Kotlin Null

Sources: http://kotlinlang.org/docs/reference/basic-syntax.html
http://petersommerhoff.com/dev/kotlin/kotlin-for-java-devs/

Null Safety

In Kotlin, the type system distinguishes between references that can hold ***null (nullable references)*** *and those that* ***cannot (non-null references).***

The Kotlin compiler makes sure you don't, by accident, operate on a variable that is null.

Null Safety – a non-null reference

A regular variable of type String can not hold null

```
var a: String = "abc"
a = null // syntax error
```

Calling a method / accessing a property on variable a, is guaranteed not to cause an NullPointerException

```
val l = a.length
```

Null Safety – a nullable reference

To allow nulls, we can declare a variable as nullable string, written String?

```
var b: String? = "abc"
b = null // ok
```

```
val l = b.length    // syntax error: variable
                    //'b' can be null

                    // …many ways around this…
```

Null Safety – a nullable reference

To allow nulls, we can declare a variable as nullable string, written String?

```
var b: String? = "abc"
b = null // ok
```

Option 1: you can explicitly check if b is null, and handle the two options separately:

```
val l = if (b != null) b.length else -1
```

Null Safety – a nullable reference

To allow nulls, we can declare a variable as nullable string, written String?

```
var b: String? = "abc"
b = null // ok
```

Option 2: you can use the safe call operator ?.
This returns **b.length** if **b** is not null, and null otherwise.

```
b?.length
```

Null Safety – a nullable reference

To allow nulls, we can declare a variable as nullable string, written String?

```
var b: String? = "abc"
b = null // ok
```

Option 3: you can use the **!!** Operator. This force a call to our method and will return a non-null value of **b** or throw an NPE if **b** is null. Use sparingly!

```
val l = b!!.length
```

Null Safety – The Elvis Operator, ?:

When we have a nullable reference **r**, we can say:

"if **r** is not null, use it, otherwise use some non-null value **x**"

```
val l: Int = if (b != null) b.length else -1
```

Null Safety – The Elvis Operator, ?:

When we have a nullable reference **r**, we can say:

"if **r** is not null, use it, otherwise use some non-null value **x**"

```
val l: Int = if (b != null) b.length else -1
```

Along with the complete if-expression, this can be expressed with the Elvis operator, written **?:**

```
val l = b?.length ?: -1
```

If the expression to the left of **?:** is not null, the elvis operator returns it, otherwise it returns the expression to the right.

Nullable – nullable returns

A reference must be explicitly marked as nullable (i.e. ?) when null value is possible.

Return null if the return value does not hold an integer:

```
fun parseInt(str: String): Int? {
    // ...
}
```

Type Checks & Casts

is and !is operators

```
fun main(args: Array<String>) {
   val aString = "I am a String"

   if (aString is String) {
      println("String length is: ${aString.length}")
   }

   if (aString !is String) { // same as !(aString is String)
      print("Not a String")
   }
   else {
      println("String length is: ${aString.length}")
   }
}
```

Console

```
<terminated> Config - Main.kt [Java Application] C:\Program Files\Java\jre1.8.0_77\
String length is: 13
String length is: 13
```

Smart Casts (an example using if)

```
fun main(args: Array<String>) {
     demo ("I am a String")
     demo (12)
}

fun demo(x: Any) {
    if (x is String) {
        println(x.length) // x is automatically cast to String
    }
    else{
        println(x.javaClass)
    }
}
```

Console

```
<terminated> Config - Main.kt [Java Application] C:\Program Files\Java\jre1.8.0_77\bin\javav
13
class java.lang.Integer
```

Smart Casts (an example using when)

```
fun main(args: Array<String>) {
    demo (12)
    demo ("I am a String")
    demo (intArrayOf(1,2,3,4))
}

fun demo(x: Any) {
    when (x) {
        is Int -> println(x + 1)
        is String -> println(x.length + 1)
        is IntArray -> println(x.sum())
    }
}
```

Console

```
<terminated> Config - Main.kt [Java Application] C:\P
13
14
10
```