

컴파일러개론 5주차 실습

UGLY PRINT USING LISTENER

2024. 10. 11.

TA: 박정필

✉: 202350941@o.cnu.ac.kr

공지, 질문 방법

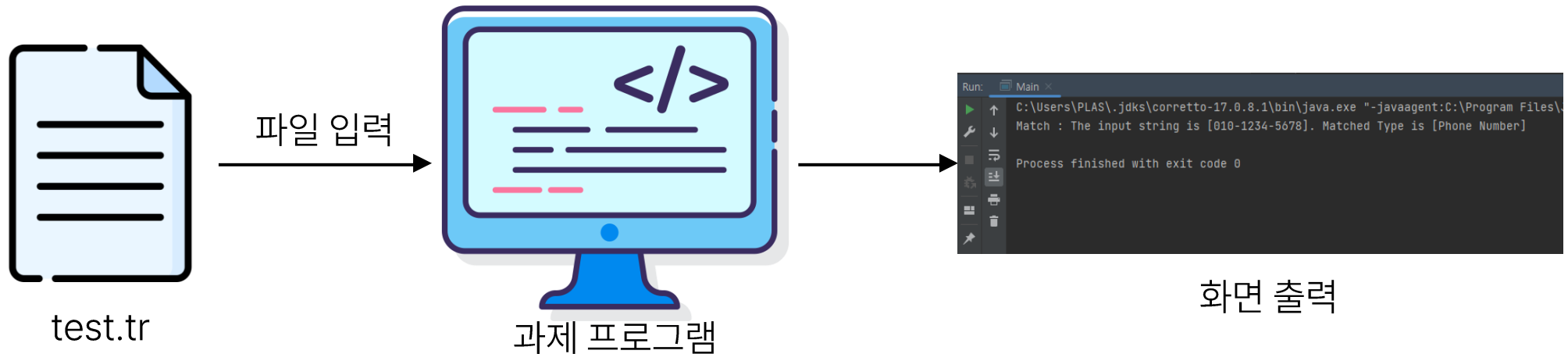
- 강의자료, 동영상
 - 사이버캠퍼스 업로드 예정
- 공지
 - 카카오톡 오픈채팅, 사이버캠퍼스
 - <https://open.kakao.com/o/gkfF7JMg>
 - 오픈프로필 사용 가능
 - 참여코드: 2024cp01
- 질문
 - 수업시간, 카카오톡 오픈채팅에 질문
 - 과제 관련 질문은 제출기한 전날까지만 가능 (당일 질문은 답변 X)
 - 이메일
 - 이메일 제목은 "[컴파일러개론][분반] ...", 제목 반드시 준수
 - 교수님 : eschough@cnu.ac.kr
 - TA : 202350941@o.cnu.ac.kr

목차

- 5주차 과제
 - tinyR4 Ugly Print using Listener

5주차 과제: tinyR4 Ugly Print

- Java, ANTLR 를 이용하여 tinyR4 문법으로 된 코드에 대한 ugly print 코드를 listener를 활용하여 작성하기
 - 구조



5주차 과제: tinyR4 Ugly Print

- Java, ANTLR 를 이용하여 tinyR4 문법으로 된 코드에 대한 ugly print 코드 작성하기
 - 1, 2주차 완료 후, 결과 예제

```
fn main() {
    let a = 3;
    let b = 4;
        c = a + b;
    d = 3 - (4 + 5);
    println!(c);
    println!("Hello World");
}

fn sum(a:u32, b:u32) -> u32 {
    return a + b;
}
```

test.tr

```
fn main () {
    let  a  = 3;
    let  b  = 4;
    c = a + b;
    d = 3 - (4 + 5);
    println!(c);
    println!("Hello World");|
}
fn sum (a:u32, b:u32) -> u32{
    return a + b;
}
```

