CS315

PROGRAMMING LANGUAGES

HOMEWORK 3

HACI ÇAKIN

21802641

Table of Content

1) Kotlin

1) Nested Subprogram Definitions

Ex.

```
fun main() {

    fun sumOfSubractionMultiplication(x: Int): Int{
        val sum = x + 10
        fun multiply(y: Int): Int{
            return sum * y
        }
        fun substract(y: Int): Int{
            return sum - y
        }
        return multiply(10) + substract(10)
    }
    var x = 5
    println("Result is : ${sumOfSubractionMultiplication(x)} ")
}
```

Output.

Result is : 155

Explanation Of Code:

In this code part, we define subprogram in another subprogram and we call them. First of all, we add 10 to 5 then sum will be 15. We return summation of multiplication between 15 and 10 and subtraction between 15 and 10. This causes the result 155.

2) Scope of Local Variables

Ex.

```
    var numberForScope1 = 0
    var numberForScope2 = 1
    var numberForScope3 = 5

    fun scopeOfSubProgram(value : Int){
        var numberForScope1 = 2
        numberForScope2 = 7
        var numberForScope4 = value
        numberForScope1 = 6;
        println("NumberForScope1 i is defined and called in function ${numberForScope1} in
the function")
        println("NumberForScope2 i is defined and called in function ${numberForScope2} in
the function")
        println("NumberForScope3 i is defined and called in function ${numberForScope3} in
the function")
    }
    println("NumberForScope1 i is called in outside of function ${numberForScope1}
before function called")
    println("NumberForScope2 i is called in outside of function ${numberForScope2}
before function called")
    println("NumberForScope3 i is called in outside of function ${numberForScope3}
before function called")
    scopeOfSubProgram(10)
    //println("NumberForScope3 i is defined and called in function $
{numberForScope4}") // unreachable
    println("NumberForScope1 i is called in outside of function ${numberForScope1} after
function called")
    println("NumberForScope2 i is called in outside of function ${numberForScope2} after
function called")
    println("NumberForScope3 i is called in outside of function ${numberForScope3} after
function called”)
```

Output.

NumberForScope1 i is called in outside of function 0 before function called

NumberForScope2 i is called in outside of function 1 before function called

NumberForScope3 i is called in outside of function 5 before function called

NumberForScope1 i is defined and called in function 6 in the function

NumberForScope2 i is defined and called in function 7 in the function

NumberForScope3 i is defined and called in function 5 in the function

NumberForScope1 i is called in outside of function 0 after function called

NumberForScope2 i is called in outside of function 7 after function called

NumberForScope3 i is called in outside of function 5 after function called

Explanation Of Code:

In kotlin if variable is called again with var or val keyword, its scope is inside the function where it is defined. Also subprograms can use the outer program function and change them. If without calling outer parameter with val or var keywords, then it affects it. However, if in the subprogram there is a variable with same name which is calling in function by using var or val keywords can not be used from the outer. After this point, subprogram can not affect outer parameter whose name is same.

3) Parameter Passing Methods

In kotlin we can give the function as a parameter of another function, which means that functions can be parameter in kotlin.

Ex.

```
fun calculateProduct():Int{
    return 4 * 3
}

fun parameterPassingMethod(func:()->Int):Int{
    return func()
}
println("Parameter passing method's result is ${parameterPassingMethod(::calculateProduct)} \n\n");
```

Output.

Parameter passing method's result is 12

Explanation Of Code:

In kotlin, functions can be used parameter of another function. In this example parameterPassingMethod takes another function as a parameter and call it in itself.

4) Keyword and Default Parameters

We are able to specify the names of arguments that we are passing to the function and giving default parameters in kotlin. Also we can use given names while we are calling the function.

Ex.

```
fun keywordAndDefaultParameter(temp: Int = 0, trace: Int =2, number : Int=3){
    println("In the method parameter temp is ${temp}")
    println("In the method parameter trace is ${trace}")
    println("In the method parameter number is ${number}")
}

fun sayMyName(name: String = "Empty",surname:String = "Empty"){
    println("My name is ${name} ${surname}")
}

println("With 0 parameter ---- \n")
sayMyName()
keywordAndDefaultParameter()
println("With 1 parameter ---- \n")
sayMyName("Hacı")
keywordAndDefaultParameter(15)
println("With 2 parameter ---- \n")
keywordAndDefaultParameter(30)
println("With keywords \n")
sayMyName(name= "Hacı",surname="Çakın" )
```

```
keywordAndDefaultParameter(number= 3333)
```

Output.

With 0 parameter ----

My name is Empty Empty

In the method parameter temp is 0

In the method parameter trace is 2

In the method parameter number is 3

With 1 parameter ----

My name is Hacı Empty

In the method parameter temp is 15

In the method parameter trace is 2

In the method parameter number is 3

With 2 parameter ----

In the method parameter temp is 30

In the method parameter trace is 2

In the method parameter number is 3

With keywords

My name is Hacı Çakın

In the method parameter temp is 0

In the method parameter trace is 2

In the method parameter number is 3333

Explanation Of Code:

In kotlin, function parameters can be declared using various ways. Also they can have default value. While calling these functions, user can enter or not any parameter because of default values or can enter value to parameter by using parameter's name.

5) Closures

In kotlin, there is a usage of closure which means that having access to variables that were local but are no longer in scope.

Ex.

```
var total = 0
var studentNumber = 0
studentGrades = listOf(90,80,85,30,90,90,20,10)
studentGrades.forEach(it)->{
    total = total + it
    studentNumber++
}
println("Average scor is ${total/studentNumber}”)
```

Output.

Average score is 61

Explanation Of Code:

In this example, it is not in the scope but usage of closure makes it enable to reach this variable.

2) Evaluation of Kotlin

Kotlin as a language, is above the average in terms of writability and readability. In this perspective, its syntax is closer to javascript and dart(closures, keywords). Closures increases the writability and using keywords in syntax increases the readability. Using default parameters in functions increases the reliability but decreases the writability. Of course when it gains from the writability, it loses from the readability. However, Kotlin tries to satisfy each part. It tries to get each good points from various languages. Nested subprogram mechanism is also similar to dart. Being compatible with java and being null-safety is positive points of the Kotlin

3) Learning Strategy and Sources

While I’m learning the Kotlin to complete homework, I mainly used the official documentation(https://kotlinlang.org/). I had spent time with Kotlin in the past for mobile development. However, I forgot most of the things. Looking code examples from the documentation and sites like stackoverflow ,udemy was enough to remember.

Compiler : https://play.kotlinlang.org/