코틀린을 활용한 안드로이드 앱 개발

코틀린 기초문법 이해 – Variable, Type, Collection

한국기술교육대학교
온라인평생교육원

학습내용

1. 코틀린 파일의 구성
2. Variable의 이해
3. Data Type
4. Array, List, Map

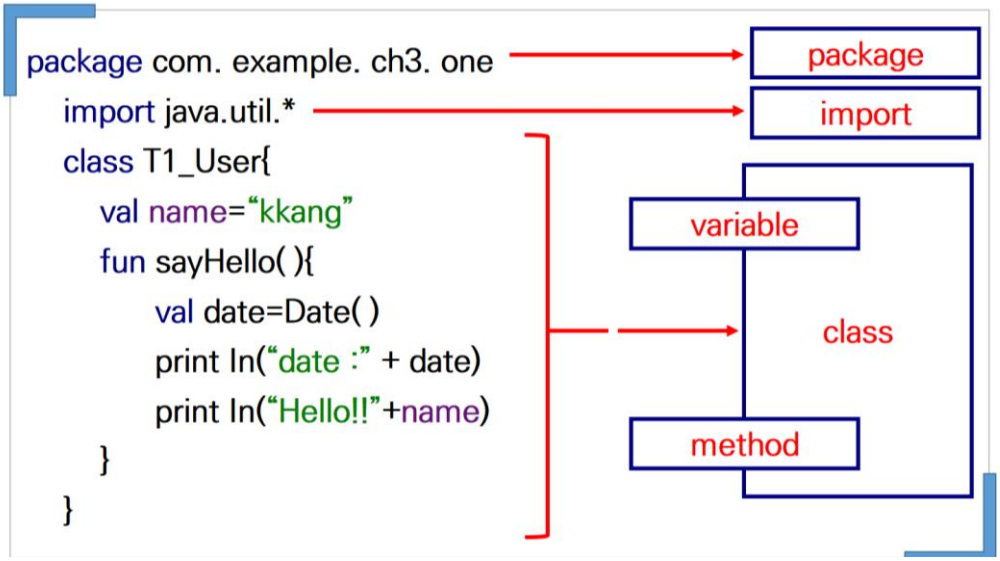
학습목표

1. 코틀린 파일의 구성에 대해 이해할 수 있다.
2. Variable에 대해 이해할 수 있다.
3. Data type에 대해 이해할 수 있다.
4. Array, List, Map에 대해 이해할 수 있다.