결과 화면 출력

5주차 과제: tinyR4 Ugly Print

- Java, ANTLR 를 이용하여 tinyR4 문법으로 된 코드에 대한 ugly print 코드 작성하기
 1. fn main 함수와 사칙연산 구현
 2. 2진 연산자와 피 연산자 사이에는 빈칸 1칸
 3. 사이버 캠퍼스에 올라온 tinyR4.g4 문법 파일을 사용
 4. 그 외의 프로젝트 설정은 2주차 실습 자료 참고
 5. Listener 방식을 이용하여 코드 구성

5주차 과제: tinyR4 Ugly Print

- Java, ANTLR 를 이용하여 tinyR4 문법으로 된 코드에 대한 ugly print 코드 작성하기
 1. fn main 함수와 사칙연산 구현
 - 그 외 나머지 문법은 구현 X
 2. 2진 연산자와 피 연산자 사이에는 빈칸 1칸
 - $X + Y$;

5주차 과제: tinyR4 Ugly Print

- Java, ANTLR 를 이용하여 tinyR4 문법으로 된 코드에 대한 ugly print 코드 작성하기
 - 3. 사이버 캠퍼스에 올라온 tinyR4.g4 문법 파일을 사용
 - 5주차 과제는 fn main 함수 관련과 사칙연산 문법만 구현

<pre> 1 grammar tinyR4; 2 3 program : decl+ ; 4 5 decl : fun_decl ; 6 7 fun_decl : FUNC id '(' params ')' ret_type_spec compound_stmt ; 8 9 params : 10 param '(' ',' param)* 11 ; 12 13 param : id ':' type_spec ; 14 15 type_spec : U32 ; // how about including array or string? 16 17 ret_type_spec : 18 RARROW type_spec 19 ; 20 21 compound_stmt: '{' local_decl* stmt* '}' ; </pre>	<pre> stmt : expr_stmt compound_stmt if_stmt for_stmt return_stmt break_stmt loop_stmt ; expr_stmt : expr ';' ; expr : additive_expr relative_expr id '=' expr ; </pre>	<pre> additive_expr: left=additive_expr op=('+' '-') right=multiplicative_expr multiplicative_expr ; multiplicative_expr: left=multiplicative_expr op=('*' '/' '%') right=unary_expr unary_expr ; unary_expr: op=('-' '+' '--' '++' '!') expr factor ; factor : (literal id) '(' expr ')' id '!'? '(' args ')' ; </pre>
---	---	--

5주차 과제: tinyR4 Ugly Print

- Java, ANTLR 를 이용하여 tinyR4 문법으로 된 코드에 대한 ugly print 코드 작성하기
 - 과제 입력 파일 조건
 - Parsing 이 잘되는 코드
 - 에러가 없는 코드
 - 라인 별로 구분된 코드

```
1 fn main() {  
2     ... a + b;  
3     ... c - d;  
4     ... 10 * 20;  
5     ... 40 / 2;  
6     ... a + 3 - 2 / 5 * 10;  
7 }
```

5주차 과제: tinyR4 Ugly Print

■ Background

- 코드를 출력하기 위해 생성된 parse tree를 순회하면서 값을 출력
- 순회 방법은 Listener, Visitor 두 가지가 존재하나 그 중 **Listener** 로 구현
- 2주차 과제를 생각하면 .g4 rule을 활용하여 출력을 조작 했음

```

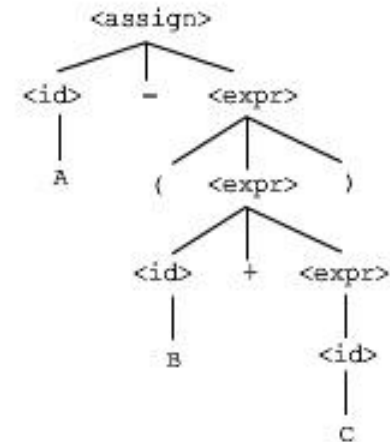
program      : decl+ ;

decl         : fun_decl ;

fun_decl     : FUNC ID '(' params ')' ret_type_spec compound_stmt ;

params       :
    | param (',' param)*
    ;
  
