SUNG KYUN KWAN UNIVERSITY

ARCS

Kotlin + ANTLR

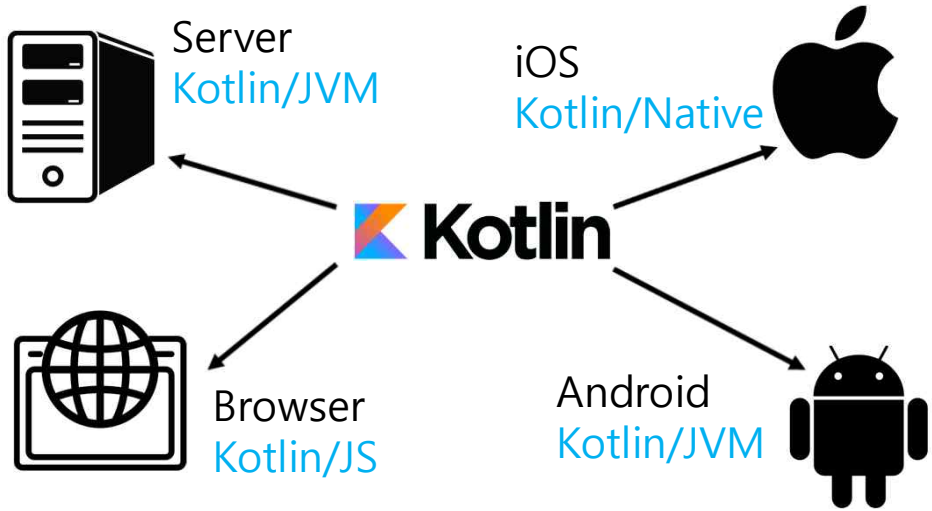
PA Overview

Overview: Programming Assignments

- PA #2
  - ANTLR based mini-Kotlin
  - implement Kotlin.g4
  - parsing basic syntax of Kotlin
    - https://kotlinlang.org/docs/reference/basic-syntax.html
- PA #3
  - Kotlin-to-Java (Source code-to-Source code) compiler
  - Type Inference

Kotlin

- General-purpose language
- OOP + FP
- Static typing
  - However, do not need type keywords



Kotlin Basic Syntax

- Defining functions

Kotlin	Java
1. Function with *return*	

`fun sum(a: Int, b: Int): Int {`
`  return a + b`
`}`
	`int sum(int a, int b) {`
	`  return a + b;`
	`}`

2. Function with an expression and inferred return type	
`fun sum(a: Int, b: Int): = a + b`	

Kotlin Basic Syntax

- Defining variables

Kotlin	Java
val: Read-only local variables	final variables

`val a: Int = 1`
`val b = 2`
`val c: Int`
`c = 3`
	`final int a = 1;`
	`final int b = 2;`
	`final int c;`
	`c = 3;`

var: Reassign-available variables	
`var x = 5` `x += 1`	`int x = 5;` `x += 1;`

Kotlin Basic Syntax

- String templates

Kotlin

```
fun main() {
    var a = 1
    val s1 = "a is $a"

    a = 2
    val s2 = "${s1.replace("is", "was")},
                                but now is $a"
    println(s2)
}
```

Java

```
class Main{
    public static void main(String[] args) {
        int a = 1;
        String s1 = "a is " + a;

        a = 2;
        String s2 = s1.replace("is", "was") +
                                ", but now is " + a;
        System.out.println(s2);
    }
}
```

Result

a was 1, but now is 2

Kotlin Basic Syntax

- Nullable values

Kotlin

```
fun StringLength(obj: Any): Int? {
    if (obj is String)
        return obj.length
    return null
}
fun main(){
    println(StringLength("String"))
    println(StringLength(123))
}
```

Java

```
class Main{
  static Integer StringLength(Object obj){
    if (obj instanceof String)
      return ((String) obj).length();
    return null;
  }
  public static void main(String[] args) {
    System.out.println(StringLength("String"));
    System.out.println(StringLength(123));
  }
}
```

Result

```
6
null
```

Kotlin Basic Syntax

- Nested functions(methods)