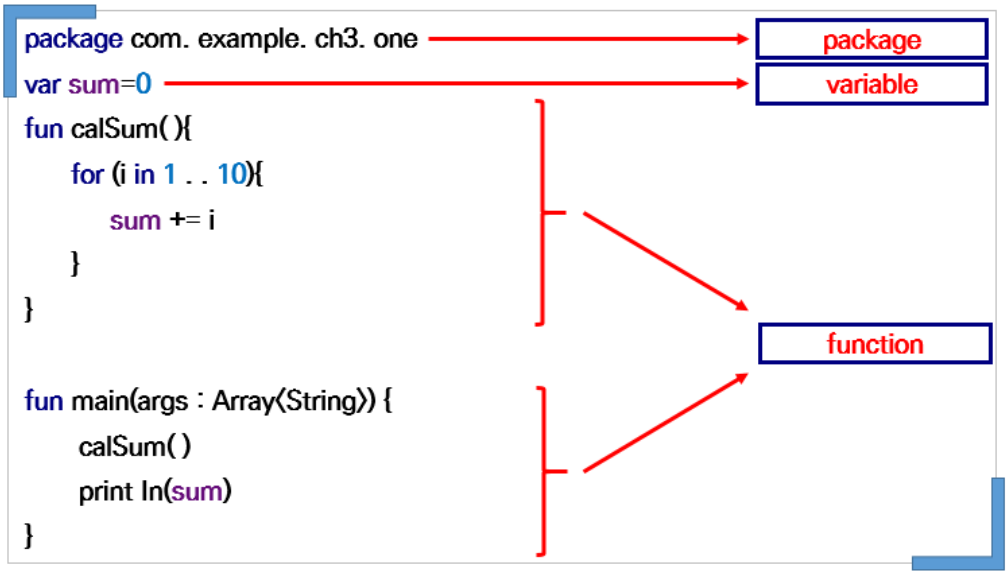
1. 코틀린 파일의 구성

1 코틀린 파일

- 패키지(package), 임포트(import), 클래스, 변수, 함수 선언과 주석이 파일에 포함



- 클래스를 사용하지 않고 변수와 함수로만 구성 가능

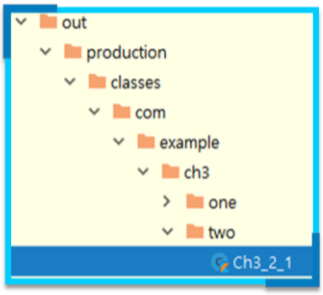


1. 코틀린 파일의 구성

2 패키지

- 관련된 클래스들을 묶기 위한 물리적인 개념

```
package com.example.ch3.two

class Ch3_2_1
```



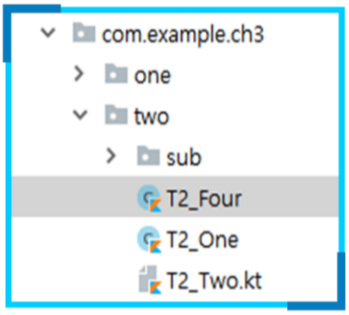
- 이용하려는 클래스가 다른 패키지에 있다면 import 구문

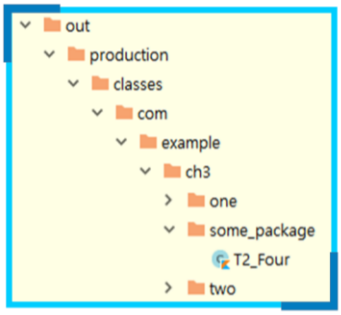
```
package com.example.ch3.two

import com.example.ch3.two.sub.T2_Three

fun main(args: Array <String>) {
    val one=T2_One( )
    val three=T2_Three( )
}
```

1. 코틀린 파일의 구성

2 패키지

- 코틀린 파일이 있는 폴더와 다른 패키지명을 사용할 수 있음






```
package com.example.ch3.some_package

class T2_Four
```

- 클래스로 묶지 않은 변수와 함수를 최상위 레벨로 관리
- 패키지 내에 선언된 전역변수나 전역함수처럼 취급

```
import com.example.ch3.two.sub.threeFun

import com.example.ch3.two.sub.threeVal
```

2. Variable의 이해

1 변수 선언

- val(혹은 var) 변수명 타입 = 값
- val(value)은 Assign-once 변수, var(variable)은 Mutable 변수
- 타입 추론 지원

```
val data1: Int = 10
val data2 = 20
var data3 = 30

fun main(args: Array<String>) {
//    data2 = 40//error
    data3 = 40//ok~~~~
}
```

2. Variable의 이해

2 변수 초기화

- 변수 선언은 최상위(클래스 외부), 클래스 내부, 함수 내부에 선언
- 최상위 레벨이나 클래스의 멤버 변수는 선언과 동시에 초기화해주어야 함
- 함수 내부의 지역 변수는 선언과 동시에 초기화하지 않아도 됨
  - 초기화한 후 사용할 수 있음

```
val topData1: Int//error

class User {
    val objData1: String//error

    fun some(){
        val localData1: Int//ok...
        var localData2: String//ok...

        println (localData1)//error

        localData2="hello" //ok...
        println (localData2)//ok...
    }
}
```

2. Variable의 이해

3 null이 될 수 있는 변수와 null

- 코틀린에서는 null을 대입할 수 없는 변수와 있는 변수로 구분
- 변수에 null 값을 대입하려면 타입에 "?" 기호를 이용하여 명시적으로 null이 될 수 있는 변수로 선언

```
//val nonNullData: String = null//error
val nullableData1 : String? = null
var nullableData2 : String? = null

fun main(args: Array<String>) {
    nullableData2 ="hello"//ok
}
```

