KOTLIN Y LA POO

Como ya deberías saber, **POO** significa **Programación Orientada a Objetos**. No vamos a extendernos demasiado en esto porque ya habéis trabajado con Java, que también es POO. Digamos que la programación orientada a objetos consiste en crear objetos que contienen tanto los datos como los métodos de un programa.

Todo en **Kotlin** son clases y objetos. En general, funciona igual que en Java, aunque a diferencia de Java, no existen los tipos de datos primitivos, por lo que siempre es necesario reservar memoria para un objeto. Crear una clase en Kotlin es sencillo, puesto que sólo se necesita utilizar la palabra reservada **class**.

```
class Car {
  var brand = ""
  var model = ""
  var year = 0
}
```

Como ves, faltan la mayoría de las palabras reservadas usadas en Java. En Kotlin no son necesarias. Todas las propiedades y funciones son públicas a menos que especifiques lo contrario. No hay mucho más que contar respecto a atributos y funciones.

Al igual que Java, Kotlin utiliza **constructores** para inicializar objetos. Lo que pasa es que se escriben de una forma 'especial'. Se definen mediante el uso de dos paréntesis () después del nombre de la clase, donde especificas las propiedades. El constructor inicializará las propiedades cuando crees un objeto de una clase.

```
class Car(var brand: String, var model: String, var year: Int)
```

Finalmente, Kotlin también dispone de mecanismos para implementar la **herencia**. Funciona de forma similar a Java, dado que se heredan las propiedades y las funciones de la clase padre. Para indicar una herencia se utiliza ":"

```
class MyChildClass: MyParentClass() {
```

Lamentablemente, la cosa es un poquito más complicada que simplemente poner los dos puntitos. Por defecto, **Kotlin** asume que todas las clases, funciones y atributos son finales, es decir, que no puedes extender de ellos o sobrescribirlos (**override**). Para permitirlo, debemos marcar las clases, métodos o atributos como abiertas (**open**).

Por tanto, en el caso anterior, la clase MyParentClass debería ser declarada así:

```
open class MyParentClass {
  val x = 5
}
```

Bloque prácticas 2

Accede a la página del tutorial y revisa el segundo bloque **Kotlin Classes**. Estos puntos te resumen los elementos básicos de la POO con Kotlin.

A continuación, lee y ejecuta las clases **Main09.kt** hasta **Main012.kt** para información adicional y ver ejemplos funcionales. Prueba tú mismo a crear tus propios ejemplos.

Ejercicio 6

Igual que en el Ejercicio 4, pero en este caso en lugar de nombres tenemos Alumnos. Los Alumnos tienen:

- Nombre
- Apellido 1
- Apellido 2
- Edad
- Curso

Ejercicio 7

Lo mismo, pero cada Alumno tiene una o más Notas:

- Asignatura
- Nota

Añade al menú las siguientes opciones:

1) Añadir nueva nota
2) Ver todas las notas de alumno
3) Buscar nota de asignatura
4) Modificar nota de asignatura
5) Borrar nota de asignatura

Ejercicio 8

Crea un programa en Kotlin que lea un fichero de texto y lo muestre por pantalla.

NOTA: El objetivo de este ejercicio es que entiendas que TODO está en internet. Si tienes claro lo que buscas, tardas tres minutos en solucionar el ejercicio. ¿Estamos de acuerdo? Bien, porque cuando empecemos con Android Studio y la programación en móviles, habrá muchas cosas que no te explicaré y tendrás que buscarte la vida. ¡Ánimo!