Kotlin

```
fun main(){
    fun StringLength(obj: Any): Int? {
        if (obj is String)
            return obj.length
        return null
    }
    println(StringLength("String"))
    println(StringLength(123))
}
```

Result

```
6
null
```

Java

```
class Main{
  public static void main(String[] args) {
    class Inner{
      Integer StringLength(Object obj){
        if (obj instanceof String)
          return ((String) obj).length();
        return null;
      }
    }
    System.out.println(new Inner().
                        StringLength("String"));
    System.out.println(new Inner().
                        StringLength(123));
  }
}
```

Kotlin Basic Syntax

- Iterating over a range

Kotlin

```
fun main(){
    for (x in 1..5) {
        print(x)
    }
}
```

Result
12345

```
fun main(){
    for (x in 1..10 step 2) {
        print(x)
    }
    println()
    for (x in 9 down 0 step 3) {
        print(x)
    }
}
```

Result
13579
9630

Kotlin Basic Syntax

- Using collections

Kotlin

```
fun main(){
    val items = listOf("apple", "banana",
                                    "kiwifruit")
    for (item in items) {
        println(item)
    }
}
```

Result

apple
banana
kiwifruit

```
fun main(){
    val items = setOf("apple", "banana",
                                    "kiwifruit")
    when {
        "orange" in items -> println("juicy")
        "apple" in items -> println("apple is
                                    fine too")
    }
}
```

Result

apple is fine too

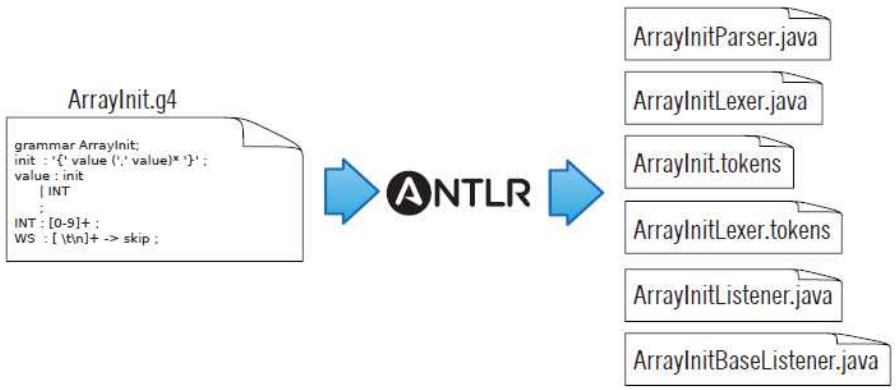
ANTLR G4 Grammar

```
/* ArrayInit.g4 */
grammar ArrayInit;

init  : '{' value ( ',' value)*  '}'  ;
value  :  init | INT  ;

INT  :   [0-9]+  ;
WS  :   [ \t\r\n]+  ->  skip  ;
```

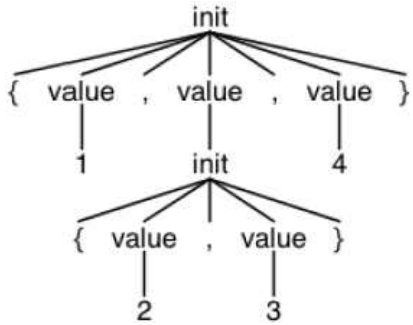
Grammar name(header, filename)

