Lec 04

코틀린에서 연산자 를 다루는 방법

1. 단항 연산자 / 산술 연산자

▼ 단항 연산자 ⇒ 단항 연산자와 산술 연산자는 **Java, Kotlin 모두 동일**

- ++
- —

▼ 산술 연산자

- +
- -
- *
- /
- %

▼ 산술 대입 연산자

- +=
- -=
- *=
- /=
- %=

2. 비교 연산자와 동등성, 동일성

▼ 비교 연산자 ⇒ Java, Kotlin 사용법은 동일

- >
- <
- ≥
- ≤

⇒ Java와 다른 점 : 개체를 비교할 때 비교 연산자를 사용하면 자동으로 **compareTo()를 호출**

- 동등성과 동일성
  - 동등성(Equality) : 두 객체의 **값**이 같은가?
  - 동일성(Identity) : 완전히 동일한 객체인가? ⇒ **주소**가 같은가?
- 동등성 동일성 비교 예시

	1	2	3
값	JavaMoney (1000)	JavaMoney (1000)	JavaMoney (1000)
주소	0x101	0x101	0x201

  - 1번 객체와 2번 객체 : 정체성이 동일, Java에서 == 을 사용
  - 2번 객체와 3번 객체 : 값이 동등, Java에서 equals를 사용
- Java와 Kotlin에서의 동등성과 동일성

	Java	Kotlin
동일성	**==**	**===**
동등성	**equals** 직접 호출	**==** 간접적으로 equals 호출

3. 논리 연산자와 Kotlin에 있는 특이한 연산자

▼ 논리 연산자 ⇒ Java와 완전히 동일, Java처럼 **Lazy연산** 수행

- &&
- ||
- !

⇒ Lazy 연산

- {true} || { } → 무조건 true ⇒ 뒷 문장(코드) 실행하지 않음
- {false} && { } → 무조건 false ⇒ 뒷 문장(코드) 실행하지 않음

- 특이한 연산자
  - in : 컬렉션이나 범위에 포함 되어 있는가
    - 예시

```
println(1 in numbers)
```

  - !in : 컬렉션이나 범위에 포함 되어 있지 않은가
  - a..b : a부터 b 까지의 범위 객체 생성
  - a[i] : a에서 특정 index i로 값을 가져옴
    - 예시

```
val str = "ABC"
println(str[2])  // 두 번째 인덱스에 있는 C 반환
```

  - a[i] = b : a의 특정 index i에 b를 넣음

---

4. 연산자 오버로딩

- Kotlin에서는 객체마다 연산자를 직접 정의할 수 있음

```
val money1 = Money(1_000L)
val money1 = Money(2_000L)
println(money1 + money2)   // Money(amount = 3000)
```

📌 **총정리**

- 단항 연산자, 산술 연산자, 산술 대입연산자는 Java와 Kotlin 모두 동일
- 비교 연산자의 사용법은 Java와 동일
  - 단, 객체끼리도 자동 호출되는 compareTo를 이용해 비교 연산자 사용
- in, !in / a..b / a[i] / a[i] = b 와 같이 Kotlin에서 새로 생긴 연산자도 있음
- 객체끼리의 연산자를 직접 정의할 수 있음