Kotlin basictype and function_7/12 수

Variables

var/val 변수명 : 변수타입 // “초기화”

val은 읽기 전용 변수 - 데이터 할당이 1번만 되는 불변 상수 값 // value (java의 final과 비슷)
var은 다시 할당 가능한 변수 // variable

var a : Int = 1

val a : String = “Hello”

Package and Import

Kotlin 프로젝트는 모듈이 있고 이 모듈들은 패키지로 구성되며,

패키지는 파일로 구성된다.
(프로젝트>모듈>패키지>파일>클래스)

```
package ~~ //패키지 정의/선언 -> 패키지 생성
import ~~ //패키지 사용

//- 패키지 정의 및 import
//- 소스 코드의 최상단에 기술
```

패키지를 만드는 이유
—> 패키지를 구별해 같은 이름의 파일(클래스)로 인한 오류를 피하기 위해서

#만약 패키지 선언을 하지 않는다면, default 패키지가 됨

#작성하지 않아도 기본적으로 import됨(아래 8개)

- kotlin.*
- kotlin.annotation.*

- kotlin.collections.*
- kotlin.comparisons.*
- kotlin.io.*
- kotlin.ranges.*
- kotlin.sequences.*
- kotlin.text.*

Basic Type

1. Number

Type	Size (bits)	Min value	Max value
`Byte`	8	-128	127
`Short`	16	-32768	32767
`Int`	32	-2,147,483,648 (-2^{31})	2,147,483,647 ($2^{31} - 1$)
`Long`	64	-9,223,372,036,854,775,808 (-2^{63})	9,223,372,036,854,775,807 ($2^{63} - 1$)

정수

- 명시적 유형 지정 없이 정수형으로 변수를 초기화하면 기본적으로 Int 유형,
  Int 유형 크기를 초과할 경우 Long유형으로 자동 지정

(기본 형식)
val 변수명 : 변수타입 = 값

```
val x = 1 // Int 형
```

유형	크기(비트)	중요한 비트	지수 비트	십진수
`Float`	32	24	8	6-7
`Double`	64	53	11	15-16

부동소수점 유형

- 부동소수점 유형을 타입 지정 없이 변수 초기화 시 기본적으로 Double 유형 사용됨, Float 유형을 명시적으로 지정하려면 따로 지정해주어야 함.

```
val y = 2.253 // Double 형
```

끝에 접미사 f나 L을 이용해 타입 선언도 가능
val a = 10L // Long 형
val b = 2.44f // float 형

- 다른 유형으로 변환법
  toByte(): Byte

  toShort(): Short

  toInt(): Int

  toLong(): Long

  toFloat(): Float

  toDouble(): Double

  예) 변수.toInt()

2. Unsigned integer type

- 부호 없는 정수 유형(앞에 U추가)

UByte: 부호 없는 8비트 정수, 범위는 0~255

UShort: 부호 없는 16비트 정수, 범위는 0~65535

UInt: 부호 없는 32비트 정수, 범위는 0~2^32 - 1

ULong: 부호 없는 64비트 정수, 범위는 0~2^64 - 1

3. Boolean

val myTrue : Boolean = true

val myFalse : Boolean = false

4. Character

val aChar : Char = 'a’

Char는 문자 타입 없어도 추론되어 자동 선언

5. String

val str = “”

- 문자열은 변경할 수 없다.
- 문자열을 변환하는 작업은 원래 문자열을 변경하지 않고 새 개체에서 결과를 반환한다.

원시 문자열 -> 줄 바꿈과 임의의 텍스트 포함 가능
val text = """

    내용

    내용

"""

6. Array

val array = arrayOf(element1, element2, , ,)

```
fun main() {
    val number = arrayOf(1,2,3)
    number.forEach { println(it) }
}

//
1
2
3
```

val 배열명 = Array(길이) {값들 설명}

```
fun main () {
  val asc = Array(5) { i -> (i * i).toString() }
  asc.forEach { println(it) }
}
//
0
1
4
9
16
```

원시 유형 배열(ByteArray, ShortArray, IntArray)

val 배열 이름: 배열 타입 = 소문자타입ArrayOf(값)

```
val x: IntArray = intArrayOf(1,2,3)
```

```
fun main() {
    val x: ShortArray = shortArrayOf(1,2,3)
    x.forEach{println(it)}
}
//
1
2
3
```

val arr = IntArray(5) // [0,0,0,0,0] 사이즈 5 정수형 배열 선언

값 선언 없을 시 모든 요소 0으로 할당
val arr = IntArray(5) {42} // 하나의 값만 입력시 모든 요소가 42로 할당
val arr = IntArray(5) {it * 1} // [0, 1, 2, 3, 4]

