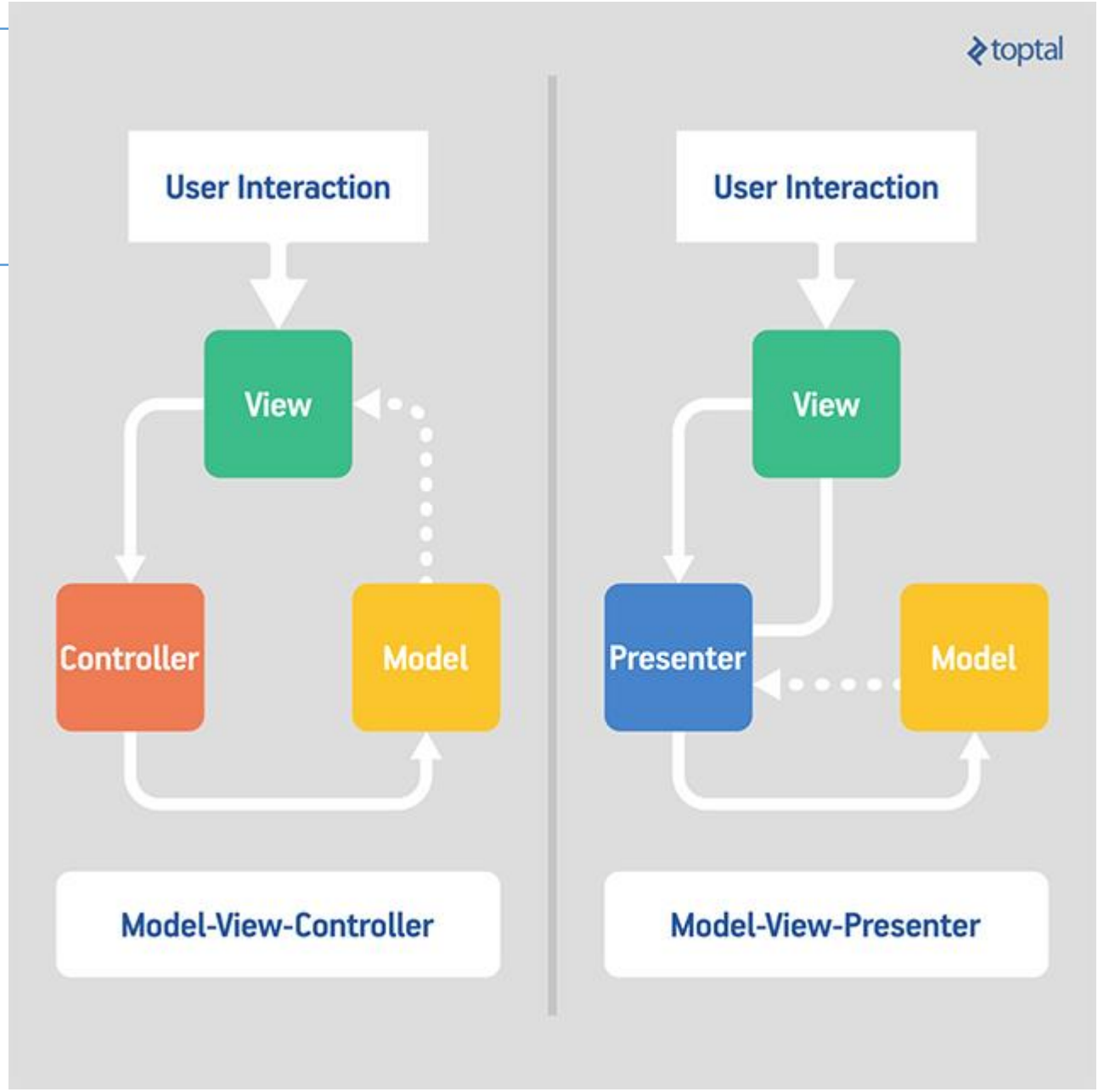
Kotlin

Desarrollo de aplicaciones en Android

Kotlin

Características

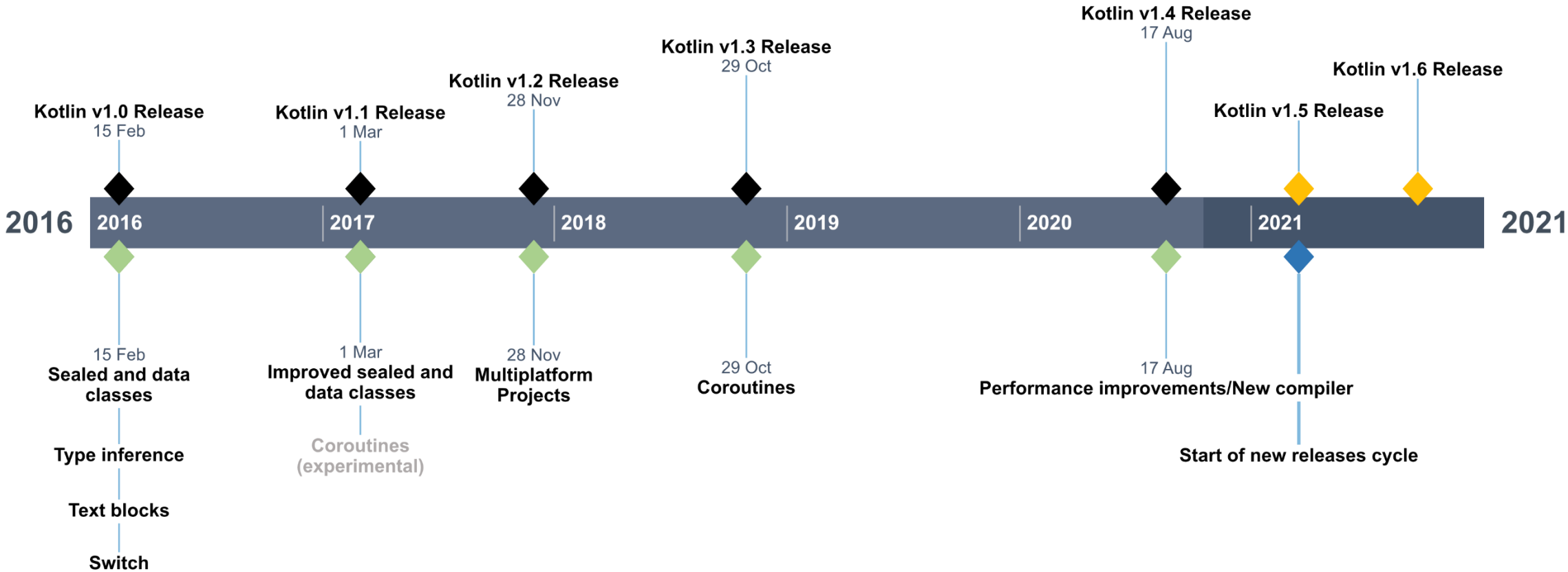
- Menos código
- Lenguaje maduro.
  - Desarrollado en 2011
  - Integrado en Android Studio
- Soporte de diferentes librerías
  - Extensiones KTX
  - Coroutines
  - Lambas
  - ….
- Interoperable con Java
- Multiplataforma
  - Android / IOS / backend web…
- Código seguro



Versiones

- Kotlin 1.7.20 (versión actual)
  - Mejora en compilación. Soporte de compiladores all-open
  - Operador ..< para crear open-endend ranges
  - Gestor de memoria nativa de Kotlin habilitada por defecto
- Kotlin 1.7.0

