Kotlin2Java

PA #3

Programming Assignment #3

- Using Kotlin.g4 in PA #2
  - Kotlin.g4 for parsing basic syntax of Kotlin
    - https://kotlinlang.org/docs/reference/basic-syntax.html

- ANTLR based translator (visitor pattern based)
  - Kotlin-to-Java (Source code-to-Source code) compiler
  - Type inference for implicitly defined variables
    - Build symbol table while traversing AST
    - Use information in symbol table for type inference

Kotlin Basic Syntax

- Functions with return type inferrence

Kotlin

1. Function with *return*

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

2. Function with an expression and inferred return type

```
fun sum(a: Int, b: Int): = a + b
```

Java

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

Kotlin Basic Syntax

- Types of variables inferred by *rhs* expressions

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 (X - skip)

- String templates handled in a parser, now translate

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 – class Integer vs. int

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) – local classes in Java

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 (X - skip)

- *when* statement in Kotlin – *switch* statement in Java

Kotlin

```kotlin
fun feeling(day: String): String {
    when (day) {
        "Mon"   -> return "sad"
        "Sat"    -> return "happy"
        else      -> return "bad"
    }
}
fun main() {
    println("Wednesday is ${feeling("Wed")}")
    println("Saturday is ${feeling("Sat")}")
}
```

Java

```java
class Main{
    public static String feeling(String day) {
        switch(day) {
            case "Mon":   return "sad";
            case "Sat":     return "happy";
            default:         return "bad";
    } }
    public static void main(String args[]) {
        System.out.println("Wednesday is “
                                        +feeling("Wed"));
        System.out.println("Saturday is “
                                          +feeling("Sat"));
    }
}
```

Result

```
Wednesday is bad
Saturday is happy
```

Kotlin Basic Syntax

- Iterating over a range – ('..' | 'downTo') in Kotlin

Kotlin

```
fun main(){
    for (x in 1..5) {
        print(x)
    }
    println()
    for (x in 9 downTo 0 step 3) {
        print(x)
    }
}
```

Java

```
class Main{
    public static void main(String args[]) {
        for (int x = 1; x <= 5; x++) {
            System.out.print(x);
        }
        System.out.println();
        for (int x = 9; x >= 0; x=x-3) {
            System.out.print(x);
        }
    }
}
```

Result

```
12345
9630
```

Kotlin Basic Syntax

- Iterating over a range – *var : collection* in Java

Kotlin

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

Java

```
import java.util.*;
class Main{
    public static void main(String args[]) {
        List<String> items = List.of("apple",
                                  "banana", "kiwifruit");
        for (String item : items) {
            System.out.println(item);
        }
    }
}
```

Result

```
apple
banana
kiwifruit
```

Kotlin Basic Syntax

- Using collections

Kotlin

```
fun list(){
    val fruits = listOf("banana", "avocado",
                                    "apple", "kiwifruit")
    fruits.filter{it.startsWith("a")}.sortedBy{it}
            .map{it.toUpperCase()}.forEach{println(it)}
}

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

Java

```
import java.util.*;
class Main{
    public static void list() {
    List<String> fruits = List.of("banana“, "avocado",
                                                  "apple", "kiwifruit");
    fruits.stream().filter(it -> it.startsWith("a"))
            .sorted().map(it -> it.toUpperCase())
            .forEach(it -> System.out.println(it));
    }
    public static void main(String args[]) {
        Set<String> items = Set.of("apple", "banana",
                                                    "kiwifruit");
        for (String item: items) {
            System.out.println(item);
        }
        list();
    }
}
```

Result