Function

```
fun 함수명(매개 변수: 데이터 타입) : 반환 타입{
}
```

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

* 반환 타입 Unit = Void

**Unit은 생략 가능(생략 시 암시적으로 반환 타입 Unit)

fun sum (a: Int, b: Int) = a+b -> 함수 표현식으로 한 줄 표현, 반환 타입 추론됨

1. infix functions

함수는 기본적으로 객체.함수(매개변수)
중위 함수를 사용 시 객체 함수 파라미터 형태 호출 가능

```
infix fun String.add(value: String): String {
    return this + value
}

println("dev" add "Hudi") // devHudi
```

2. local functions

함수 안에 함수를 선언해서 사용하는 것

```
fun foo() {
    fun bar() {
        println("Hello, world!")
    }

    bar()
}
fun main(){
    foo()
}
///Hello, world!
```

3. extension functions

- 코틀린에서는 특정 클래스의 메소드를 클래스 밖에서 정의할 수 있다. 이 기능을 확장 함수라고 부른다.

```
fun String.printTwice(): String {
    return this.repeat(2)
}
```

String 클래스에 새로운 함수 정의한 예시.

```
println("Hello".printTwice()) // HelloHello
```

`String` 을 수신 객체 타입, `this` 를 수신자라고 부른다.

조건문과 반복문

조건문

1. if문

```
if(조건식){
}else if(조건식){
}else{
}
```

2. when문(java의 switch문을 대신함)

```
when(변수) {
값 -> 명령 //변수와 값이 같으면 명령 실행
값1, 값2 -> 명령 //변수가 값1 혹은(or) 값2와 같은 경우 명령 실행
in 1..10 -> 명령 //1<=변수<=10 일 경우 명령 실행(1~10에 포함될 경우)
!in 1..10 -> 명령 //1~10이 아닐 경우 명령 실행<1..10은 1~10이란 뜻 1<=x<=10)
else -> 명령 //조건이 없을 경우 명령 실행(default)
}
```

반복문

1. for문

```
for(i in 1..10){
  명령
}
//i가 1~9까지 1씩 증가 반복 1<=i<10
for(i in 1..10){
  명령
}
//i가 1(포함)~10(포함)까지 1씩 증가 반복 -> 10번 반복
for(i in 1..10 step 2){
  명령
}
//i가 1(포함)~10(포함)까지 2씩 증가 반복 -> 5번 반복
for(i in 10 downTo 1){
  명령
}
//i가 10(포함)~1(포함)까지 1씩 감소 반복
for(i in 10 downTo 1 step 2){
}
//i가 10(포함)~1(포함)까지 2씩 감소 반복
for(i in 배열명){
}
//배열 요소만큼 반복, i에 인덱스 0부터 순차적으로 대입
```

2. while문

- 자바와 동일

아래는 for문 예제 및 문자열 명령어 설명

```
fun main() {
    val a = arrayOf(33,12,95)
    for ((index,value) in a.withIndex()){
        print("Index: ${index} Value: ${value}\n")
    }
}
//인덱스 값과 해당 인덱스의 배열 값을 받아오고(withIndex사용)
//그 횟수만큼 반복
Index: 0 Value: 33
Index: 1 Value: 12
Index: 2 Value: 95
```

```
fun main() {
    val a = arrayOf(33,12,95)
```

```
    for (i in a.indices){
        println(i)
    }
}

//배열 a길이만큼 반복 i에 index(indices로 받아옴)
0
1
2
```

출력할 때 문자열에서 사용할 수 있는 명령어

- ${변수} → 변수 값 가져와서 문자열에 포함

```
val a = 10
println("a값은 ${a}")
//
a값은 10
```

- `\t` – tab (탭)
- `\n` – new line (LF) (줄 바꿈 / 엔터)
- `\'` – single quotation mark(작은 따옴표 출력할 때)
- `\"` – double quotation mark(큰 따옴표 출력할 때)
- `\\` – backslash(백슬래쉬(\ ) 출력할 때)
- `\$` – dollar sign(달러 모양 출력할 때)

```
println("a\tb\n")
println("\'끝\'")
//
a     b

'끝'
```