

프로그래밍언어론 Assignment 10

과제 제출일자 : 2019/06/08

조원, 팀 번호 :

18조

201502085, 이규봉 (조장)

201502094, 이재호

분반 : 01

과제 : Cute Interpreter Define 처리 (Item 2)

구현 사항에 대한 설명

```
public class DefineTable {

    private static Map<String, Node> defineTable = new HashMap<String, Node>();

    public static Node lookupTable(String id) {
        return defineTable.get(id);
    }

    public static void define(String id, Node value) {
        // if same key is in table, change the value
        defineTable.put(id, value);
    }
}
```

DefineTable 클래스는 defineTable이란 HashMap static 객체를 갖는 클래스입니다. lookupTable()를 사용해 map에 접근하고, define을 통해 맵에 새 요소를 정의합니다.

```
public boolean isDefineStatement(String str) {
    // input에 define이 존재한다면 define 구문으로 인식함.
    // 같은 id를 가진 요소를 재정의하기 위해 필요한 함수임
    if(Scanner.stream(str).filter((token)->token.type() == TokenType.DEFINE).count() > 0) {
        return true;
    }
    return false;
}
```

isDefineStatement는 CuteParser 클래스에 정의되어 있으며, CuteParser는 객체 생성 시 input 값을 생성자 인자로 받습니다. 이 input 값이 위 함수의 인자인 str에 들어가게 되며, Scanner.stream를 통해, 각 Token 형태로 분리됩니다. 그 후 filter를 통해 Define 타입의 토큰이 있는지를 검사하여, 문장에 define이 들어 있는지를 검사해, 존재한다면 true를 반환합니다.

```

case DEFINE:

    Node keyNode;
    Node valueNode;

    if(!(operand.car() instanceof IdNode)) {
        errorLog("Not defined behavior");
        return null;
    }

    keyNode = (IdNode) (operand.car());

    if((operand.cdr().car()) instanceof ListNode) {
        valueNode = (((ListNode) (operand.cdr().car())));
    }
    else {
        valueNode = operand.cdr().car();
    }

    DefineTable.define(keyNode.toString(), valueNode);
    DefineTable.Debugging();
    return null;

```

위는 CuteInterpreter 클래스의 runFunction내 Define 처리 부분입니다. keyNode가 define의 첫 번째 피연산자에, valueNode가 두 번째 피연산자에 해당하며, DefineTable.define을 통해 테이블에 키 값 쌍을 저장합니다.

```

case ID:
    if(isDefineStatement(input)) {
        return new IdNode(tLexeme);
    }
    if (DefineTable.LookupTable(tLexeme) == null) {
        return new IdNode(tLexeme);
    } else {
        return DefineTable.LookupTable(tLexeme);
    }

```

위는 CuteParser 클래스의 parseExpr 메서드 내 ID 처리 부분입니다. ID 타입의 토큰을 파싱할 때, isDefineStatement 메서드를 사용해 define 문장인지 확인하고, define 문장이라면 테이블에 키가 존재하는지 확인하지 않고 바로 IdNode를 반환합니다. 이렇게 구현한 이유는, a란 요소가 테이블에 있다면 (define a 5) 등의 문장으로 a를 재정의하려 할 때, 이 문장이 (define (IntNode) 5) 식으로 바뀌어 예외가 나는 것을 처리하기 위한 것입니다. 반면 define 문장이 아닐 땐, (+ a 3) 같은 문장에선 a란 key가 테이블에 존재하는지 확인해서 존재하지 않는 경우, a란 IdNode를 그대로 반환하고 (문장이 그대로 반환됨) key가 테이블에 존재하는 경우, 테이블에 저장해 놓은 Node를 반환하기 위한 것입니다.

```

private boolean isInBuildInFunction(String input) {
    switch (input) {
        case "clear":
            commandLogArea.setText(initialString);
            textField.setText("");
            return true;
    }

    return false;
}

public void setErrBuffer(String err) {
    errBuffer = err;
}

```

isInBuildInFunction는 빌트인 함수를 사용하기 위해 만든 함수입니다. clear를 입력하면 파싱 과정을 거치지 않고 위 함수에서 콘솔을 clear하는 작업을 진행합니다. setErrBuffer는 Console 이외의 클래스에서 에러가 발생했을 때 errBuffer에 대응하는 오류 메시지를 넣어놓고, 콘솔에 출력하기 위한 에러처리 메서드입니다.

```

private static Console INSTANCE;

private String errBuffer = "";

public static Console getInstance() {
    if(INSTANCE == null) {
        INSTANCE = new Console();
    }
    return INSTANCE;
}


```

콘솔 객체는 1개만 있으면 충분하기 때문에, 싱글톤으로 만들어 전역에서 접근 가능하게 만들었습니다. errBuffer는 에러를 출력하는데 사용하는 변수이고, 매 입력 때 마다 비워집니다.

결과 스크린샷

```
Welcome to Cute Interpreter!
$ ( define a #T )
$ a
... #T
$ ( define a ( - 3 1 ) )
$ a
... 2
$
```

a의 재정의

 Cute Interpreter

```
Welcome to Cute Interpreter!
$ ( define a ' ( 1 2 3 ) )
$ ( cdr a )
... '( 2 3 )
$ ( car a )
... 1
$
```

a를 연산

```
Welcome to Cute Interpreter!
$ ( car 1 )
... Wrong Function Use
$
```

잘못된 입력 처리

역할 분담

이규봉 :

CuteParser 작성, 중복 IdNode 버그 수정, 보고서 작성

이재호 :

GUI 관련 코드 추가 작성, DefineTable 작성, CuteInterpreter define 로직 작성