4 상기변수 선언

- 코틀린에서 변수 → 프로퍼티(property)
- val로 선언한 변수의 초기값을 변경할 수 없음
  - 일반적인 상수변수와는 차이가 있음
- const라는 예약어를 이용해 상수변수를 만듦
- 최상위 레벨로 선언할 때만 const 예약어 사용 가능

```
const val myConst: Int = 10
//const var myConst2: Int = 10//error

class MyClass {
//    const val myConst3 = 30//error
}
fun some(){
//    const val myConst4= 40//error
}
```

3. Data Type

1 코틀린 타입

- Int, Double, Float, Long, Short, Byte, Char, Boolean, String, Any, Unit, Nothing 타입을 제공
- Int, Double 등은 클래스이며 이 클래스로 타입을 명시하여 선언한 변수는 그 자체로 객체

```
val intData : Int = 10
val result = intData.minus(5)
```

2 숫자 타입

- 코틀린의 숫자 타입의 클래스들은 모두 Number 타입의 서브 클래스
- 문자(Characters)는 Number Type이 아님
- Number Type에 대한 자동 형 변형(implicit conversions for number)을 제공하지 않음

```
val a3: Byte=0b00001011
val a4: Int=123
val a5: Int=0x0F
val a6: Long =10L
val a7: Double=10.0
val a8: Double=123.5e10
val a9: Float=10.0f
```

3. Data Type

3 논리, 문자와 문자열 타입

```
val isTrue1 : Boolean = true && false
val isTrue2 : Boolean = true || false
val isTrue3 : Boolean = !true
```

```
val charData ='C'
fun check(c: Char) {
    if (c == 1) { }//error
}
```

- 문자열은 escaped string 과 raw string 으로 구분

```
val str2="Hello ₩n World"
val str3="""Hello
World"""
```

- string template 지원

```
println ("result : $name .. ${sum(10)}")
```

3. Data Type

4 Any 타입

- 코틀린 클래스의 최상위 클래스가 Any임

```
fun getLength(obj : Any) : Int {
    if(obj is String) {
        return obj. length
    }
    return 0
}

fun main(args: Array<String>) {
    println(getLength("Hello"))
    println(getLength(10))
}
```

5 null 허용 타입

```
val a: Int =null//error
val b: Int? = null//ok~
```

- Any, Any? 타입의 상하위 관계

```
val myVal1: Any = 10
val myVal2: Any?= myVal1

val myVal3: Any? = 10
val myVal4: Any = myVal3//error
val myVal5: Any = myVal3 as Any
```

```
val myInt1: Int = 10
val myInt2: Int? = myInt1

val myInt3: Int? = 10
val myInt4: Int = myInt3//error
val myInt5: Int = myInt3 as Int
```

3. Data Type

6 Unit과 Nothing

```
fun myFun1(){ }
fun myFun2(): Unit { }
```

```
fun myFun(arg: Nothing?): Nothing {
    throw Exception()
}
```

4. Array, List, Map

1 배열

- Array로 표현되며 Array는 get, set, size 등의 함수를 포함하고 있는 클래스
- arrayOf 함수를 이용해 선언

```
fun main(args: Array<String>) {
    //arrayOf 함수 이용
    var array = arrayOf(1, "kkang", true)
    array[0]=10
    array[2]="world"
    println("${array[0]} .. ${array[1]} .. ${array[2]}")
    println("size : ${array.size} .. ${array.get(0)} .. ${array.get(1)} .. ${array.get(2)}")

}
```

```
var arrayInt = arrayOf<Int>(10, 20, 30)
```

```
var arrayInt2= intArrayOf(10, 20, 30)
var arrayDouble= doubleArrayOf(10.0, 20.0, 30.0)
```