```

2주차 실습 예시



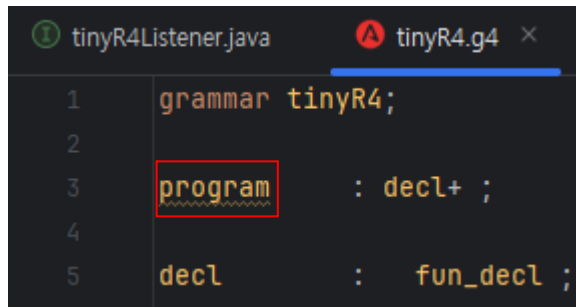
Parse Tree for the
Statement A = (B + C)

(금주) Parse tree 이용예정

5주차 과제: tinyR4 Ugly Print

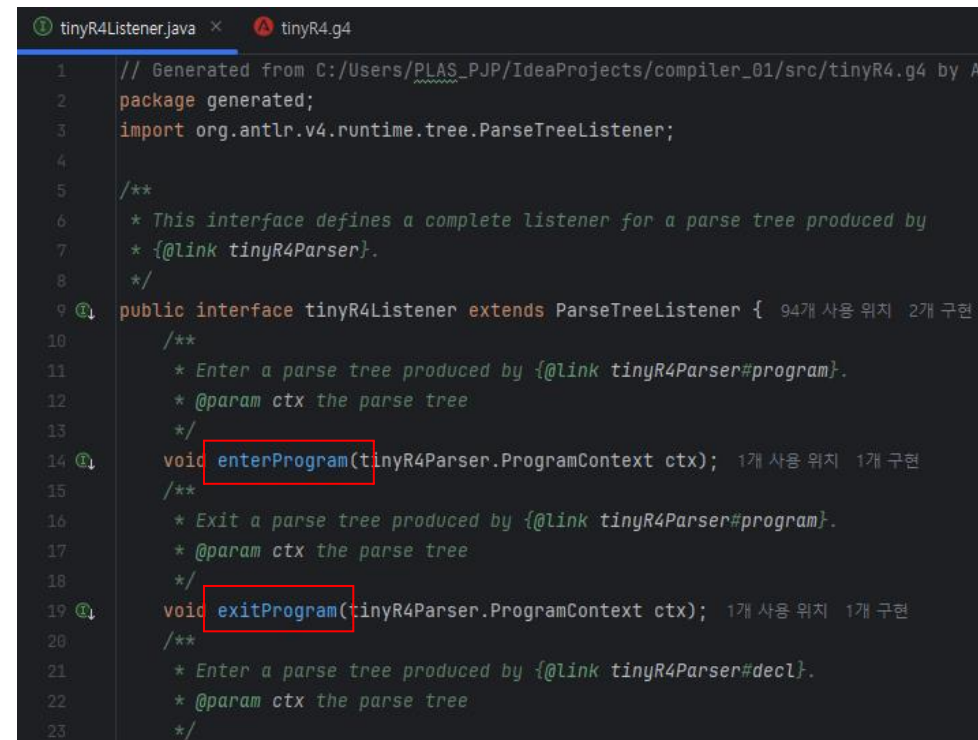
■ Background

- Listener 방식
- Non-Terminal node를 만날 경우 enter, exit 두 함수를 실행하여 노드 방문을 확인



```

1 grammar tinyR4;
2
3 program      : decl+ ;
4
5 decl        : fun_decl ;
  
```



```

1 // Generated from C:/Users/PLAS_PJP/IdeaProjects/compiler_01/src/tinyR4.g4 by ANTLR
2 package generated;
3 import org.antlr.v4.runtime.tree.ParseTreeListener;
4
5 /**
6  * This interface defines a complete listener for a parse tree produced by
7  * {@link tinyR4Parser}.
8  */
9 public interface tinyR4Listener extends ParseTreeListener {
10     /**
11      * Enter a parse tree produced by {@link tinyR4Parser#program}.
12      * @param ctx the parse tree
13      */
14     void enterProgram(tinyR4Parser.ProgramContext ctx);
15     /**
16      * Exit a parse tree produced by {@link tinyR4Parser#program}.
17      * @param ctx the parse tree
18      */
19     void exitProgram(tinyR4Parser.ProgramContext ctx);
20     /**
21      * Enter a parse tree produced by {@link tinyR4Parser#decl}.
22      * @param ctx the parse tree
23      */
  
