

## Ejercicio Java8 Caso Ca11 sintaxis

Preguntas: conteste en Java8. Al final se le pide hacerlo en estilo FP-DRY, donde sea posible

Requisitos:

[El curso 2017 >](#)

**Traducción**

FP\_Intro\_Compilation.rar (1668k)

[Videos Kokoslan](#)

[El curso 2017 >](#)

**Tools**

antlr-4.7-complete.jar (1998k)

Carlos Loria-Saenz, 26 jun. 2017 9:47

antlr4\_bats.rar (1302k)

Carlos Loria-Saenz, 16 oct. 2017 21:56

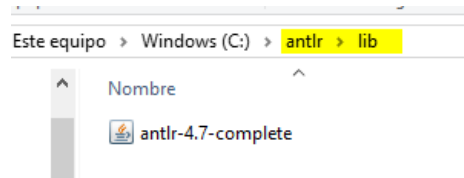
Disponga del demo de Kokoslan.

bats  
build  
cases  
classes  
grammar  
hello  
lib  
output  
src

Se quiere desarrollar el manejo sintáctico de un `call` (pero por ahora sin considerar aún su evaluación). Haremos este ejercicio desde una consola usando los `bats` que se

le entregaron junto con el demo. Trabajaremos con ANTLR 4.7. Asumimos el path apunta al lugar adecuado. En este caso como se muestra a continuación:

```
PP:where antlr4.bat
C:\antlr\bats\antlr4.bat
PP:
```



```
PP:echo %classpath%
.;classes;C:\antlr\lib\antlr-4.7-complete.jar
PP:
```

```
PP:antlr4
ANTLR Parser Generator Version 4.7
-o ____ specify output directory where all output is generated
-lib ____ specify location of grammars, tokens files
-atn ____ generate rule augmented transition network diagrams
-encoding ____ specify grammar file encoding; e.g., euc-jp
-message-format ____ specify output style for messages in antlr, gnu, vs2005
-long-messages show exception details when available for errors and warnings
-listener generate parse tree listener (default)
-no-listener don't generate parse tree listener
```

- 1) Cree un caso de prueba así y guarde en `cases\call.kl`

```
/*
    Second Demo Test: Call syntax
*/
let x = 666
f(x)
```

Al final del paso 9) de este ejercicio lo siguiente será la salida. Note que genera una excepción por que **no evalúa call todavía** sólo lo parsea y genera la salida en el directorio output

```

PP:bats\build_parser.bat
PP:bats\build_compiler.bat
error: source file or directory not found: src/kotlin/*.kt

PP:bats\test_project.bat call.kl
Prueba el caso de prueba: cases\call.kl output\call.kl.out.kl

>>> KoKoc v0.0 CR EIF400.II-2017 <<<

>>> KoKoc Reading from cases\call.kl <<<
visiting visitMult_expr (STILL ICOMPLETE!!)
visiting visitMult_expr (STILL ICOMPLETE!!)
visiting visitMult_expr (STILL ICOMPLETE!!)
>>> KoKoc is writing to output\call.kl.out.kl <<<
>>> KoKoc starts evaluating to console <<<
Exception in thread "main" kokoslan.ast.KoKoEvalException: KoKoCall: eval not implemented
    at kokoslan.ast.KoKoCall.eval(KoKoCall.java:30)
    at kokoslan.ast.KoKoProgram.eval(KoKoProgram.java:25)
    at kokoslan.ast.KoKoProgram.eval(KoKoProgram.java:30)
    at kokoslan.compile.KoKoCompiler.eval(KoKoCompiler.java:51)
    at kokoslan.compile.KoKoc.main(KoKoc.java:71)
"*** Test failed ***"
PP:

```

Salida esperada en output\call.kl.out.kl

```

let x = 666.0
f (x )

```

- 1) Modifique la gramática KoKoslan.g4 y verifique con build\_parser:

```

expression : part_expr (',' part_expr)*
;

// Value Expressions
value_expr : '(' expression ')' #ParentValueExpr
           | value_expr call_args #CallValueExpr
           | atomic_value         #AtomicValueExpr
           | list_value           #ListValueExpr
           | case_value           #CaseValueExpr
;

call_args : '(' list_expr? ')'
;

```

- 2) Actualice KoKoc.java para que use ANTLR4 v4.7 sin dar mensaje de deprecación de ANTLRInputStream.

```

// Setup Lexer/Parser
//ANTLRInputStream input = new ANTLRInputStream(is);
CharStream input = CharStreams.fromStream(is);