Kotlin Release cycle



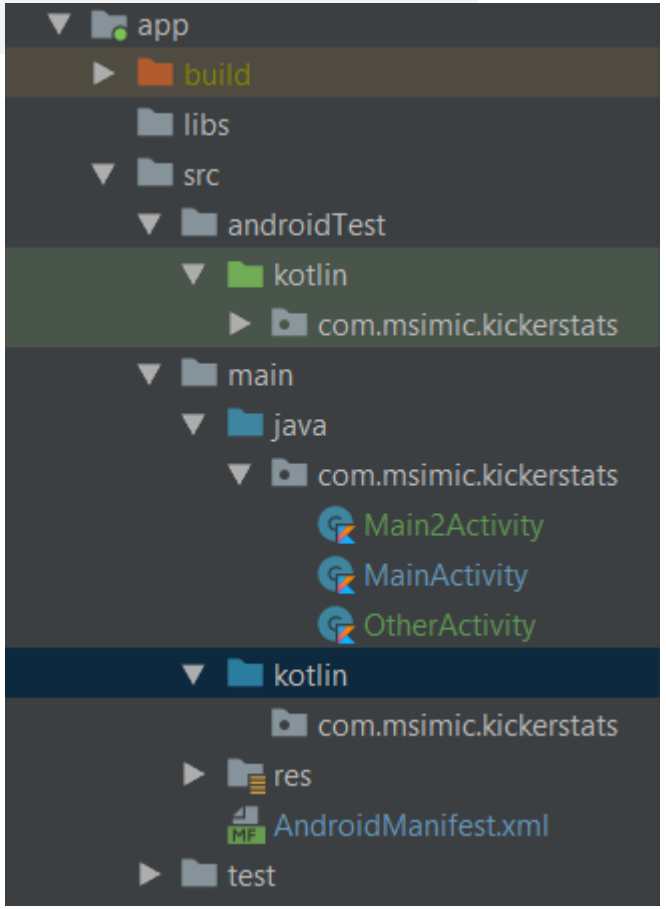
Package

- La definición del paquete se puede incluir al inicio del archivo
- Para importar librerías, como en Java, utilizamos la notación del punto
- * importa el total de la librería

```
package my.demo

import kotlin.text.*

// ...
```



Funciones

- Se puede indicar el tipo de dato de los parámetros.
- También la función puede tener un tipo de dato de retorno
- El tipo de retorno también puede ser inferido

```
fun sum(a: Int, b: Int): Int {
    return a + b
}
```

```
fun sum(a: Int, b: Int) = a + b
```

Variables

- La variables de sólo lectura se declaran usando val
- Kotlin puede detectar el tipo de dato inferido al declarar el valor
- Var permite declarar variables que pueden ser reasignadas

```
var x = 5 // `Int` type is inferred
x += 1
```



```
val a: Int = 1  // immediate assignment
val b = 2   // `Int` type is inferred
val c: Int  // Type required when no initializer is provided
c = 3       // deferred assignment
```

Clases

```
class Rectangle(var height: Double, var length: Double) {
    var perimeter = (height + length) * 2
}
```

- Las clases se declaran con la palabra clave class
- Las propiedades de una clase se pueden indicar en su declaración o en el cuerpo de la clase
- Al instanciar la clase NO es necesario utilizar new

```
class Greeter(val name: String) {
    fun greet() {
        println("Hello, $name")
    }
}

fun main(args: Array<String>) {
    Greeter(args[0]).greet()
}
```

```
val rectangle = Rectangle(5.0, 2.0)
println("The perimeter is ${rectangle.perimeter}")
```

Condicionales

```
fun maxOf(a: Int, b: Int): Int {
    if (a > b) {
        return a
    } else {
        return b
    }
}
```

```
fun describe(obj: Any): String =
    when (obj) {
        1          -> "One"
        "Hello"    -> "Greeting"
        is Long    -> "Long"
        !is String -> "Not a string"
        else       -> "Unknown"
    }
```

```
fun maxOf(a: Int, b: Int) = if (a > b) a else b
```

Bucles

```
val items = listOf("apple", "banana", "kiwifruit")
var index = 0
while (index < items.size) {
    println("item at $index is ${items[index]}")
    index++
}
```

```
val items = listOf("apple", "banana", "kiwifruit"
for (item in items) {
    println(item)
}
```

```
val x = 10
val y = 9
if (x in 1..y+1) {
    println("fits in range")
}
```

```
val items = listOf("apple", "banana", "kiwifruit")
for (index in items.indices) {
    println("item at $index is ${items[index]}")
}
```

