04

04-4. 람다함수

다양한 기법

람다 표현식

람다 표현식이란?

- 익명함수(Anonymous Functions)를 지칭하는 용어

- fun 함수이름(매개변수) { 함수내용 }
- { 매개변수 -> 함수내용 }

- 람다함수는 항상 { }에 의해 감싸 표현되어야 한다.
- { }안에 -> 표시가 있으며 -> 왼쪽은 매개변수, 오른쪽은 함수내용이 위치한다.
- 매개변수 타입은 선언되어 있어야 하며 추론이 가능한 경우에는 생략이 가능하다.
- 함수의 리턴 값은 함수내용의 마지막 표현식이다.

```
fun sum(x1: Int, x2: Int): Int {
    return x1+x2
}
```

```
val sum1={ x1: Int, x2: Int -> x1+x2 }

fun main(args: Array<String>) {
    val result1=sum1(10, 20)
}
```

람다 표현식

람다함수를 정의하자 마자 함수 호출

```
{ println("hello")}()
run { println("world") }
```

매개변수가 없는 람다함수 정의

```
val sum2 = { -> 10 + 20 }
```

```
val sum2 = { 10 + 20 }
```

람다 표현식

함수 내용이 여러 문장으로 된 경우의 리턴값

- 함수 내용이 여러 줄일 때 함수의 반환값은 마지막 줄의 결괏값입니다

```
val sum3 = { x1:Int, x2: Int ->
    println("call sum3()....")
    x1 + x2
}
```

람다 표현식

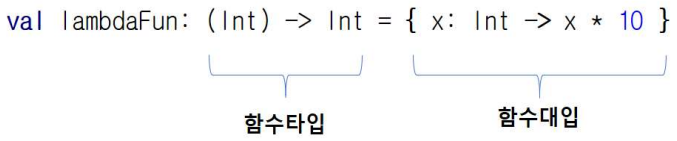
함수 타입

- 변수를 이용할 때 데이터 타입을 명시하고 그 타입에 맞는 값을 할당합니다
- 함수에도 변수처럼 타입이 있습니다.

```
val name: String = "kkang"
val no: Int = 10
```

```
fun myFun(x1: Int, x2: Int): Boolean {
    return x1 > x2
}
```

```
val lambdaFun: (Int) -> Int = { x: Int -> x * 10 }
```



람다 표현식

Typealias를 이용한 타입 정의

- 함수 타입을 정의할 때 typealias 키워드를 이용하여 반복해서 사용되는 함수 타입을 쉽게 정의할 수 있습니다.

```
typealias MyType = (Int) -> Boolean

val myFun: MyType = { it > 10 }
```

람다 표현식

매개변수 타입 생략

- 일반적으로 람다 함수를 정의할 때 매개변수가 있다면 매개변수 타입을 선언하여 작성합니다.
- 그런데 만약 컴파일러가 매개변수의 타입을 추론할 수 있을 때는 타입 선언을 생략할 수 있습니다.

```
val lambdaFun1 =  { x -> x * 10 }//error

val lambdaFun2: (Int) -> Int = { x -> x+ 10 }
```

람다 표현식

---

it을 이용한 매개변수 이용

- 매개변수 하나인 경우에는 함수 내용에서 it으로 매개변수를 지칭
- it을 이용한 람다 함수의 정의는 함수 타입이 정의되어 있는 곳에서만 사용이 되어야 합니다.

```
val lambdaFun3: (Int) -> Int = { x -> x+10 }

val lambdaFun4: (Int) -> Int = { it + 10  }
```

```
val itTest2 = { it * 10 }//it 에러..
```

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare