Null safety in Kotlin

Faragó János

Null pointer exceptions

- In Java every reference can be null (Hoare Billion Dollar Mistake)
- In Kotlin there are optional type

```
val nullableString: String? = “Optional”
```

- In Kotlin you cannot access nullable variables without checking it is not null (or asserting)

How to check nullity

- If statement
- If expression
- Safe call
- Elvis operator
- Sure operator

Smart cast

- Compiler does it, when you check nullability

```
if (nullableString is String)  {
     length = nullableString.length
}
```

- Within statement

```
when (nullableString)  {
     null -> length = -1
     is String -> length = nullableString.length
     else -> length = nullableString.length
}
```

If statement

- Like in Java
- Smart casted to non optional

```
if (nullableString != null)  {
     length = nullableString.length
}
else {
     length = -1
}
```

Safe call

- Returns the value
- Or null
- Iterate over lists
- Let function (special case)

```
val length = nullableString?.length
```

```
list?.forEach {
    print(it)
}
```

```
nullableString?.let {
    it.length
}
```

Elvis operator

- Provides an alternate value

```
val length = nullableString?.length ?: -1
```

- Can be used to throw exceptions or early return as well

```
val length = nullableString?.length ?: return
```

If expression

- Smart cast
- Like Elvis operator

```
val length: Int = if (nullableString != null) {
     nullableString.length
}
else {
     -1
}
```

Sure operator

- Throws a NullPointerException
- An assert followed by a smart cast

```
val length = nullableString!!.length
```

Kotlin library calls

- String?.isNullOrEmpty()
- String?.isNullOrBlank()
- filterNotNull()
- emptyList() emptySet()
- getOrElse()

From now on,

L👀k out for your Nulls!

Thank you, your attention is appreciated!