Parser rules start with lowercase letters

Lexer rules start with uppercase letters

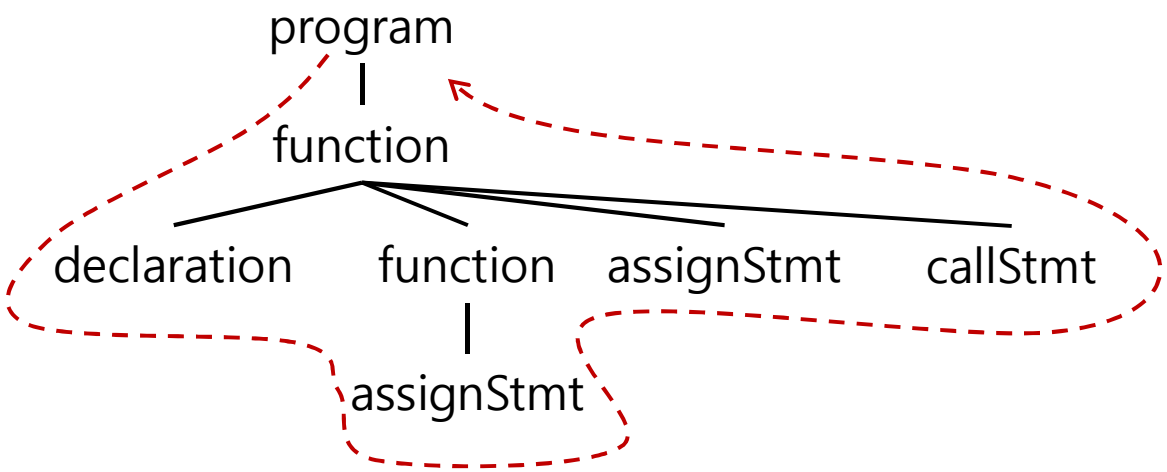


input: {1,{2,3},4}



ANTLR Listener

```
fun main( ) {
    var g: Int
    fun A(a: Int) {
        g = 3 + a
    }
    g = 1
    A(3)
}
// g = 6
```

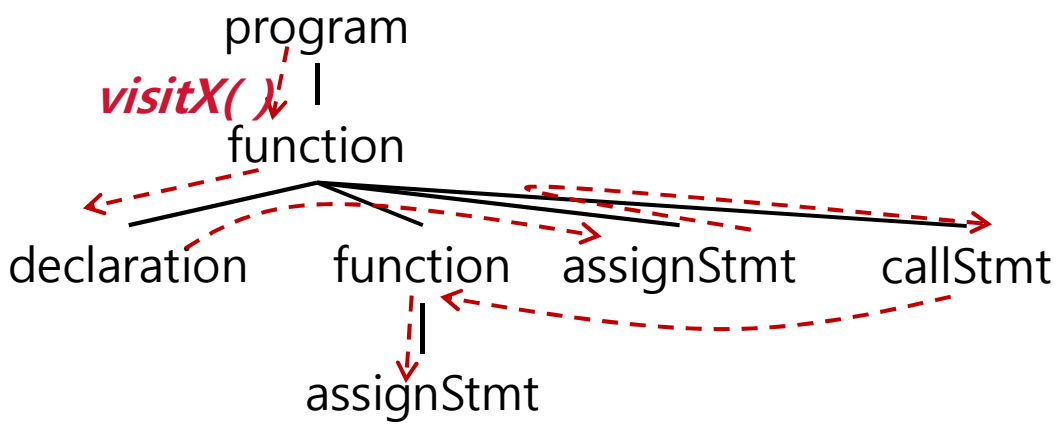


- only DFS
- *g* = 1?

```
enterProgram
  enterFunction
    enterDeclaration      var g: Int
    exitDeclaration       var g: Int
    enterFunction
      enterAssignStmt     g = 3 + a
      exitAssignStmt      g = 3 + a
    exitFunction
    enterAssignStmt       g = 1
    exitAssignStmt        g = 1
    enterCallStmt
    exitCallStmt
  exitFunction
exitProgram
```

ANTLR Visitor

```
fun main( ) {
      var g: Int
      fun A(a: Int) {
            g = 3 + a
      }
      g = 1
      A(3)
}
// g = 6
```




```
visitProgram
   visitFunction
      visitDeclaration          var g: Int
      visitAssignStmt            g = 1
      visitCallStmt
      visitFunction
         visitAssignStmt        g = 3 + a
```

- Any node can be visited
- $g = 6$

Reference

- Kotlin Basic Syntax
  - https://kotlinlang.org/docs/reference/basic-syntax.html

- ANTLR
  - https://www.antlr.org/
  - The Definitive ANTLR 4 Reference – Terence Parr