**DwKotlinScriptCompiler**

**Student name:** Dong Wang

**Language name:** DwKotlinScript

**Why this language, and why this language design?**

I am familiar with Android development. Recently Google have announced Kotlin is the first option language in Android development rather than Java. This is why I select Kotlin like language design. Kotlin is based on JVM and compatible with Java. Kotlin is also very similar to Swift language which is used for iOS development. Obviously, I only design small part of Kotlin language features in my language. I keep the array creation, type inference, semicolon inference, string interpolation features in my language.

**Code snippets**

var a = arrayOf(3, 2, 5, 6, 8, 9, 2, 4)

ArrayOf expression can create array with initialed values. You can also see type inference here, a does not declared as any type, but it will figure out what the type of a is.

var primes = Array(n + 1, {i -> true});

Array expression can also create array with size and lambda expression. N + 1 is the size of this array, i is the index of array. True is the return expression of each index of array. Basically, this sentence means initial each primes[i] = true. You can also initial array with return expression also include i, For instance, if you use Array(5, {i-> 2\* i}), it will generate an array like [0, 2, 4, 6, 8].

for(i in a) {  
 println(i)  
}

This is for-in array loop, a is an array reference. In this loop, i will be iterator to be assigned every value in the array.

for(i in 1..10 step 2) {  
 println(i)  
}

This is for-in range loop, var(Int)..var(Int) is a range expression, step means how long iterator will cross in every loop. Range expression includes the start variable but not include the end variable. You can also not set step value, it will make step with 1 as default in every loop period.

println("$count prime(s) less than or equal to $n")

This is println statement, it use string interpolation, if count is 168, n is 1000, it will print “168 prime(s) less than or equal to 1000”. You can put expression in a string, if you want print the value of expression, you just need to add $before expression, for a + b expression,

Use ${a + b}, it will figure out what a + b is and print it out.

**Know limitations**

This language does not support class, it limits me to support lambda expression even there is support syntax but not make it happen in code generation phase, this language can only compile one single source code file, it cannot import library from java standard library.

**What I know now, what would I do differently?**

Before code generation phase, I do not learn a lot actually. After code generation phase, I have learnt a lot about JVM bytecode. The code in syntax level is very different in JVM bytecode level. Actually, you can define syntax whatever you want, but in code generation, it is not easy to do so. For instance, I got a lot of problems when I deal with assign statement. When you assign a value to variable, you cannot load the value of that variable, you can only load the reference of the variable. Assign a value to an array with index variable, you can only load the array reference and the index, then assign value to the array with index. For SelfOperation expression, such as i++, i--. It also makes some troubles for me. If you let i++ in Additive expression like ++i + 2, you need to load the value of ++i, but if you only let ++i; as a statement, after you do i increase by 1 operation, you should not load the value of i. It will make unknown bug from JVM and it is hard to track where is bug is.

Other thing, I want to notice is I use the ASM Outline code plugin in Intellij to help me compile java file and show me JVM bytecode and ASM code. It helps me a lot to figure out how to write JVM bytecode with ASM library. The Intellij also help me see the class file after compiled in readable format, which means Intellij can show me what bytecode looks like in java format.

The last thing I want to say is I learned something about JUnit test. Before taking this class, I heard about JUnit test but I never touch it in development. Now I think I will use it for future development.

I will probably not implement high order function in my language if I will not support class in my design language now. I will also not support mutableList in my language, because it will touch List class method but class is not support in this language.

**How to compile the compiler?**

It is very easy to compile the compiler. Because I wrote this compiler in Intellij, you can just clone the source code, load the project in Intellij to compile it.

**How to run the compiler?**

Run the main method in Dwks.java file, you will get the whole compiler to use. Put the source code file end with .ks in project folder. Type dwks \*.ks to compile the source code file, \* refer the name of source code file, then you can type java \* to run the source code.

**Formal syntax definition**

Basic type::= Int | String | Boolean | Unit | Any

type :: = Int | Boolean| String | Array<basic type> | ‘(‘type\*’)’ -> type | T

e :: = ‘(‘ e ’)’ | e1 binop e2 | unop e | fn’(‘e’)’ | var | arrayOf(var\*) | Array(e1(Int), {e2((Int) -> basic type)}) | var = e | var += e | var -= e | var \*= e | var /= e |‘{‘ (var : type)\* -> e(return basic type) ‘}’ | '$'var | '$''{'e'}'

binop :: = ‘-’ | ‘+’ | ‘/’ | ‘\*’ | ‘%’ | ‘||’ | ‘&&’ | ‘<’ | ‘>’ | ‘<=’ | ‘>=’ | ‘==’ | ‘!=’ | ‘+=’ | ‘-=’ | '\*=' | '/=' | '='

unop :: = ‘!’ | ‘++’ | ‘--’

s :: = ‘if’ ‘(’ e ‘)’ ‘{‘ s\* ‘}’ ‘else’ ‘{‘ s\* ‘}’ | ‘if’ ‘(’e’)’ ’{’ s\* ‘}’ | ‘while’ ‘(’ e ‘)’ ‘{’ s\* ‘}’ | ‘return’ e | break | continue | ‘for’ ‘(’ var ‘in’ Array<basic type> ‘)’ ‘{’ s\* ‘}’ |‘for’ ‘(’ var ‘in’ var(Int)'..'var(Int) ‘)’ ‘{’ s\* ‘}’ | ‘for’ ‘(’ var ‘in’ var(Int)'..'var(Int) step var(Int)‘)’ ‘{’ s\* ‘}’ | ‘var’ var = e | ‘val’ var = e | print’(‘var’)’ | println’(‘var’)’ | fun funcName((var: Type)\*) { s\* }

P :: = s\*