```

5주차 과제: tinyR4 Ugly Print

■ ParserRuleContext (tinyR4PrintListener.java)

- 한 Context는 Tree의 한 노드이기도 한 점 파악하기
- 상속받은 ctx 자료구조에 대해서는 ANTLR API 문서 참고
- [ParserRuleContext \(ANTLR 4 Runtime 4.13.1 API\)](#)

OVERVIEW
PACKAGE
CLASS
USE
TREE
DEPRECATED
INDEX
HELP

ALL CLASSES
SEARCH:

SUMMARY: NESTED | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD

Package org.antlr.v4.runtime

Class ParserRuleContext

```

java.lang.Object
  org.antlr.v4.runtime.RuleContext
    org.antlr.v4.runtime.ParserRuleContext

```

All Implemented Interfaces:
ParseTree, RuleNode, SyntaxTree, Tree

Direct Known Subclasses:
InterpreterRuleContext, RuleContextWithAltNum

```

public class ParserRuleContext
  extends RuleContext

```

A rule invocation record for parsing. Contains all of the information about the current rule not stored in the RuleContext. It handles parse tree children list, Any ATN state tracing, and the default values available for rule invocations: start, stop, rule index, current alt number. Subclasses made for each rule and grammar track the parameters, return values, locals, and labels specific to that rule. These are the objects that are returned from rules. Note text is not an actual field of a rule return value; it is computed from start and stop using the input stream's toString() method. I could add a ctor to this so that we can pass in and store the input stream, but I'm not sure we want to do that. It would seem to be undefined to get the .text property anyway if the rule matches tokens from multiple input streams. I do not use getters for fields of objects that are used simply to group values such as this aggregate. The getters/setters are there to satisfy the superclass interface.

5주차 과제: tinyR4 Ugly Print

■ ParserRuleContext (tinyR4PrintListener.java)

- ctx를 통해 .g4 파일의 문법에 접근
- ParseTreeProperty 객체를 생성하여 각 문법에 접근
- ParseTreeProperty r4Tree 객체를 통한 예시 (객체 이름은 자유)
 - `r4Tree.get(ctx.fun_decl())` = ParseTree에서 `ctx.fun_decl()` 문법에 해당하는 문자열을 가져옴
 - `r4Tree.put(ctx, fun_decl)` = ParseTree에 `ctx.fun_decl()` 문법에 해당하는 문자열 `fun_decl`를 저장함

```
ParseTreeProperty<String> r4Tree = new ParseTreeProperty<>();  
@Override public void exitDecl(tinyR4Parser.DeclContext ctx) {  
    String fun_decl = r4Tree.get(ctx.fun_decl());  
    r4Tree.put(ctx, fun_decl);  
}
```

5주차 과제: tinyR4 Ugly Print

- 프로젝트 설정 추가
 - visitor 관련 코드 삭제(선택)
 - tinyR4PrintListener.java 작성
 - main 코드 작성

5주차 과제: tinyR4 Ugly Print

- visitor 관련 코드 삭제 (선택)
 - ANTLR는 Listener, Visitor에 관련된 모든 코드를 생성
 - 이 중 Visitor 와 관련된 코드 제거 (선택)
 - Visitor 삭제는 개인의 선택입니다.
강요하지 않습니다.

