Objetos y Clases

En kotlin todo es un objeto, los datos Int realmente son objetos, las funciones son objetos, etc. Por lo tanto, ya estuvimos utilizando objetos, objetos que pertenecen a clases del sistema Kotlin. En este cuaderno nos introducimos al uso de objetos que son instancias de clases escritas por nosotros mismos.

Una primera visión de lo que es un objeto

Los objetos almacenan datos usando para ello propiedades val/var y pueden contener en su interior definición de operaciones que probablemente utilizan los datos anteriores para hacer algo.

Algunas definiciones para ir arrancando:

- Una clase: Define en su interior propiedades y funciones. A las clases también se les llama tipos datos definidos por el usuario. Así tenemos los tipos propios del lenguaje como Int y String y los que va definiendo en sus aplicaciones el usuario como la clase Coche, Persona etc..
- Miembro: Una clase contiene en su interior miembros. Hay dos tipos principales de miembros a los que ya aludimos: propiedades y y funciones.
- En kotlin una función se puede definir dentro de una clase o fuera de toda clase y según este criterio hay dos tipos de funciones
  - funciones top-level. Las que se escriben directamente en el fichero "al top level del fichero" fuera de toda clase.
  - funciones miembro. Se escriben dentro de una clase y su funcionamiento está ligado a un objeto específico de la clase.
- Crear un objeto: Hacer un val o var de una clase. A los objetos también se les llama instancia de una clase.

DEFINICIÓN DE UNA CLASE

En general la fexibilidad de Kotlin redunda en multitud de posibilidades sintácticas para escribir lo mismo. Aquí recogemos sólo las posibilidades más fundamentales.

Una clase es una plantilla o modelo que se utilizará para crear objetos. Una clase de Kotlin se define usando la palabra clave *class.* El cuerpo de una clase puede contener propiedades y/o funciones miembro. En otro cuaderno afinaremos el concepto de *propiedad (property)*. Como punto de partida podemos ver una propiedad como una variable que pertenece a la clase.

La declaración básica de un clase consta del nombre de la clase y el cuerpo de la clase rodeado de llaves.

```
class Persona{
  var nombre = ""
  var edad = 0
  fun printMe() {
      print(nombre+ " " + edad)
   }
}
```

Tanto el encabezado como el cuerpo son opcionales; si la clase no tiene cuerpo, se pueden omitir las llaves. La siguiente es una declaración válida de clase. Una clase vacia que se define sólo con la palabra reservada class seguida de un nombre.

```
class miClaseVacia
```

CREACIÓN DE OBJETOS

Los objetos se crean a partir una clase que funciona a modo de plantilla o molde. *"El molde"* describe propiedades y comportamientos. Por lo tanto, todos los objetos de una clase tendrán la misma estructura de propiedades y comportamientos.

La sintaxis más básica para crear un objeto de una clase es:

```
var varName = ClassName()
```

Podemos acceder a las propiedades y métodos de una clase usando el operador punto "."

var varName = ClassName()

varName.property = valor

In [ ]:
```
class Persona{
  var nombre = ""
  var edad = 0
  fun printMe() {
      print(nombre+ " " + edad)
   }
}
val p = Persona()
p.nombre="yo"
p.edad=14
p.printMe()
```

La referencia this

Es una variable especial que se utiliza para referenciar al propio objeto dentro del propio objeto. Se puede usar la variable this para referenciar a los miembros de un objeto desde dentro de ese objeto. El siguiente ejemplo es equivalente al anterior, la única diferencia es que desde printMe() para aludir a los propiedades también usamos la referencia this.

In [1]:
```
class Persona{
  var nombre = ""
  var edad = 0
  fun printMe() {
      print(this.nombre+ " " + this.edad)
   }
}
val p = Persona()
p.nombre="yo"
p.edad=14
p.printMe()
```

```
yo 14
```

Usar o no usar this

Como observaste, los dos últimos ejemplos son equivalentes., entonces, ¿se debe usar this o no?. Es una cuestión de estilo, no de compilación y en el caso del lenguaje Kotlin, cuando this no es necesario, se prefiere no usarlo en aras de la limpieza y concisión.

Y dicho esto, hay diversas situaciones en que el uso de this es necesario y las iremos examinando a medida que lo requiramos.