소스코드 설명

```
import org.antlr.v4.runtime.*;

public class TestMiniC {
    public static void main(String[] args) throws Exception {
        MiniCLexer lexer = new MiniCLexer( new ANTLRFileStream( fileName: "test.c"));
        CommonTokenStream tokens = new CommonTokenStream( lexer );
        MiniCParser parser = new MiniCParser( tokens );
        genAST(parser.program());
    }

    public static void genAST(RuleContext context)
    {
        System.out.println("201904243 Rule " + context.getRuleIndex());
        for (int i = 0; i < context.getChildCount(); i++)
        {
            var elem : ParseTree = context.getChild(i);
            if (elem instanceof RuleContext)
                genAST((RuleContext) elem);
        }
    }
}
```

ANTLR에서 만들어주는 MiniCParser 코드는 g4 파일에 기재된 Rule들을 RuleParserContext에 확장해서 {RuleName}Context로 class로 만드는 것을 확인했습니다.

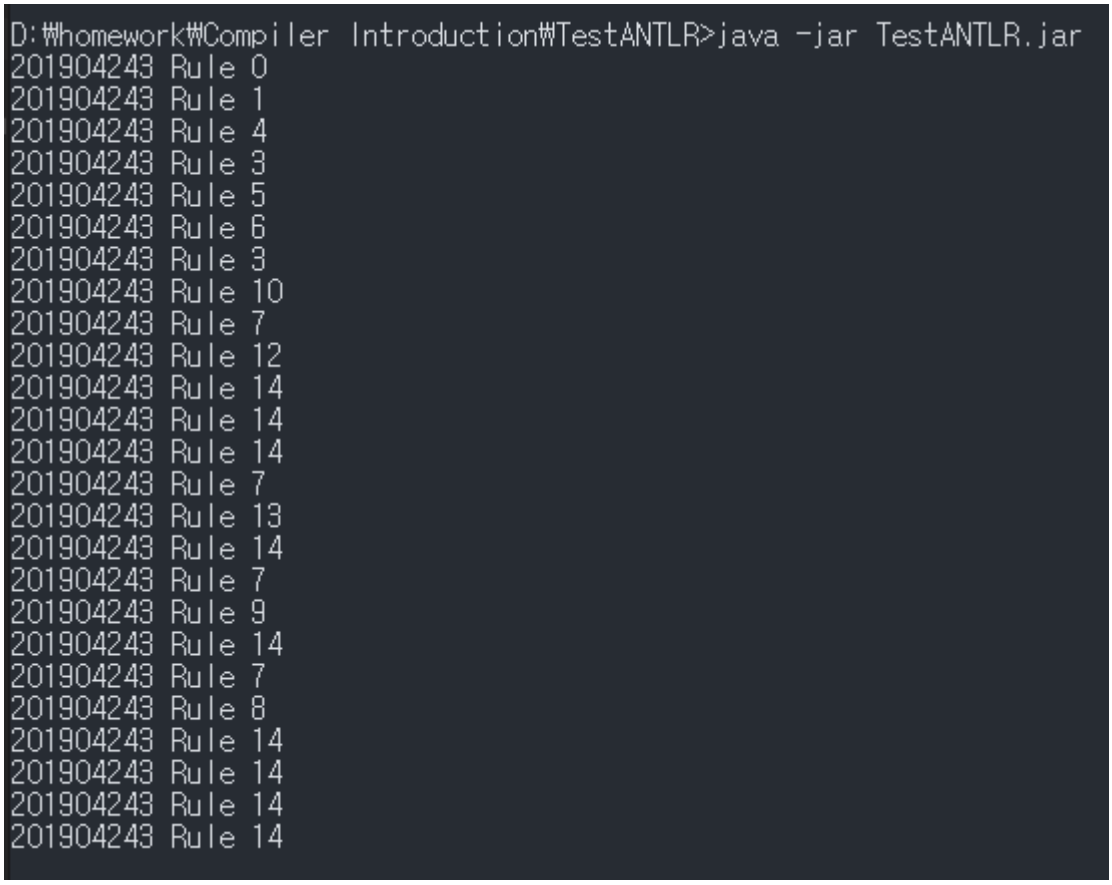
그리고 각자 멤버 변수로 자신 밑에 있는 ParseTree 하부 컴포넌트들을 children으로 가지고있는 것이 확인되서 이걸 이용해서 AST를 뽑아주는 재귀함수를 짜고 그 안에 print함수를 넣어서 RuleNumber를 출력해주도록 하였습니다.

RuleParserContext는 RuleContext를 확장한 클래스이고 RuleContext에서 GetRuleIndex 함수를 구현해 놓았기 때문에 RuleContext 클래스를 인자로 받고 그 자식들을 재귀함수로 계속 호출 해주도록 코드를 구현하였습니다.

실행 결과 화면

```
int test (int p)
{
    if (p == 1)
        return 3;
    else
        while (1) p = p + 1;
}
```

test.c의 내용



Jar 파일 실행 결과 화면

느낀 점

직접 ANTLR를 써보라는 취지의 과제의 의도는 좋았으나 조금 더 ANTLR Generating 구조라던가 문법 짜는 방법을 상세히 설명해주셨으면 좋았을거 같습니다. 이 부분을 아예 모르고 단순히 공유해주신 Setting pdf만 보고 따라했을때는 뭐가 뭔지 몰라서 ANTLR Docs를 찾아보거나 API Docs, gen된 코드들 분석을 하느라 쉬운 과제라고 말씀 해주셨음에도 불구하고 꽤나 힘들었던 것 같습니다.