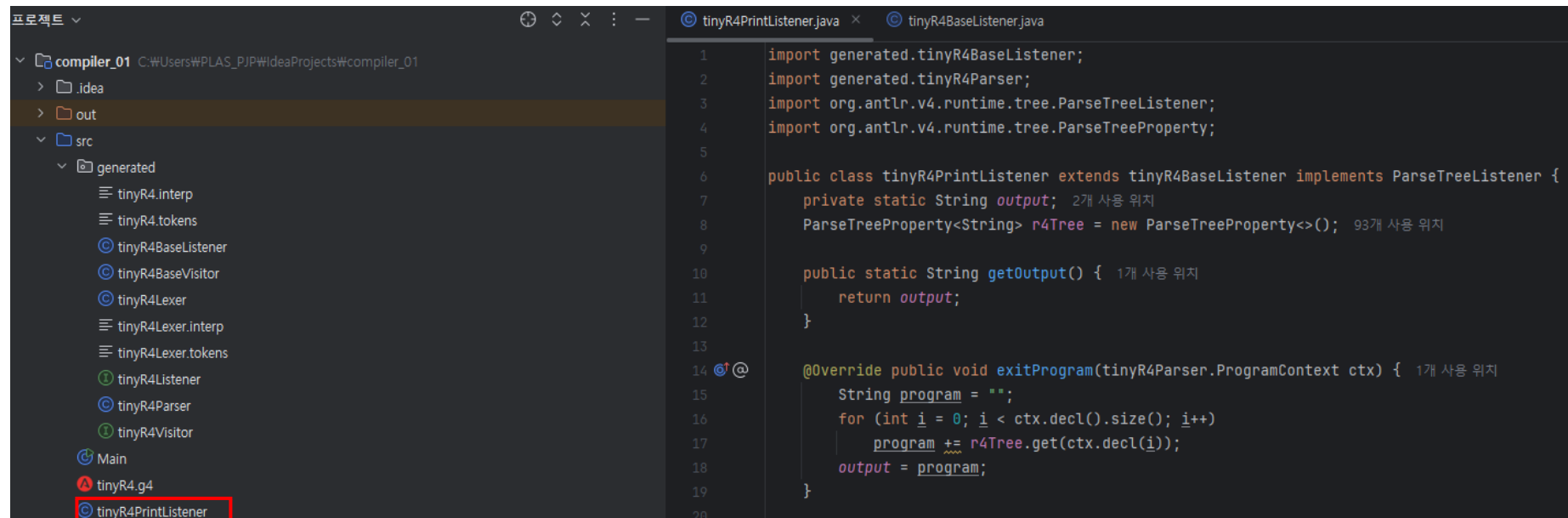
```

13 public class tinyR4Parser extends Parser {
103
104     @Override no usages
105     public String[] getRuleNames() { return ruleNames; }
106
107     @Override no usages
108     public String getSerializedATN() { return _serializedATN; }
109
110     @Override no usages
111     public ATN getATN() { return _ATN; }
112
113     public tinyR4Parser(TokenStream input) { 2 usages
114         super(input);
115         _interp = new ParserATNSimulator( parser, this, _ATN, _decisionToDFA, _sharedContextCache);
116     }
117
118     @SuppressWarnings("CheckReturnValue") 10 usages
119     public static class ProgramContext extends ParserRuleContext {
120         public List<DeclContext> decl() { 1 usage
121             return getRuleContexts(DeclContext.class);
122         }
123         public DeclContext decl(int i) { 1 usage
124             return getRuleContext(DeclContext.class,i);
125         }
126         public ProgramContext(ParserRuleContext parent, int invokingState) { 1 usage
127             super(parent, invokingState);
128         }
129         @Override public int getRuleIndex() { return RULE_program; } no usages
130         @Override no usages
131         public void enterRule(ParseTreeListener listener) {
132             if ( listener instanceof tinyR4Listener ) ((tinyR4Listener)listener).enterProgram( cbc, this);
133         }
134         @Override no usages
135         public void exitRule(ParseTreeListener listener) {
136             if ( listener instanceof tinyR4Listener ) ((tinyR4Listener)listener).exitProgram( cbc, this);
137         }
138     }
  
```

5주차 과제: tinyR4 Ugly Print

■ tinyR4PrintListener.java 생성

- tinyR4BaseListener를 상속 받는 하위 클래스로 생성
- tinyR4BaseListener는 tinyR4Listener 인터페이스(스켈레톤) 코드를 상속 받음
- 위의 두 코드를 참고 하여 각 함수들을 상속받아서 구현