- Array 클래스로 선언

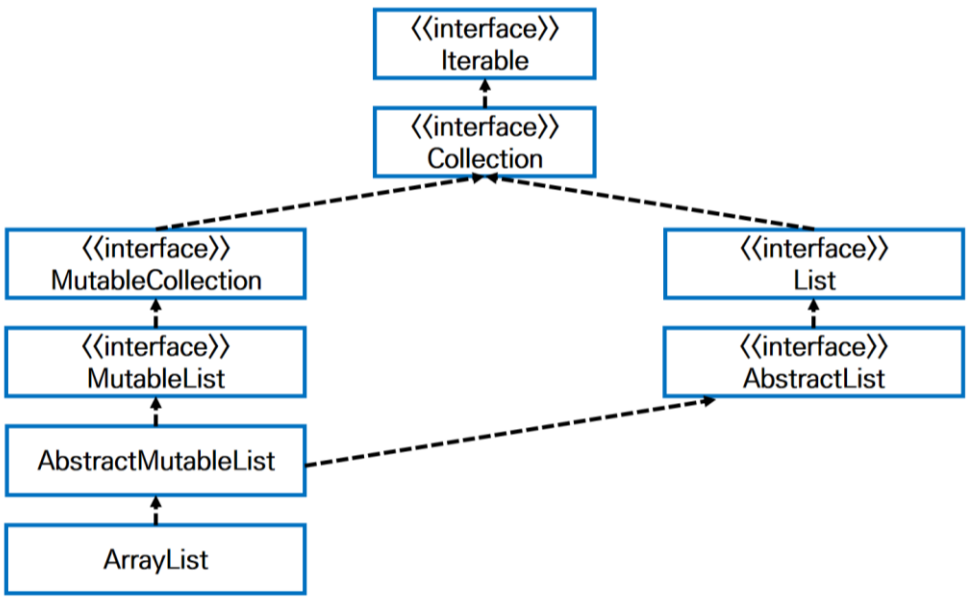
```
var array3=Array(3,  { i -> i*10})
```

```
var array4=Array<Int>(3, {i -> i * 10})
var array5=IntArray(3, { i -> i *10})
```

4. Array, List, Map

2 List, Set, Map

- Collection 타입의 클래스들은 mutable 클래스와 immutable 클래스로 구분
- kotlin. collection. List 인터페이스로 표현되는 객체는 immutable이며 size(), get() 함수만 제공
- kotlin. Collection. MutableList 인터페이스로 표현되는 객체는 mutable이며 size(), get() 함수 이외에 add(), set() 함수 제공



	타입	함수	특징
List	List	listOf()	Immutable
	MutableList	mutableListOf()	mutable
Map	Map	mapOf()	Immutable
	MutableMap	mutableMapOf()	mutable
Set	Set	setOf()	Immutable
	MutableSet	mutableSetOf()	mutable

```
fun main(args: Array<String>) {
    val immutableList: List<String> = listOf("hello", "world")
    println("${immutableList.get(0)} .. ${immutableList.get(1)}")
}
```

```
val mutableList: MutableList<String> = mutableListOf("hello", "world")
mutableList.add("kkang")
mutableList.set(1, "korea")
println("${mutableList.get(0)} .. ${mutableList.get(1)} .. ${mutableList.get(2)}")
```

▪ 정리하기

1. 코틀린 파일의 구성

파일 구성

- 클래스를 사용하지 않고 변수와 함수로만 구성 가능
- 클래스로 묶지 않은 변수와 함수를 패키지 내에 선언된 전역변수나 전역함수처럼 취급

2. Variable의 이해

- val(혹은 var) 변수명 : 타입 = 값
- val(value)는 Assing-once 변수, var(variable)은 Mutable 변수
- 코틀린에서는 null을 대입할 수 없는 변수와 있는 변수로 구분

3. Data Type

- Int, Double, Float, Long, Short, Byte, Char, Boolean, String, Any, Unit, Nothing 타입을 제공
- Int, Double 등은 클래스이며 이 클래스로 타입을 명시하여 선언한 변수는 그 자체로 객체

3. Array, List, Map

- arrayOf 함수를 이용해 배열 선언
- Collection 타입의 클래스들은 mutable 클래스와 immutable 클래스로 구분