💛

Lec 03

`코틀린에서 Type을 다루는 방법`

0. 복습

- 키워드
    1. var, val
    2. val x**: Int**
    3. 참조 타입(reference type)
    4. nullable
        - ex) val x : Int**?**
    5. Person person = new Person(); ⇒ Java
        - val person = Person() ⇒ Kotlin
    6. Safe Call
        - 변수명?.( )
    7. Elvis 연산자
        - (null이 들어올 수 있는 문장)?: (null이 들어왔을 때 반환할 값)

8. 널 아님 단언!!
    - 변수!!

---

1. 기본 타입

- *기본 타입 : Byte, Short, Int, Long, Float, Double, 부호 없는 정수들*
    - *Int, Long, Float, Double를 가장 많이 사용*

▼ Kotlin에서는 선언된 기본값을 보고 **타입을 추론**함

- 예시

```
val number1 = 3.0f  // Float
val number2 = 3.0   // Double
```

👉 Java와 다른 내용

- Java ⇒ 기본 타입간의 변환은 **암시적**으로 이루어질 수 있음
- Kotlin ⇒ 기본 타입간의 변환은 **명시적**으로 이루어져야 함

▼ Java 타입변환 예시

- 자동

  → long x = 10;

- 강제

  → int x = (int)10L;

Kotlin에서이 타입변환

- 명시적으로 이루어져야 함
- **to변환타입()** 사용

▼ 예시

▼ (1)

```
val number1: Int = 4
val number2: Long = number1.toLong()

println(number1 + number2)
```

▼ (2)

```
val number1 = 3
val number2 = 5
val result = number1 / number2.toDouble()

println(result)
```

▼ (3)

```
val number1: Int = 4
val number2: Long = number1.toLong()

println(number1 + number2)
```

---

2. 타입 캐스팅

```
public static void printAgeIfPerson(Object obj) {
    if(obj instanceof Person) {
        Person person = (Person) obj;
        System.out.println(person.getAge());
    }
}
```

- instanceof : 변수가 주어진 타입이면 true, 그렇지 않으면 false
- (타입) : 주어진 변수를 해당 타입으로 변경 ⇒ 강제 타입 변환

```
// 위 코드 Kotlin으로 바꾸기
fun printAgeIfPerson(obj: Any) {
    if(obj is Person) {
        val person = obj as Person
        println(person.age)
    }
}
```

Java	Kotlin
intanceof	**is**
(Person) obj	obj **as** Person

- instanceof의 반대
  - value 가 type이면 false, 그렇지 않으면 true

Java	Kotlin
(!(객체 instanceof 타입))	(객체 !is 타입)

```
fun printAgeIfPerson(obj: Any) {
    if (obj is Person) {
        printfln(obj.age).   //스마트 캐스트
    }
}
```

- 만약 객체에 null이 들어올 수 있는 경우

▼ is / !is

- value **is** type

  → value가 type이면 true, 그렇지 않으면 false

- value **!is** type

  → value가 type이면 false, 그렇지 않으면 true

▼ as / as?

- value **as** type

  → value가 type이면 type으로 타입캐스팅

  → 그렇지 않으면 예외발생

- value **as?** type

  → value가 type이면 type으로 타입캐스팅

  → value가 null이면 null

  → value가 type이 아니면 null

---

3. Kotlin의 특이한 타입

- Any, Unit, Nothing

Any

- **Java의 Object**역할 - 모든 객체의 최상위 타입
- 모든 Primitive Type(기본 타입)의 최상위 타입
- Any 자체로는 null을 표현할 수 없음, null을 표현하고 싶다면 Any?로 표현
- Any에 equals / hashCode / toString 존재

Unit

- **Java의 void**역할
- void와 다르게 Unit은 그 자체로 타입 인자로 사용 가능
  - (Java) void x; → 불가능

    (Kotlin) Unit x → 가능
- 함수형 프로그래밍에서 Unit은 단 하나의 객체(인스턴스)만 갖는 타입을 의미, 즉 코틀린의 Unit은 실제 존재하는 타입이라는 것을 표현

Nothing

- **함수가 정상적으로 끝나지 않았다는 사실을 표현**하는 역할
- 무조건 예외를 반환하는 함수 / 무한 루프 함수 등

```
fun fail(message: String): Nothing {
    throw IllegalArgumentException(message)
}
```

---

4. String interpolation / String indexing

- ${변수}

  ```
  val person = Person("OOO", 100)
  val log = "사람의 이름은 ${person.name}이고, 나이는 ${person.age}
  ```

  - `${변수}`를 사용하면 값이 들어감

    (변수가 들어갈 자리에 식도 들어갈 수 있음)

- $변수

  ```
  val name = "OOO"
  val age = 100
  val log = "사람의 이름 : $name 나이 : $age"
  ```

  - `$변수`를 사용할 수도 있음
  - 변수 이름만 사용하더라도 **${변수}**를 사용하는 것이 가독성, 일괄 변환, 정규식 활용 측면에서 좋음

- “”” (내용) “””

  ```
  val withoutIndent =
  """
      ABC
      123
      456
  """.trimIndent()
  println(withoutIndent)
  ```

    - “””(내용)”””을 사용하면 문자열을 여러 줄 작성할 수 있음
- Java에서 특정 문자열 가져오기

```
String str = "ABCDE";
char ch = str.charAt(1);

// -> str의 1번째 인덱스에 있는 B를 반환
```

- Kotlin에서 특정 문자열 가져오기

```
val str = "ABCDE"
val ch = str[1]

// -> str의 1번째 인덱스에 있는 B를 반환
```

📌 **총정리**
- 코틀린의 변수는 초기값을 보고 타입을 추론하며, 기본 타입들 간의 변환은 명시적으로 이루어짐
- 코틀린에서는 is, !is, as, as?를 이용해 타입을 확인하고 캐스팅함
** - is, !is : instanceof**
** - as, as? : 강제 타입 변환**
- Kotlin의 Any는 Java의 Object와 같은 최상위 타입
- Kotlin의 Unit은 Java의 void와 동일
- Kotlin의 Nothing은 정상적으로 끝나지 않는 함수의 반환을 의미
- 문자열을 가공할 때 ${변수}와 “”” “””을 사용하면 깔끔하게 코드 작성 가능
- Kotlin의 문자열에서 특정 문자를 가져올 때 Java의 배열 처럼 [ ]를 사용