The screenshot shows an IDE with two tabs: `tinyR4PrintListener.java` and `tinyR4BaseListener.java`. The left sidebar shows the project structure for `compiler_01`, with the `generated` folder expanded, listing various files including `tinyR4PrintListener`, which is highlighted with a red box. The main editor displays the code for `tinyR4PrintListener.java`:

```

1  import generated.tinyR4BaseListener;
2  import generated.tinyR4Parser;
3  import org.antlr.v4.runtime.tree.ParseTreeListener;
4  import org.antlr.v4.runtime.tree.ParseTreeProperty;
5
6  public class tinyR4PrintListener extends tinyR4BaseListener implements ParseTreeListener {
7      private static String output; 2개 사용 위치
8      ParseTreeProperty<String> r4Tree = new ParseTreeProperty<>(); 93개 사용 위치
9
10     public static String getOutput() { 1개 사용 위치
11         return output;
12     }
13
14     @Override public void exitProgram(tinyR4Parser.ProgramContext ctx) { 1개 사용 위치
15         String program = "";
16         for (int i = 0; i < ctx.decl().size(); i++)
17             program += r4Tree.get(ctx.decl(i));
18         output = program;
19     }
20

```


5주차 과제: tinyR4 Ugly Print

■ main 코드 작성

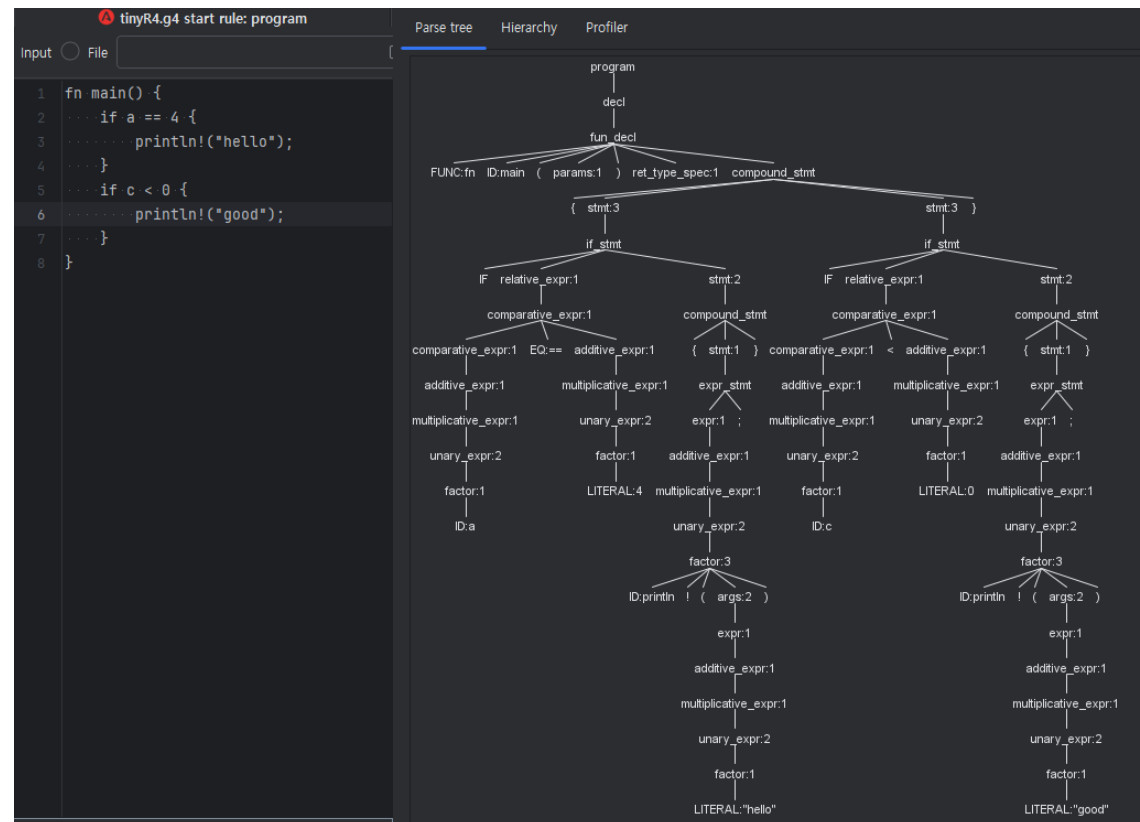
- Walker: 생성된 parse tree를 depth-first로 순회하면서 Listener 동작 수행

```
Main.java x
1  import org.antlr.v4.runtime.*;
2  import org.antlr.v4.runtime.tree.ParseTreeWalker;
3  import org.antlr.v4.runtime.tree.ParseTree;
4  import generated.*;
5
6  public class Main {
7      public static void main(String[] args) throws Exception {
8          CharStream code = CharStreams.fromFileName("./src/rust.tr");
9          tinyR4Lexer lexer = new tinyR4Lexer(code);
10         CommonTokenStream tokens = new CommonTokenStream(lexer);
11
12         try {
13             tinyR4Parser parser = new tinyR4Parser(tokens);
14             ParseTree tree = parser.program();
15
16             ParseTreeWalker walker = new ParseTreeWalker();
17             walker.walk(new tinyR4PrintListener(), tree);
18
19             System.out.println(tinyR4PrintListener.getOutput());
20         } catch (RuntimeException e) {
21             System.out.println("Error");
22         }
23     }
24 }
```

