코틀린(Kotlin)이란

* 자바 플랫폼에서 사용하는 프로그래밍 언어이다.(자바와 비슷한 성능 수준)
* JVM(Java Virtual Machine)을 돌리는 언어이다.
* 함수형 프로그래밍 객체지향 프로그래밍 모두를 지원하는 언어이다.

코틀린(Kotiln)의 장점

* 실용성 : 연구를 위한 언어가 아니고 다른 언어들이 채택한 검증된 기능을 추가한 언어이다.
* 간결성 : 자바에 비해 코드가 간결하다. 언어가 요구하는 구조를 만족하기 위한 부분들을 삭제하여 보다 내용을 파악하기 쉽고 코드의 유지보수가 쉽다.
* 안전성 : NullSafety 언어이며 Null로부터 안전하다.

코틀린(Kotlin)의 기본문법

-변수-

1. 변수의 종류

- 변경 가능한 변수 : var (variable, 변수)

- 변경 불가능한 변수 : val (value, 값)

2. 선언하는 방법

val/var 변수이름 = 값

ex) val school = “비전 초등학교” (변수 이름을 짓는 것을 네이밍이라 한다)

반드시 선언과 할당을 동시에 해야 한다.

Val 을 선언 할 떄는 대문자로 var 을 선언할 때는 소문자로 네이밍한다.

3. 변수의 네이밍이 중요한 이유

가독성을 향상시키는데 매우 중요하다

두개의 단어를 합쳐서 네이밍을 할 때의 방법

1) 카멜케이스 스타일

대소문자를 반복한다. Ex) phoneNumber

2) 스네이크 스타일

언더바(\_)를 사용한다 Ex) PHONE\_NUMBER

-자료형-

변수의 Type이다.

1. 정수형

음수, 양수 모두를 표현한다.

Byte, Short, Int, Long

양수만 표현 -> UByte, UShort, UInt, ULong

2. 실수형

실수를 표현한다.

Float, Double

3. 논리형

참과 거짓의 표현한다. (Boolean)

True, False

4. 문자형

한글자만을 표현한다.

작은 따옴표로 표현한다.

Ex) ‘A’, ‘가’

5. 문자열형

여러 글자를 표현한다.

큰 따옴표로 표현한다.

Ex) “안녕하세요”, “hello”

6. 자료형 추론

var/val 변수이름: Type = 값

변수를 선언할 때, 자료형을 적어주는 것이 좋다 만약,

Type을 쓰지 않으면..

* 정수의 경우 Int로 추론한다.
* 실수의 경우 Double로 추론한다.

추론힌트를 주는 방법(잘 사용하지 않는다.)

Ex) var number1 = 100 -> Int, Long 등등

-함수-

1. 함수의 기본형

fun 함수명 (변수명: Type, 변수명: Type, …..): 반환값의 Type{

함수내용

함수내용

…….

return 반환값

}

2. 함수의 유형

1) 기본값이 있는 함수

fun 함수명 (변수명: Type = 기본값, 변수명: Type, …..): 반환값의 Type{

함수내용

함수내용

…….

return 반환값

}

기본값을 가지는 인자에 아무런 값이 들어오지 않으면 기본값으로 함수가 돌아간다

2) 반환값이 없는 함수

fun 함수명 (변수명: Type, 변수명: Type, …..): Unit{

함수내용

함수내용

…….

}

Unit은 자바의 void와 비슷하다.

3) 가변인자를 가지는 함수

fun 함수명 (vararg 변수명: Type): 반환값의 Type{

함수내용

함수내용

…….

return 반환값

}

4) 간단하게 선언하는 방법

fun 함수명 (변수명: Type, 변수명: Type) = 함수내용

-분기문-

반복문 내 실행 흐름을 변경하는 구문이다.

1. if문

\* if (만약에)  
- 만약에 ~이라면 A를 해라  
if (A) / else if (B)  
- 만약에 A 이라면 A1를 하고 만약에 B라면 B1을 해라  
if (A) / else if (B) / else  
- 만약에 A 이라면 A1를 하고 만약에 B라면 B1을 그리고 나머지 경우에는 C를 해라  
  
- if 문법  
if (조건){ 조건 -> 숫자 5라면  
 동작1  
}else if(조건){ -> 숫자 10이라면  
 동작2  
}else if(조건){ -> 숫자 10이라면  
 동작3  
}else { -> 5,10 둘다 아닌 경우  
 동작3  
}  
  
- else if 는 여러개 쓸 수 있다  
- if 와 else는 1개만 쓸 수 있다  
- else와 else if 는 필요하지 않으면 사용하지 않을 수 있다

-kotlin에서는 만약 if문 블럭 내의 여러 줄의 식이 있다면 리턴값은 마지막 줄이 된다.  
  
\* when

다른 언어에서의 swith문과 동일하다.

when (조건을 검사할 값){ "안녕하세요"  
 case1 -> 동작1 "안녕하세요" -> return 10  
 case2 -> 동작2 "어서오세요" -> return 20  
 else -> 동작3  
}

-반복문-

1.for문

기본 format

for(변수명 in 범위){

반복문의 내용

}

2.while문

기본 fomat

while(조건){

내용

}

* 조건을 만족하는 동안 본문(내용)을 반복한다.

do/while문 format

do{

내용

}while(조건)

* 조건을 만족하지 못하더라도 반드시 한번은 시행한다. (do부분, 본문)

주의점

while문은 조건을 만족하는 동안 본문이 반복 시행되기 때문에, 무한반복을 하지 않기 위해서는 무조건 본문에 while문의 조건을 만족하지 못하도록 하는 내용을 포함해야만 한다.