<terminated> Main (1) [Java Application] C:\Java\jec  
PRINT RESULT : 70.0
```

```
[INT TABLE]
```

a	10.0
b	20.0
c	3.0
NONE	NONE
NONE	NONE

```
[FLOAT TABLE]
```

x	70.0
y	NONE
z	NONE
NONE	NONE
NONE	NONE

● 개발 후기

스캐너와 같이 작동하는 파서를 만들게 되면서 코드의 모듈화에 대한 진지한 고민을 하게 되었습니다. 제 자신의 부족함도 많이 느끼고 시간이 무한정 있었다면 어땠을까 하는 생각도 했지만, 스스로 논리구조를 만들어 코드를 컨트롤하는 느낌이 굉장히 즐거운 시간이었습니다. 중간고사때 제출한 스캐너에 파서를 추가하여 스캐너가 넘겨주는 토큰값과 정보를 받아와 제가 만든 문법에 맞춰서 문맥을 파악하는 과정은 생각만큼 까다로운 과정이었습니다. 예제 input코드는 짧고 요구사항도 많지 않아서 비교적 아귀가 맞지 않는 부분이 있어도 돌아갈 수 있지만 대체로 정확하게 돌아가며 여러가지 환경에서 작동하는 파서를 만든다는 것은 엄청난 고뇌가 필요할 것 같습니다. 이번과제를 하면서 아쉬웠던 부분은 코드를 모듈화시키고 재귀함수의 특성도 넣어서 작성했으면 훨씬 짧은 코드가 되었을 것 같아서 아쉬웠습니다.