프로그래밍 언어 개론 00분반

201402447 한원희

FunctionNode.java

```
public class FunctionNode extends Node{
    public enum FunctionType{
        ATOM_Q  { TokenType tokenType() { return TokenType.ATOM_Q;} },
        CAR     { TokenType tokenType() { return TokenType.CAR;} },
        CDR     { TokenType tokenType() { return TokenType.CDR;} },
        COND    { TokenType tokenType() { return TokenType.COND;} },
        CONS    { TokenType tokenType() { return TokenType.CONS;} },
        DEFINE  { TokenType tokenType() { return TokenType.DEFINE;} },
        EQ_Q    { TokenType tokenType() { return TokenType.EQ_Q;} },
        LAMBDA  { TokenType tokenType() { return TokenType.LAMBDA;} },
        NOT     { TokenType tokenType() { return TokenType.NOT;} },
        NULL_Q  { TokenType tokenType() { return TokenType.NULL_Q;} };


        private static Map<TokenType, FunctionType> fromTokenType = new HashMap<TokenType, FunctionType>

        static {
            for(FunctionType fType : FunctionType.values()){
                fromTokenType.put(fType.tokenType(), fType);
            }
        }

        static FunctionType getFunctionType(TokenType tType) {
            return fromTokenType.get(tType);
        }
```

BinaryOpNode.java 의 모습을 참고하여 만들었습니다. Enum의 원소들은

```
case ATOM_Q:
case CAR:
case CDR:
case COND:
case CONS:
case DEFINE:
case EQ_Q:
case LAMBDA:
case NOT:
case NULL_Q:
```

CuteParser.java에 있습니다.

```
case DIV:
case EQ:
case MINUS:
case GT:
case PLUS:
case TIMES:
case LT:
    BinaryOpNode binaryNode = new BinaryOpNode();
    if(tLexeme == null)
        System.out.println("???");
    binaryNode.setValue(tType);
    return binaryNode;
```

```
case NOT:
case NULL_Q:
    FunctionNode functionNode = new FunctionNode();
    if(tLexeme == null)
        System.out.println("???");
    functionNode.setValue(tType);
    return functionNode;
```

두개의 파트는 위의 INT 타입의 설정을 참고하여 작성했습니다.

```
case INT:
    IntNode intNode = new IntNode();
    if (tLexeme == null)
        System.out.println("???");
    intNode.value = new Integer(tLexeme);
    return intNode;
```

Value를 설정해 줄때는

```
public void setValue(TokenType tType) {
    FunctionType fType = FunctionType.getFunctionType(tType);
    value = fType;
}
```

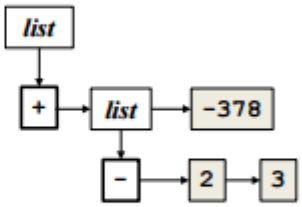
각각의 java 파일에 있는 setValue 메소드를 활용했습니다. tType은 맨 처음 switch문에 들어올 때 설정되어 있습니다.

```
case L_PAREN:
    ListNode listNode = new ListNode();
    if(tLexeme == null)
        System.out.println("???");
    listNode.value = parseExprList();
    return listNode;
```

왼쪽 괄호가 나오면 해줄 행동입니다. 먼저

다음과 같은 프로그램이 있다고 가정하면, parse tree 는 다음과 같이 된

(+ (- 2 3) -378)



그리고 프로그램의 출력 결과는 다음과 같다.

([PLUS] ([MINUS] [INT:2] [INT:3])[INT:-378])

그림을 보면 왼쪽 괄호가 시작되면 list로 취급하여 생성해주는 모습입니다.

위의 것이랑 같게 value를 설정해 주어야 하는데

```
public class ListNode extends Node{
    public Node value;

}
```

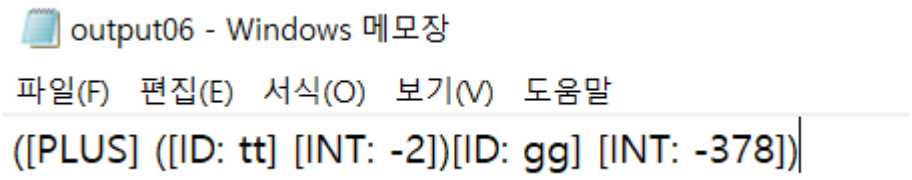
ListNode.java엔 Node 타입의 value가 선언되어 있습니다.

또한 바로 밑에

```java
// List의 value를 생성하는 메소드
private Node parseExprList() {

    Node head = parseExpr();
    // head 의 next 노드를 set 하시오.
    if (head == null) // if next token is RPAREN
        return null;
    head.setNext(parseExprList());
    return head;
}
```

이런식으로 list의 value를 생성하는 메소드가 있습니다. Node 타입을 반환해줍니다.



결과 화면입니다.