🧡

Lec 02

`코틀린에서 null를 다루는 방법`

1. Kotlin에서 null 체크

```
// Java
public boolean startsWithA(String str) {
    return str.startsWith("A");
}
```

▼ 위 코드가 불안정한 코드인 이유 ⇒ str에 null이 들어오면 NPE 발생

- NPE : Null Pointer Exception
  프로그램이 null 값으로 초기화되지 않은 객체를 사용하려고 할 때 발생하는 예외

▼ Java에서 null

▼ startsWithA1

```
// Java
public boolean startsWithA1(String str) {
    if(str == null) {
        throw new IllegalArgumentException("null이 들어왔습니
    }
    return str.startsWith("A");
}
```

```
// (1) str이 null일 경우 Exception을 냄
```

▼ startsWithA2

```
// Java
public boolean startsWithA2(String str) {
    if(str == null) {
        return null;
    }
    return str.startsWith("A");
}


// (2) str이 null일 경우 null 반환
```

▼ startsWithA3

```
// Java
public boolean startsWithA3(String str) {
    if(str == null) {
        return false;
    }
    return str.startsWith("A");
}


// (3) str이 null일 경우 false 반환
```

▼ Java와 Kotlin 비교

▼ (1)

```
// Java
public boolean startsWithA1(String str) {
    if(str == null) {
```

```
        throw new IllegalArgumentException("null이 들어왔습니
    }
    return str.startsWith("A");
}
```

```
// Kotlin
fun startsWithA1(str: String?): Boolean {
    if(str == null) {
        throw IllegalArgumentException("null이 들어왔습니다")
    }
    return str.startsWtih("A");
}
```

▼ (2)

```
// Java
public boolean startsWithA2(String str) {
    if(str == null) {
        throw null;
    }
    return str.startsWith("A");
}
```

```
// Kotlin
fun startsWithA2(str: String?): Boolean {
    if(str == null) {
        throw null
    }
    return str.startsWtih("A");
}
```

▼ (3)

```java
// Java
public boolean startsWithA3(String str) {
    if(str == null) {
        throw new IllegalArgumentException("null이 들어왔습니
    }
    return str.startsWith("A");
}
```

```kotlin
// Kotlin
fun startsWithA3(str: String?): Boolean {
    if(str == null) {
        throw IllegalArgumentException("null이 들어왔습니다")
    }
    return str.startsWtih("A");
}
```

💡 Kotlin에서는 null이 가능한 타입과 가능하지 않은 타입을 완전히 다르게 취급

---

2. Safe Call과 Elvis 연산자

👉 **Safe Call**(?.)

- null이 아니면 실행하고 null이면 실행하지 않음(그대로 null)

▼ Safe Call

```kotlin
val str: String? = "ABC"
str.length   // 불가능
str?.length. // 가능
```

👉 **Elvis 연산자**(?:)

- 앞의 연산 결과가 null이면 뒤의 값 사용

▼ Elvis 연산자

```
val str: String? = "ABC"
str?.length?: 0
```

▼ 예시

▼ (1)

```
fun startsWithA1(str: String): Boolean {
    return str?.startsWith("A")?: throw IllegalArgumentExc
}
```

- 주어진 문자열이 “A”로 시작하면 true 반환
- 주어진 문자열이 “A”로 시작하지 않는 경우 false 반환
- 주어진 문자열이 null인 경우 예외 발생

▼ (2)

```
fun startsWithA2(str: String?): Boolean? {
    return str?.startsWith("A")
}
```

- 주어진 문자열이 null이 아닌 경우에만 문자열이 “A”로 시작하는지 확인
  - 주어진 문자열이 “A”로 시작하면 true, 그렇지 않으면 false 반환
- 주어진 문자열이 null인 경우 startsWith("A")는 실행되지 않고 null 반환

▼ (3)

```
fun startsWithA3(str: String?): Boolean{
    return str?.startsWith("A") ?: false
}
```

- 주어진 문자열이 null이 아닌 경우에만 문자열이 “A”로 시작하는지 확인
- 주어진 문자열이 null인 경우 startsWith("A")는 실행되지 않고 false 반환
- 반환타입이 nullable한 타입이 아니기 때문에 false로 기본값 지정

---

3. 널 아님 단언!!

> nullable이지만, 아무리 생각해도 null이 될 수 없는 경우사용

```
fun startsWithA1(str: String?): Boolean {
    return str!!.startsWith("A")
}
```

👉 혹시나 null이 들어오면 NPE가 발생하기 때문에 null이 아닌게 확실한 경우 널 아님 단언!!을 사용

📌 **총정리**
- 코틀린에서 null이 들어갈 수 있는 타입과 그렇지 않은 타입은 완전히 다르게 간주됨
** - 한 번 null 검사를 하면 non-null임을 컴파일러가 알 수 있음**
- null이 아닌 경우에만 호출되는 Safe Call(?.)이 있음
- null인 경우에만 호출되는 Elvis 연산자(?:)가 있음
- null이 절대 아닐 때 사용할 수 있는 널 아님 단언(!!)이 있음
- Kotlin에서 Java 코드를 사용할 때 플랫폼 타입 사용에 유의해야 함
** - Java 코드를 읽으며 널 가능성 확인 / Kotlin으로 wrapping**