5주차 과제: tinyR4 Ugly Print

■ main 코드 작성

- Walker: 생성된 parse tree를 depth-first로 순회하면서 Listener 동작 수행



5주차 과제: tinyR4 Ugly Print

■ 과제 세부사항

- 이전 실습 과제들 동일한 방식으로,
 - 사이버캠퍼스 5주차 과제 란에 제출
 - 여러 개 파일로 구현 가능, 하지만 package 사용 금지
 - main 함수 있는 파일 이름은 **Main.java**
 - **.java** 파일들 **학번.zip**으로 압축해서 제출
- 코드에 주석 잘 달기 (필수 X)
- **입력 파일은 test.tr**
- **Visitor로 구현하면 0점**
- **과제 제출 프로젝트의 트리 구조는 다음 페이지 참고 (src 폴더 압축)**
- **javac로 컴파일 안될 경우 감점 예정**

■ 마감 (기간 준수)

- **2024년 10월 18일 금요일 23시 59분 (기한 엄수)**
- 추가 제출 기한 없음

5주차 과제: tinyR4 Ugly Print

■ 과제 구현 참고

- 총 2번을 거쳐 과제가 진행되며, 1번째 Ugly Print 과제는 main 함수와 사칙연산만을 구현, 2번째 Pretty Print 과제는 나머지 문법을 구현
- Visitor 삭제(Slide 15)는 개인의 선택입니다. 강요하지 않습니다.
- g4 문법 규칙의 각 부분의 의미와 해당 부분에서 실제로 어떤 토큰들이 매치되어 어떤 트리가 그려지는지 파악한다면 쉽게 구현 가능

5주차 과제: tinyR4 Ugly Print

■ 과제 구현 참고

예제 1

```
1 fn main() {
2     a + b;
3     c - d;
4     10 * 20;
5     40 / 2;
6     a + 3 - 2 / 5 * 10;
7 }
```



```
Main x
C:\Tools\java\jdk-22.0.2\bin\j
fn main () {
a + b;
c - d;
10 * 20;
40 / 2;
a + 3 - 2 / 5 * 10;
}
```

예제 2

```
1 fn main() {
2     x + y + x + y - x - y;
3     a + b - c * d / 2 + 2024 + 99;
4     x - y + y - y + y * 2 / 10;
5     (x - 1);
6 }
```



```
Main x
C:\Tools\java\jdk-22.0.2\bin\java.exe "-j
fn main () {
x + y + x + y - x - y;
a + b - c * d / 2 + 2024 + 99;
x - y + y - y + y * 2 / 10;
(x - 1);
}
```

5주차 과제: tinyR4 Ugly Print

■ 과제 디렉토리 구조

- tinyR4.g4 generated한 파일
 - 4개의 필수 파일
 - tinyR4BaseListener, tinyR4Lexer, tinyR4Listener, tinyR4Parser
- Main.java
- tinyR4.g4
- tinyR4PrintListener

