프로그래밍언어 개론 보고서

[07] List Parser

제출일	2017/07/04
학 번	201000287
소 속	일어일문학과
이 름	유다훈

1 과제문제

1 주어진 코드에서 함수 구현

- Node parserExpr()
  - 토큰을 전달받아 분류하는 작업을 하는 메소드
- Node parserExprList()
  - 전달받은 토큰을 헤드 노드에 설정하고 헤드의 다음 노드값을 설정해주는 메소드
  - Recursion 하게 작성
- void printNode(Node node)
  - 전달받은 노드가 ListNode 인지, 나머지의 다른 노드인지를 구분하여 노드의 속성타입값과 함께 출력하는 메소드
  - Recursion 하게 작성

2 과제 해결 방법

- Node parserExpr()

```
case MINUS:
    binaryOpNode = new BinarayOpNode();
    binaryOpNode.value = BinType.MINUS;
    return binaryOpNode;

case PLUS :
    binaryOpNode = new BinarayOpNode();
    binaryOpNode.value = BinType.PLUS;
    return binaryOpNode;

case TIMES:
    binaryOpNode = new BinarayOpNode();
    binaryOpNode.value = BinType.TIMES;
    return binaryOpNode;
case DIV:
    binaryOpNode = new BinarayOpNode();
    binaryOpNode.value = BinType.DIV;
    return binaryOpNode;
case LT :
    binaryOpNode = new BinarayOpNode();
    binaryOpNode.value = BinType.LT;
    return binaryOpNode;
case GT :
    binaryOpNode = new BinarayOpNode();
    binaryOpNode.value = BinType.GT;
    return binaryOpNode;
case EQ :
    binaryOpNode = new BinarayOpNode();
    binaryOpNode.value = BinType.EQ;
    return binaryOpNode;
```

```
case ATOM_Q:
    FunctionNode atom = new FunctionNode();
    atom.value = FunctionType.ATOM_Q;
    return atom;
// FunctionNode
// FunctionNode
case DEFINE:
    FunctionNode define = new FunctionNode(
    define.value = FunctionType.DEFINE;
    return define;
case LAMBDA:
    FunctionNode lambda = new FunctionNode(
    lambda.value = FunctionType.LAMBDA;
    return lambda;
case COND:
    FunctionNode cond = new FunctionNode();
    cond.value = FunctionType.COND;
    return cond;
case NOT:
    FunctionNode not = new FunctionNode();
    not.value = FunctionType.NOT;
    return not;
case CDR:
    FunctionNode cdr = new FunctionNode();
    cdr.value = FunctionType.CDR;
    return cdr;
```

- 정해진 Enum 의 토큰 타입값들에 대해서 전부 구현해준다.
- 해당 토큰타입의 노드객체를 생성하고, 해당 노드 객체의 value 값을 지정해준 다음 생성한 노드를 반환한다.

- Node parserExprList()

```
private Node parseExprList() {

    Node head = parseExpr();

    if (head == null) {
        return null;
    }
    head.setNext(parseExprList());


    return head;
}
```

- 헤드의 값을 토큰을 검사하고 리턴 되는 값으로 설정해준 다음,
- 헤드의 다음 값을 설정하기 위해서 자기 자신을 다시 호출한다.
- 이 때 닫는 괄호 ')' 일 경우에는 토큰의 리스트가 끝나는 것이므로 null 값을 리턴 해준다.
- 계속해서 자기 자신을 호출해주면서 모든 토큰을 노드에 설정하고, 리턴해줌으로써 헤드의 다음 값으로 설정해준다.

- void printNode(Node node)

```
    //recursive call or iteration...
    //void printNode
    public void printNode(Node node) {
        if(node != null){
        if(node instanceof ListNode) {
            ListNode listNode = (ListNode)node;
            ps.print("(");
            printNode(listNode.value);
            ps.print(")");

        } else {
            ps.print("[" + node.toString() + "] ");
        }
        printNode(node.getNext());
    }
    }
}
```

- 전달받은 노드들을 해당 타입과 함께 호출한다.
- 여는괄호 “)”에 모든 노드가 연결되어 있으므로, ListNode 인지를 확인하고, ListNode 라면 자기 자신을 재귀적으로 호출해준다.
- ListNode 가 아니라면 전달받은 노드 자기 자신을 출력한다.
- node 의 타입마다 toString 이 Overriding 되어 있기 때문에 타입마다 해당 타입에 맞게 적절한 값이 출력된다.

3 과제를 해결하면서 느낀 점

1 Overriding 덕분에 같은 내용의 코드를 반복해서 쓰지 않아도 간편하게 출력할 수 있었다.

2 ListNode 를 구별할 때, 닫는 괄호 “)” 가 null 로 처리해서, 이 부분이 잘 실행되지 않았다. 이해 하는데에 시간이 걸렸다.

3 메소드의 재귀적인 호출로 인하여 길어질 코드를 간편하게 할 수 있었다.