05

05-4. 어노테이션

다양한 기법

어노테이션

어노테이션 작성 및 이용

- 어노테이션은 클래스, 함수, 프로퍼티 선언 앞에 추가되는 구문으로 @로 시작하는 구문

- 컴파일러에게 코드 문법 에러를 체크하기 위한 정보를 제공
- 개발 툴이나 빌더에게 코드 자동 추가를 위한 정보
- 실행시 특정 기능을 실행하기 위한 정보

- 어노테이션은 annotation 예약어로 만들어지는 클래스
- 객체생성 불가
- 실행영역을 가질수 없다.

```
annotation class TestAnnotation

annotation class TestAnnotation2 { }//error

fun main(args: Array<String>) {
    val obj: TestAnnotation = TestAnnotation()//error
}
```

어노테이션

어노테이션 작성 및 이용

```
annotation class TestAnnotation

@TestAnnotation
class Test {

    @TestAnnotation
    val myVal: String = "hello"

    @TestAnnotation
    fun myFun() {

    }
}
```

```
annotation class TestAnnotation

class Test @TestAnnotation constructor(){
    @TestAnnotation
    val myVal: Int=10

    var myVal2: Int = 10
        @TestAnnotation set(value) { field = value }

    val myFun = @TestAnnotation{

    }
```

어노테이션

데이터 설정

- 주생성자를 이용해 데이터가 설정
- val이 추가되어야 하며 var은 허용되지 않는다.

```
annotation class TestAnnotation(val count: Int)
class Test {
    @TestAnnotation(count=3)
    fun some(){
        println("some.....")
    }
}
fun main(args: Array<String>) {
    val obj: Test = Test()
    val methods = Test::class.java!!.methods

    for(method in methods){
        if(method.isAnnotationPresent(TestAnnotation::class.java)){
            val annotation=method.getAnnotation(TestAnnotation::class.java)
            val count = annotation.count
            for(i in 1..count){
                obj.some()
            }
        }
    }
}
```

어노테이션

허용 데이터 타입

- 자바의 기초 타입(Int, Long etc.)
- string
- classes(Foo::class)
- enums
- other annotations
- array of the types listed above

```
annotation class TestAnnotation1(val count: Int)

annotation class TestAnnotation2(val otherAnn: TestAnnotation1, val arg1: KClass<*>)

class User

//annotation class TestAnnotation3(val user: User)//error

@TestAnnotation1(10)
@TestAnnotation2(TestAnnotation1(20), String::class)
class Test { }

const val myData: Int = 10
@TestAnnotation1(myData)
class Test2 { }
```

어노테이션

어노테이션 선언 옵션

- @Target : 어느 곳에 사용하기 위한 annotation인지를 명시하기 위해서 사용 (classes, functions, properties, expressions)
- @Retention : annotation을 컴파일한 클래스에 보관할지, 런타임시 Reflection에 의해 접근할수있는지에 대한 설정. (source, binary, runtime)
- @Repeatable : 이 annotation을 한곳에 반복 사용이 가능하게 설정.
- @MustBeDocumented : annotatoin을 api에 포함시켜야 하는지, api document 문서에 포함되어야 하는지에 대한 설정

어노테이션

어노테이션 선언 옵션

```
@Target(AnnotationTarget.CLASS, AnnotationTarget.FUNCTION,
        AnnotationTarget.VALUE_PARAMETER, AnnotationTarget.EXPRESSION)
@Retention(AnnotationRetention.SOURCE)
annotation class TestAnnotation

@TestAnnotation
class Test {
    @TestAnnotation constructor(no: Int){}//error

    @TestAnnotation//error
    val myVal: Int = 10

    @TestAnnotation
    fun myFun(@TestAnnotation no: Int) {
        val result = @TestAnnotation if (no>10){
            10
        }else {
            0
        }
    }
}
```

어노테이션

어노테이션 적용 대상 지정

- file
- property (annotations with this target are not visible to Java)
- field
- get (property getter)
- set (property setter)
- receiver (receiver parameter of an extension function or property)
- param (constructor parameter)
- setparam (property setter parameter)

```
class Test constructor(@param: TestAnnotation var email: String){

    @get:[TestAnnotation TestAnnotation2]
    var no: Int=10

    @property: TestAnnotation
    var name: String = "kkang"

    @field: TestAnnotation
    var age: Int = 30

    @setparam: TestAnnotation
    var phone: String= "0100000"
}
fun @receiver: TestAnnotation Test.myFun(){ }
```

어노테이션

자바 어노테이션 이용

- 자바로 선언된 어노테이션을 코틀린에서 사용이 가능
- 데이터 설정이 되어야 한다면 데이터의 순서적인 문제
- 데이터를 대입시킬 때 꼭 이름을 명시

```java
public @interface JavaAnnotation {
    int intValue();
    String stringValue();
}
```

```kotlin
annotation class KotlinAnnotation(val no: Int, val name: String)

@KotlinAnnotation(10, "kkang")
//@JavaAnnotation(10, "kkang")//error
@JavaAnnotation(intValue = 10, stringValue = "kkang")
class Test { }
```

어노테이션

자바 어노테이션 이용

- 자바 어노테이션의 함수명이 value 로 되어 있다면 이때는 이름을 명시하지 않아도 된다.

```
public @interface JavaAnnotation2 {
    int value();
    String strValue();
}
```

```
@JavaAnnotation2(10, strValue = "kkang")
class Test { }
```

- 데이터가 배열로 대입되어야 하는 경우 배열의 함수가 value 로 선언되었다면 데이터만 나열, value 함수가 아닌 경우는 arrayOf() 를 이용

```
public @interface JavaAnnotation3 {
    int[] value();
    String[] strVale();
}
```

```
@JavaAnnotation3(10, 20, strVale = arrayOf("kkang", "kim"))
class Test { }
```

감사합니다

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

윌리엄 셰익스피어
William Shakespeare