CS 315 Programming Languages

Homework 3 Report

Subprograms in Kotlin

İlke Doğan 21702215

Section 1

TABLE OF CONTENTS

Kotlin Language

Code Explanation:

In Part 1, in Kotlin language, nested subprograms are tested by using functions in function. The terminology here was to test reaching variables inside functions in function variables. In Part 2, scope of local variables are tested by 3 different algorithms. First one is to find the behavior of global or local variables. The second one is to find a local variable in function with the same name. Third one is a local variable test in 'for' loop with a temp variable. In part 3, parameter passing methods are tested by using 2 functions. A Function is called from the other function by using parameter passing methods. In that way, parameter passing method behaviour was observed. In part 4, keyword and default parameters concept is tested by using a function. Default behaviour of functions is observed by changing the parameter count at a time in the main function. In that way the behavior of giving default is reached one by one. In part 5, Kotlin closure structure enables lambda to access its local variables in the algorithm it resides in. In Kotlin, to test closure working behaviour, a string array is used. By using foreach method one by one all elements of the array are reached and to reach them "it" special word is used. Rather than 'it', `item ->` could also be used.

Compiler URL:
https://play.kotlinlang.org/#eyJ2ZXJzaW9uIjoiMS41LjAiLCJwbGF0Zm9ybSI6ImphdmEiLCJhcmdzIjoiIiwianNDb2RlIjoiIiwibm9uZU1hcmtlcnMiOnRydWUsInRoZW1lIjoiaWRlYSIsImNvZGUiOiIvKipcbiAqIFlvdSBjYW4gZWRpdCwgcnVuLCBhbmQgc2hhcmUgdGhpcyBjb2RlLiBcbiAqIHBsYXkua290bGlubGFuZy5vcmcgXG4gKi9cblxuZnVuIG1haW4oKSB7XG4gICAgcHJpbnRsbihcIkhlbGxvLCB3b3JsZCEhIVwiKVxufSJ9

1. Nested subprogram definitions,

In Kotlin, nested subprograms are tested by using two return type functions. The sum variable is created in the function, then used in the inside function in function. The variable is reached and the output message was shown.

Code:

```
 /*
    1.   Nested subprogram definitions
 */
fun sum(x: Int): Int{
    val sum = x + 10
    fun mult(y: Int): Int{
        return sum * y
    }
    return mult(15)
}
    //1.     Nested subprogram definitions
    var x = 8
    println("★★★    Nested subprogram definitions   ★★★")
    println("Function in function is reached and the result is:
${sum(x)}\n")
```

Output:

★★★ Nested subprogram definitions ★★★
Function in function is reached and the result is: 270

2. Scope of local variables

In Kotlin, to find Scope of local variables 2 different testing algorithms were used. First one is to show in the main, to show num1's output will be global or local variable. The second one is again num1 but in the function. Third one is a local variable test in 'for' loop with a temp variable.

Code:

```
val num1 = 10
fun scopeLocal(num1: Int) {
    val num1 = 1
    val temp = 12
    var tot = 0
    println("Scope of Number 1 in function called is: ${num1}\n")
    for (i in 1..num1) {
        tot += i
        val temp = 8
        println("The Scope of temp is for inside for: ${temp}")
    }
    println("The Scope of temp is for outside for: ${temp}")
}
// 2.    Scope of local variables
    println("★★★    Scope of local variables  ★★★")
    val num1 = 5
    println("Scope of Number 1 before function called is: ${num1}\n")
    scopeLocal(num1)
    println("Scope of Number 1 after function called is: ${num1}\n")
    println()
```

Output:

★★★ Scope of local variables ★★★
Scope of Number 1 before function called is: 5
Scope of Number 1 in function called is: 1
The Scope of temp is for inside for: 8
The Scope of temp is for outside for: 12
Scope of Number 1 after function called is: 5

3. Parameter passing methods

In Kotlin, to test Parameter passing methods 2 functions were used. sumPassingVal Function is called from passingVal function by using parameter passing methods.