Lambdas

```
val fruits = listOf("banana", "avocado", "apple", "kiwifruit")
fruits
    .filter { it.startsWith("a") }
    .sortedBy { it }
    .map { it.uppercase() }
    .forEach { println(it) }
```

Valores nulos

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

- Una referencia debe marcarse como nullable si el valor null es posible.
  - Los valores nullable tienen ? Al final
- El operador is chequea si una variable es de un determinado tipo

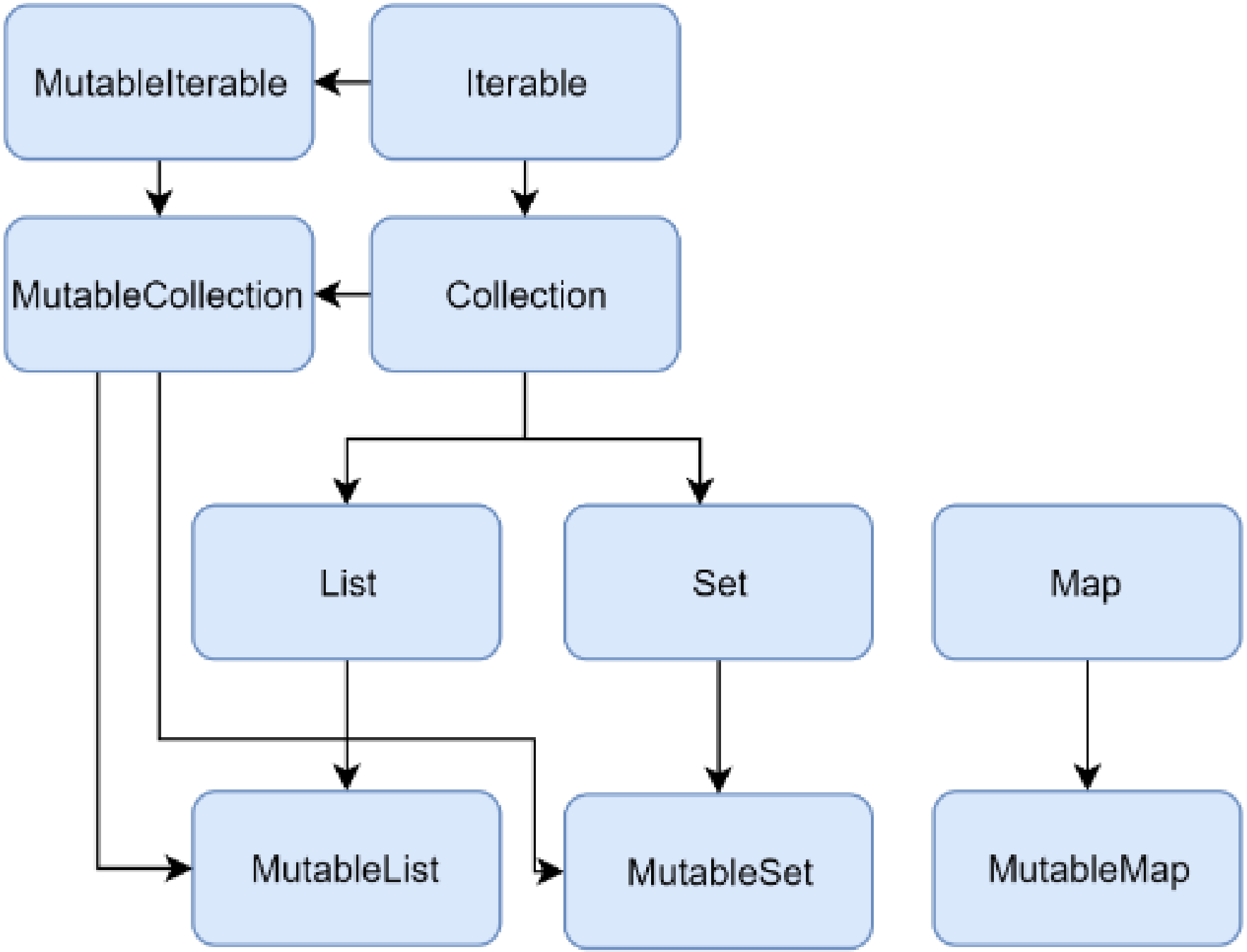
```
fun getStringLength(obj: Any): Int? {
    if (obj is String) {
        // `obj` is automatically cast to `String` in this branch
        return obj.length
    }

    // `obj` is still of type `Any` outside of the type-checked branch
    return null
}
```