```
apple
banana
kiwifruit
APPLE
AVOCADO
```

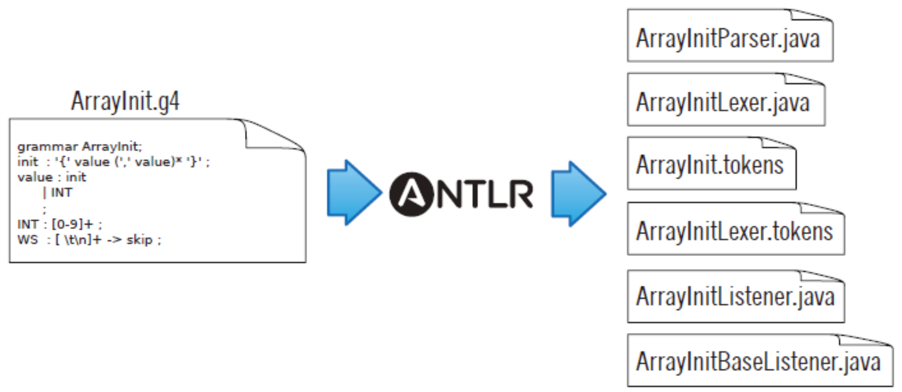
ANTLR G4 Grammar

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

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

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

Grammar name(header, filename)

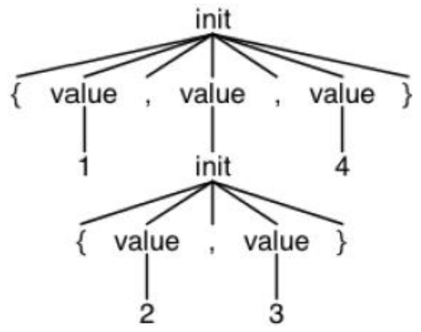
Parser rules start with lowercase letters

Lexer rules start with uppercase letters



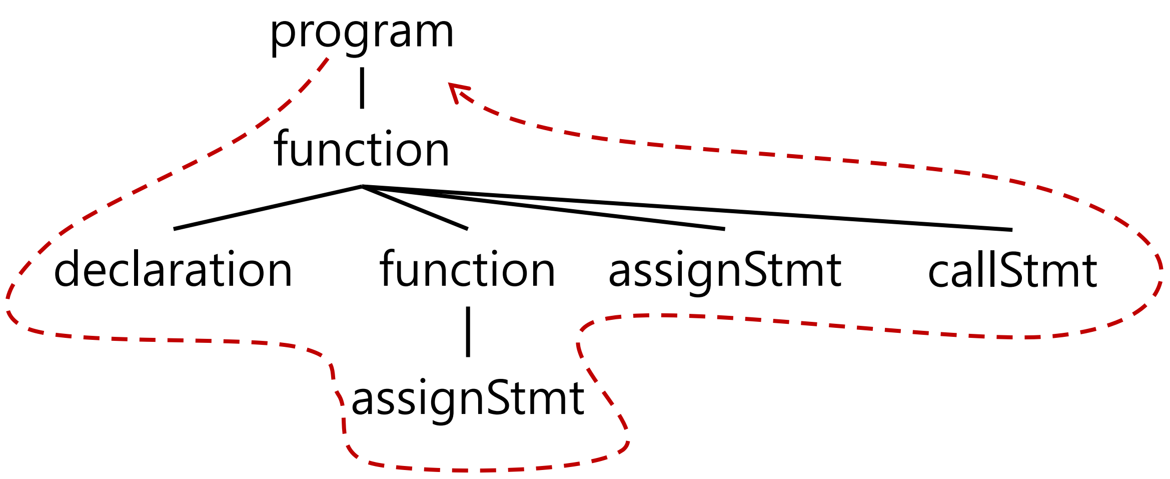
```
grammar ArrayInit;
init  : '{' value (',' value)* '}' ;
value : init
      | INT
      ;
INT : [0-9]+ ;
WS  : [ \t\n]+ -> skip ;
```

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
```

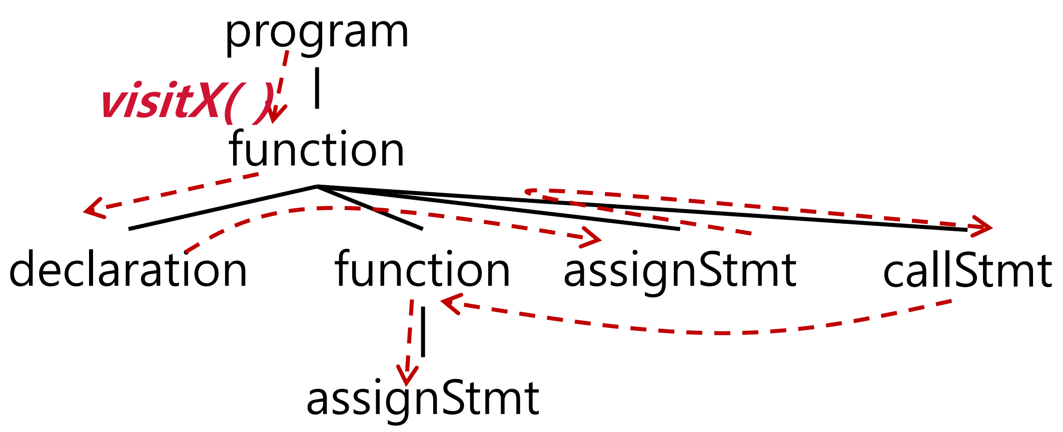


- only DFS

```
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

Programming Assignment #3 (Kotlin2Java)

- Develop Kotlin2Java.cpp with Kotlin.g4
  - Use **visitor pattern** of ANTLR for PA#3
  - Accept input and optionally output (if not specified, *input.java* is default output name) from *file-path* at command line

```
$ ./Kotlin2Java input.kt [output.java]
```

  - *output.java* should result in the same behavior as *input.kt*

Install Kotlin Compiler to Run APP

- Install zip/unzip

```
$ sudo apt update
$ sudo apt install zip unzip
```

- Install Kotlin
  - https://kotlinlang.org/docs/tutorials/command-line.html

```
$ curl –s https://get.sdkman.io | bash
$ source ~/.sdkman/bin/sdkman-init.sh
$ sdk install kotlin
```

- Run Kotlin app

```
$ kotlinc hello.kt –include-runtime –d helloKT.jar
$ java –jar helloKT.jar
Hello, World!
```

```
/* hello.kt */
fun main(){
    println(“Hello, World!”)
}
```

Java version to use

- Your output java source code must be compiled by **Java 9** compiler
  - higher version is not allowed
  - Your submission will be tested in *openjdk 9.0.4*
- Factory methods for collections
  - Create unmodifiable *set* instance in Java 8 vs. 9

```
/* Java 8 set  */
Set<String> set = new HashSet<>();
set.add("a");
set.add("b");
set.add("c");
set = Collections.unmodifiableSet(set);
```

```
/********
Java 9
similar to Kotlin
val set = setOf("a", “b", “c”)
********/
Set<String> set = Set.of("a", "b", "c");
```

  - http://openjdk.java.net/jeps/269

Install Java 9 Compiler to test

- Install Java

```
$ sudo add-apt-repository ppa:openjdk-r/ppa
$ sudo apt-get update
$ sudo apt-get install openjdk-9-jdk
```

- Check Java version

```
$ java –version
openjdk 9.0.4
…
$ javac –version
javac 9.0.4
```

Reference

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

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