Code:

```
/*
    3.    Parameter passing methods
*/
fun sumPassingVal():Int{
    var num3 = 5
    var num4 = 3
    return num3 + num4
}

fun passingVal(fn:()->Int):Int{
    val result=fn()
     return result
}
// 3.    Parameter passing methods
    println("★★★    Parameter passing methods  ★★★")
    println("Parameter passing methods is successful and the result:
${passingVal(::sumPassingVal)}\n")//parameter passing function
```

Output:

★★★ Parameter passing methods ★★★
Parameter passing methods is successful and the result: 8

4. Keyword and default parameters

In Kotlin, to test Keyword and default parameters, one function is created. To test default behaviour the function with 3 parameters is tested. One by one parameters are added in the main.

Code:

```
 /*
    4.    Keyword and default parameters
*/
fun passMethods(num5: Int = 3, num6: Int = 5, num7: Int = 7) {
    println("The default parameter is: ${num5}")
    println("The default parameter is: ${num6}")
    println("The default parameter is: ${num7}")
}
// 4.    Keyword and default parameters
    println("★★★    Keyword and default parameters  ★★★")
    println("No parameter is entered")
```

```
    passMethods()

    println("\nFirst Parameter '10' is entered ")
    passMethods(10)

    println("\nFirst '10' and Second '15' Parameters  are entered")
    passMethods(10, 15)

    println("\nFirst '10', Second '15', and third '20' Parameters are
entered")
    passMethods(10, 15, 20)
println("\nChanging the Parameter's value directly in the main")
    println("\n num5: Int = 3, num6: Int = 5, num7: Int = 7 will be
changed")
    passMethods(num5= 67, num6 = 35, num7= 53)
```

Output:

★★★ Keyword and default parameters ★★★
No parameter is entered
The default parameter is: 3
The default parameter is: 5
The default parameter is: 7

First Parameter '10' is entered
The default parameter is: 10
The default parameter is: 5
The default parameter is: 7

First '10' and Second '15' Parameters are entered
The default parameter is: 10
The default parameter is: 15
The default parameter is: 7

First '10', Second '15', and third '20' Parameters are entered
The default parameter is: 10
The default parameter is: 15
The default parameter is: 20

Changing the Parameter's value directly in the main
num5: Int = 3, num6: Int = 5, num7: Int = 7 parameters are changed
The default parameter is: 67
The default parameter is: 35
The default parameter is: 53

5. Closures

In Kotlin, to test closure working behaviour, a string array is used. By using foreach method one by one all elements of the array are reached and to reach them “it” special word is used.

Code:

```
println("\n★★★  Closures  ★★★")
    val stringClosure = arrayOf("ilke", "dogan", "says", "hello")
    println("For String array : Closures \"ilke\", \"dogan\", \"says\",
\"hello\"")
    stringClosure.forEach { val tempString = println("Closures is
successful and the result:$it")}
```

Output:

```
★★★          Closures ★★★
For String array : Closures "ilke", "dogan", "says", "hello"
Closures is successful and the result:ilke
Closures is successful and the result:dogan
Closures is successful and the result:says
Closures is successful and the result:hello
```

Learning Methodology

While learning the kotlin language and its rules, I looked at mostly the sources on the internet. I used ‘ https://kotlinlang.org/ ‘ url to gain the comprehensive perspective. Additionally, sometimes, I could not understand the meaning of the code examples on this website. I looked at other websites in other languages like Java, Python to understand the concept of example.

Conclusion (Readability & Writability)

In conclusion, as an important note for myself was that kotlin closure structure enables lambda to access its local variables in the algorithm it resides in. I think closure has a quite advantageous in terms of writability and readability. For example, I found out that the closure is a lambda expression and actually the values we process have anonymous property. For example, I learned that we can do anything that can be done with a normal object using closures, and even that they are

reversible.In terms of writability, kotlin language is quite easy to write in terms of its expression, methods, and loops. Specifically, for the parameter passing method execution, I found it useful for both for writability and readability. In terms of readability, I think kotlin language is very similar to java language and for me java language is user-friendly in terms of readability. Based on this, I found the readability of Kotlin very easy and the examples I gave above prove that argument.