Collections

- La librería standard de kotlin ofrece un conjunto de herramientas para gestionar collections
  - Grupos de un número variable de elementos que tienen características en común

- *List. Es una colección ordenada con acceso a sus elementos por índice. Soporta repetición. Un ejemplo sería un número de teléfono.*
- *Set. Colección de elementos únicos sin repetición. No es relevante su orden. Un ejemplo sería el número de un décimo de lotería.*
- *Map. También llamado dictionary. Es un conjunto de key-value. Los keys son únicos y cada uno "mapea" un único value. Los values soportan duplicados. Un ejemplo sería el id de alumno y su nota.*


MutableIterable
Iterable
MutableCollection
Collection
List
Set
Map
MutableList
MutableSet
MutableMap

```
val numbers = mutableListOf("one", "two", "three", "four")
numbers.add("five")   // this is OK
println(numbers)
//numbers = mutableListOf("six", "seven")      // compilation error
```

```
val numbers = listOf("one", "two", "three", "four")
println("Number of elements: ${numbers.size}")
println("Third element: ${numbers.get(2)}")
println("Fourth element: ${numbers[3]}")
println("Index of element \"two\" ${numbers.indexOf("two")}")
```

List

```
val numbers = mutableListOf(1, 2, 3, 4)
numbers.add(5)
numbers.removeAt(1)
numbers[0] = 0
numbers.shuffle()
println(numbers)
```

List Mutable

```
val numbers = setOf(1, 2, 3, 4)
println("Number of elements: ${numbers.size}")
if (numbers.contains(1)) println("1 is in the set")

val numbersBackwards = setOf(4, 3, 2, 1)
println("The sets are equal: ${numbers == numbersBackwards}")
```

Set

Linked Set

```
val numbers = setOf(1, 2, 3, 4)  // LinkedHashSet is the default
val numbersBackwards = setOf(4, 3, 2, 1)

println(numbers.first() == numbersBackwards.first())
println(numbers.first() == numbersBackwards.last())
```

```
val numbersMap = mapOf("key1" to 1, "key2" to 2, "key3" to 3, "key4" to 1)

println("All keys: ${numbersMap.keys}")
println("All values: ${numbersMap.values}")
if ("key2" in numbersMap) println("Value by key \"key2\": ${numbersMap["key2"]
if (1 in numbersMap.values) println("The value 1 is in the map")
if (numbersMap.containsValue(1)) println("The value 1 is in the map") // same
```

Map

```
val numbersMap = mutableMapOf("one" to 1, "two" to 2)
numbersMap.put("three", 3)
numbersMap["one"] = 11

println(numbersMap)
```

Mutable Map

Listas

archivo.kt

```kotlin
var mutableList: MutableList<String> = mutableListOf("Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado")

println(readOnly) //[Lunes, Martes, Miércoles, Jueves, Viernes, Sábado]

mutableList.add("domingo")
```

Recorrer listas

```kotlin
for (item in mutableList) {
        print(item)
    }
```

```kotlin
for ((indice, item) in mutableList.withIndex()) {
    println("La posición $indice contiene $item")
}
```

Arrays

archivo.kt

```
Search (Ctrl+Shift+F)
(args: Array<String>) {
    val weekDays = arrayOf("Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo")
        println(weekDays.get(0))
        println(weekDays.get(1))
        println(weekDays.get(2))
        println(weekDays.get(3))
        println(weekDays.get(4))
        println(weekDays.get(5))
        println(weekDays.get(6))
    }
```

```
12  weekDays.set(0, "lunes guay") //Contenía Lunes
13  weekDays.set(4, "viernes modificado") //Contenía Viernes
```

```
for (posicion in weekDays.indices){
    println(weekDays.get(posicion))

}
```

Recorrer arrays