```

- 3) Agregue src\java\eval\KoKoListValue.java

```
KoKoListValue.java
1  /**
2   * @author loriacarlos@gmail.com
3   */
4  package kokoslan.ast;
5  import java.util.*;
6  import java.io.*;
7
8
9  public class KoKoListValue extends ArrayList<KoKoValue> implements KoKoValue{
10     public KoKoListValue(List<KoKoValue> list){
11         super(list);
12     }
13     public KoKoListValue(){
14         super();
15     }
16
17
18 }
19
```

4) Agregue src\java\ast\KokoList

```
KoKoList.java
5  package kokoslan.ast;
6  import java.util.*;
7  import java.io.*;
8
9  public class KoKoList extends ArrayList<KoKoAst> implements KoKoAst{
10     public KoKoList(List<KoKoAst> list){
11         super(list);
12     }
13     public KoKoList(){
14         super();
15     }
16     public void genCode(PrintStream out){
17         if (this.size() == 0 ) return;
18         this.get(0).genCode(out);
19         this.stream()
20             .skip(1)
21             .forEach( t -> {
22                 out.print(", ");
23                 t.genCode(out);
24             });
25     }
26     public KoKoValue eval(KoKoContext ctx){
27         KoKoListValue res = new KoKoListValue();
28         for(int i = 0; i < this.size() ; i++){
29             res.add(this.get(i).eval(ctx));
30         }
31         return res;
32     }
33     public KoKoValue eval(){
34         return eval(new KoKoContext());
35     }
36 }
```

5) Agregue src\java\ast\CallAst.java

```
package kokoslan.ast;
import java.util.*;
import java.io.*;

public class KoKoCall implements KoKoAst{
    protected KoKoList args;
    protected KoKoAst head;

    public KoKoCall(KoKoAst head, KoKoList args){
        this.head = head;
        this.args = args;
    }

    public KoKoCall(KoKoAst head){
        this(head, new KoKoList());
    }

    public void genCode(PrintStream out){
        this.head.genCode(out);
        out.print("(");
        this.args.genCode(out);
        out.print(")");
    }

    public KoKoValue eval(KoKoContext ctx){
        throw new KoKoEvalException("KoKoCall: eval not implemented");
    }
}
```

6) Agregue a src\java\compiler\Emitter.java

```
default KoKoList LIST(List<KoKoAst> expressions){
    return new KoKoList(expressions);
}

default KoKoList LIST(){
    return new KoKoList();
}

default KoKoAst CALL(KoKoAst head, KoKoList args){
    return new KoKoCall(head, args);
}
```

7) Agregue a src\java\compiler\Compiler.java

```
@Override public KoKoAst visitMult_expr(KoKoslanParser.Mult_exprContext ctx) {
    System.err.format("Visiting %s (STILL ICOMPLETE!!) %n", "visitMult_expr");
    return visit(ctx.value_expr(0));
}

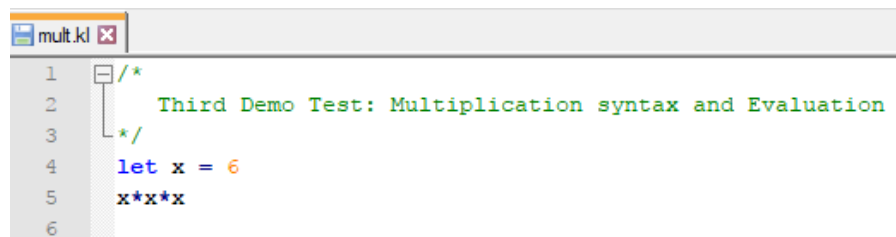
@Override
public KoKoAst visitCallValueExpr(KoKoslanParser.CallValueExprContext ctx) {
    KoKoAst head = visit(ctx.value_expr());
    KoKoList args = (KoKoList)visit(ctx.call_args());
    return CALL(head, args);
}

@Override public KoKoAst visitCall_args(KoKoslanParser.Call_argsContext ctx) {
    if ( ctx.list_expr() != null )
        return visit( ctx.list_expr() );
    else return LIST();
}

@Override public KoKoAst visitList_expr(KoKoslanParser.List_exprContext ctx) {
    List<KoKoAst> exprs = ctx.expression()
                        .stream()
                        .map( e -> visit(e) )
                        .collect(Collectors.toList());

    return LIST(exprs);
}
```

- 8) Obtenga el resultado esperado arriba
- 9) Elimine en el código incorporado patrones imperativos reemplazándolos con FP y estilo DRY.
- 10) Ahora logre este caso totalmente nuevo `cases\mult.kl`:



```
1  /*
2      Third Demo Test: Multiplication syntax and Evaluation
3  */
4  let x = 6
5  x*x*x
6
```