02

02-6. 연산자

Basic Syntax

연산자

대입 연산자

연산자	사용법	설명
=	A=B	B 값을 A에 대입

산술 연산자

연산자	사용법	설명
+	A+B	A와 B의 값을 더하기
-	A-B	A에서 B를 빼기
*	A*B	A와 B를 곱하기
/	A/B	A에서 B를 나누기
%	A%B	A를 B로 나눈 나머지

연산자

전개 연산자

연산자	사용법	설명
*	*A	A 배열의 데이터를 나열

```
fun main(args: Array<String>) {
    val array1= arrayOf(10, 20, 30)

    val list1= asList(1, 2, array1[0], array1[1], array1[2], 100, 200)
    list1.forEach({println(it)})
}
```

```
val array1= arrayOf(10, 20, 30)

val list2=asList(1, 2, *array1, 100, 200)
list2.forEach({println(it)})
```

연산자

전개 연산자

- vararg 의 argument 적용

```
fun some(vararg a: String){
    val iterator=a.iterator()
    while(iterator.hasNext()){
        println(iterator.next())
    }
}

fun main(args: Array<String>) {
    val array3= arrayOf<String>("hello","world")
    some(*array3)
}
```

- List 에 이용

```
val list3= listOf<String>("a","b")
some(*list3.toTypedArray())
```

연산자

복합 대입 연산자

연산자	사용법	설명
+=	A+=B	A와 B의 값을 더한 값을 A에 할당
-=	A-=B	A에서 B를 뺀 값을 A에 할당
*=	A*=B	A와 B를 곱한 값을 A에 할당
/=	A/=B	A에서 B를 나눈 값을 A에 할당
%=	A%=B	A를 B로 나눈 나머지 값을 A에 할당

증감 연산자

연산자	사용법	설명
++	A++, ++A	A값에 1을 더해 결과 값을 A에 할당
--	A--, --A	A값에서 1을 빼 결과 값을 A에 할당

연산자

논리 연산자

연산자	사용법	설명
&&	A && B	A와 B가 모두 true라면 결과값이 true,
||	A || B	A와 B가 하나라도 true이면 결과값은 true
!	!A	A가 true이면 결과값은 false, false이면 결과값은 true

연산자

일치 연산자

연산자	사용법	설명
==	A == B	A와 B의 값을 비교, 같은 값이라면 결과값이 true,
!=	A != B	A와 B의 값을 비교, 값이 다르다면 결과값은 true
===	A === B	A와 B가 같은 객체인지를 비교, 같은 객체면 결과값은 true
!==	A !== B	A아 B가 같은 객체인지를 비교, 다른 객체이면 결과값은 true

- ==을 structural equality, ===은 referential equality
- ==은 값에 대한 비교이고 ===은 객체에 대한 비교
- 일반 객체인지, 기초 데이터 타입의 객체인지에 따라 차이

연산자

일치 연산자

- 일반 객체 이용

```
fun main(args: Array<String>) {
    class User
    val user1=User()
    val user2=User()
    val user3=user1
    println("user1==user2 is ${user1==user2}")//false
    println("user1===user2 is ${user1===user2}")//false
    println("user1==user3 is ${user1==user3}")//true
    println("user1===user3 is ${user1===user3}")//true
}
```

```
val data1: Int=10
val data2: Int=10
println("data1==data2 is ${data1==data2}") //true
println("data1===data2 is ${data1===data2}") //true
```

연산자

비교 연산자

연산자	사용법	설명
<	A < B	A가 B 보다 작으면 true
>	A > B	A가 B 보다 크면 true
<=	A <= B	A가 B 보다 작거나 같으면 true
>=	A >= B	A가 B 보다 크거나 같으면 true

범위 연산자

연산자	사용법	설명
..	A..B	A부터 B까지의 수를 묶어 범위 표현

연산자 재정의

연산자 재정의 방법

- 연산자 재정의는 함수의 재정의를 통해 작성

```
fun main(args: Array<String>) {
    val a: Int = 10

    val b: Int = 5

    val result1: Int = a + b

    val result2: Int = a.plus(b)

    println("result1 : $result1 ... result2 : $result2")
}
```

연산자 재정의

연산자 재정의 방법

- 연산자 재정의는 연산자에 대응하는 함수를 재정의하는 것인데, 이는 클래스의 맴버로 정의할 수도 있고, 확장 함수로 추가하여 사용할 수도 있습니다.
- 연산자를 재정의하려는 함수 앞에는 operator라는 예약어를 추가해야 합니다.

```
class Test(val no: Int) {
    operator fun plus(arg: Int): Int {
        return no - arg
    }
}

fun main(args: Array<String>) {
    println("${Test(30) + 5}") //25
}
```

연산자 재정의

연산자 함수

Expression	Translated to
+a	a.unaryPlus()
-a	a.unaryMinus()
!a	a.not()

Expression	Translated to
a++	a.inc()
a--	a.dec()

Expression	Translated to
a + b	a.plus(b)
a – b	a.minus(b)
a * b	a.times(b)
a / b	a.div(b)
a % b	a.rem(b), a.mod(b)(deprecated)
a..b	a.rangeTo(b)

Expression	Translated to
a in b	b.contains(a)
a !in b	!b.contains(a)

감사합니다

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

윌리엄 셰익스피어
William Shakespeare