코틀린을 활용한 안드로이드 앱 개발

코틀린 함수형 프로그래밍

한국기술교육대학교
온라인평생교육원

학습내용

1. 람다 함수 사용법 이해
2. 고차 함수 사용법 이해
3. 기초 고차 함수에 대한 이해

학습목표

1. 람다 함수 사용법에 대해 이해할 수 있다.
2. 고차 함수 사용법에 대해 이해할 수 있다.
3. 기초 고차 함수에 대해 이해할 수 있다.

1. 람다 함수 사용법 이해

1 람다 표현식이란?

- 익명 함수(Anonymous Functions)를 지칭하는 용어
- fun 함수이름(매개변수) { 함수내용 }
  { 매개변수 -> 함수내용 }
- 람다 함수는 항상 { }에 의해 감싸 표현되어야 함
- { } 안에 -> 표시가 있음
  (왼쪽) 매개변수 -> 함수 내용 (오른쪽)
- 매개변수 타입은 선언되어 있어야 하며 추론이 가능한 경우에는 생략 가능
- 함수의 리턴 값은 함수 내용의 마지막 표현식

```
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)
}
```

1. 람다 함수 사용법 이해

1 람다 표현식이란?

매개변수가 없는 람다 함수 정의

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

↓

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

함수 내용이 여러 문장으로 된 경우의 리턴 값

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

2 함수 타입

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

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

```
val lambdaFun: (Int) -> Int = { x: Int -> x * 10 }
               ^^^^^^^^^^^^^   ^^^^^^^^^^^^^^^^^^
               함수 타입        함수 대입
```

1. 람다 함수 사용법 이해

2 함수 타입

Typealias를 이용한 타입 정의

```
typealias MyType = (Int) -> Boolean
val myFun: MyType = { it > 10 }
```

매개변수 타입 생략

```
val lambdaFun1 =  { x -> x * 10 }//error
val lambdaFun2: (Int) -> Int = { x -> x+ 10 }
```

3 it을 이용한 매개변수 이용

- 매개변수가 하나인 경우에는 함수 내용에서 it으로 매개변수를 지칭

```
val lambdaFun3: (Int) -> Int = { x -> x+10 }
val lambdaFun4: (Int) -> Int = { it + 10  }
```

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

2. 고차 함수 사용법 이해

1 고차 함수란?

- 고차 함수(High-Order Function, 고계 함수)
- 함수의 매개변수로 함수를 전달 받거나 리턴시킬 수 있는 함수를 지칭

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

```
fun hoFun(x1: Int, argFun: (Int) -> Int){
    val result=argFun(10)
    println("x1 : $x1, someFun1 : $result")
}

hoFun(10, {x -> x * x })
```

- 함수 타입의 매개변수 대입
- 함수 호출 시 ( ) 생략 가능

```
fun hoFun1(argFun: (Int) -> Int){
    val result=argFun(10)
    println("result : $result")
}
hoFun1({x -> x * x})
hoFun1 {x -> x* x }
```

2. 고차 함수 사용법 이해

1 고차 함수란?

```
val array= arrayOf(10, 20, 15, 22, 8)
array.filter{ x -> x > 10 }
        .forEach{ x -> println(x) }
```

```
fun hoFun_1(no: Int, argFun1: (Int)->Int, argFun2: (Int)->Boolean){

}
hoFun_1(10, {it * it}, {it > 10})
hoFun_1(10, {it * it}) {it > 10}
hoFun_1(10){it * it} {it > 10}//error
```

함수 타입 디폴트 값 이용

```
fun hoFun2(
        x1: Int,
        argFun1: (Int) -> Int,
        argFun2: (Int) -> Boolean = { x: : Int -> x > 10}
){
    val result = argFun1(x1)
    println("result : ${argFun2(result)}")
}

hoFun2(2, { x: Int -> x * x }, {x: Int -> x > 20})
hoFun2(4, { x: Int -> x * x })
```

2. 고차 함수 사용법 이해

2 고차 함수와 함수 반환

```
fun hoFun5(str: String): (x1: Int, x2: Int) -> Int {
    when (str){
        "-" -> return { x1, x2 -> x1 - x2 }
        "*" -> return { x1, x2 -> x1 * x2 }
        "/" -> return { x1, x2 -> x1 / x2 }
        else -> return { x1, x2 -> x1 + x2 }
    }
}

val resultFun=hoFun5("*")
println("result * : ${resultFun(10, 5)}")
```

3. 기초 고차 함수에 대한 이해

1 run()

- 단순 람다 함수를 실행시키고 그 결과 값을 획득
- 객체의 멤버에 접근

```
inline fun <R> run(block: ( ) -> R): R
```

```
val result= run {
    println("lambdas function call...")
    10 + 20
}
println("result : $result")
```

```
inline fun <T, R> T.run(block: T.() -> R): R
```

```
val runResult=user.run {
    name="kim"
    age=28
    sayHello()
    sayInfo()
    10 + 20
}

println("run result : $runResult")
```

3. 기초 고차 함수에 대한 이해

2 apply()

- apply() 함수는 run() 함수와 사용 목적은 동일한데 리턴 되는 값에 차이
- run() 함수의 리턴 값은 대입된 람다 함수의 리턴 값이 그대로 run() 함수의 리턴 값
- apply() 함수는 apply() 함수를 적용한 객체가 리턴

```
inline fun <T> T.apply(block: T.() -> Unit): T
```

```
val user3=user.apply {
    name="park"
    sayHello()
    sayInfo()
}
println("user name : ${user.name}, user3 name : ${user3.name}")
user.name="aaa"
user3.name="bbb"
println("user name : ${user.name}, user3 name : ${user3.name}")
```

3 let()

- let을 이용하는 객체를 let의 매개변수로 지정한 람다 함수에 매개변수로 전달

```
inline fun <T, R> T.let(block: (T) -> R): R
```

```
User("kim", 28).let { user ->
    letTestFun(user)
}
```

3. 기초 고차 함수에 대한 이해

4 With()

- with() 함수는 run() 함수와 사용 목적이 유사
- 객체의 멤버들을 반복적으로 접근할 때 객체명을 일일이 명시하지 않고 멤버들을 바로 이용하기 위한 용도
- run() 함수는 run() 함수를 이용한 객체가 람다 함수에서 바로 이용
- with() 함수는 with() 함수의 매개변수로 지정한 객체를 람다 함수에서 이용

```
inline fun <T, R> with(receiver: T, block: T.() -> R): R
```

```
user.run {
    name="kkang"
    sayHello()
}

with(user){
    name="kkang"
    sayHello()
}
```

▪ 정리하기

1. 람다 함수 사용법 이해

Lambda Function

- 익명함수(Anonymous Functions)
- { 매개변수 → 함수내용 }
- 매개변수 하나인 경우에는 함수 내용에서 it으로 매개변수를 지칭

2. 고차 함수 사용법 이해

High Order Function

- 함수의 매개변수로 함수를 전달 받거나 함수를 리턴시킬 수 있는 함수를 지칭

3. 기초 고차 함수에 대한 이해

Basic Kotlin Hof

- run : 단순 람다함수를 실행시키고 그 결과 값을 획득
- apply : apply() 함수를 적용한 객체가 리턴
- let : 이용하는 객체를 let 의 매개변수로 지정한 람다함수에 매개변수로 전달
- with : with() 함수의 매개변수로 지정한 객체를 람다함수에서 이용