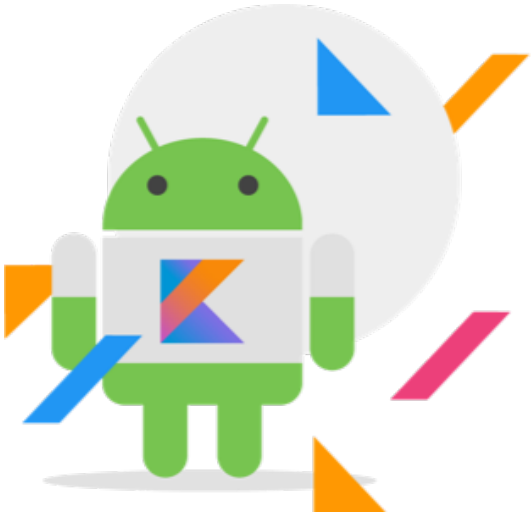
IES DR. LLUÍS SIMARRO

PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES

CURS 2025-2026

UD2: Fundamentos de Kotlin

Kotlin



Profesorado	
Juan Mateu	j.mateuruzafa@edu.gva.es
Joan Carrillo	jr.carrilloforonda@edu.gva.es
Ana Sanchis	ag.sanchisperales@edu.gva.es

UD02 -> T2_A1. FUNDAMENTOS DE KOTLIN

Instrucciones iniciales

- Las tareas se subirán a la plataforma Aules en la fecha establecida y con el formato indicado en los siguientes puntos.
- Para el desarrollo de las siguientes tareas, se creará un proyecto nuevo con Android Studio y se creará un fichero Kotlin para cada actividad.
- Cada actividad estará en un fichero Kotlin con nombre ActividadX (Actividad1, Actividad2…)

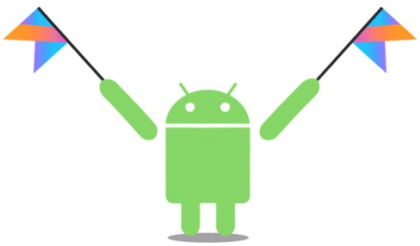
Objetivos de la actividad

1. Conocer la sintaxis del lenguaje de programación Kotlin.
2. Poner en práctica los diferentes paradigmas de programación estudiados (programación estructura, modular y POO).
3. Repasar el paradigma POO (Programación Orientada a Objetos) mediante el lenguaje de programación Kotlin.

Temporalización

La tarea tiene una estimación aproximada de tres sesiones lectivas de 55 minutos cada sesión.

FUNDAMENTOS DE KOTLIN



VARIABLES Y TIPOS DE VARIABLES

1. Definir una variable inmutable con el valor 50 que representa el lado de un cuadrado, en otras dos variables inmutables almacenar la superficie y el perímetro del cuadrado. Mostrar la superficie y el perímetro en la Consola.

2. Definir tres variables inmutables y cargar por asignación los pesos de tres personas con valores Float. Calcular el promedio de pesos de las personas

y mostrarlo. Mostrar también el promedio redondeado a 2 cifras.

3. Escribir un programa en el cual se ingresen cuatro números enteros, calcular e informar la suma de los dos primeros y el producto del tercero y el cuarto. Realizar los cálculos en el mismo print.

4. Se debe desarrollar un programa que pida el ingreso del precio de un artículo y la cantidad que lleva el cliente. Mostrar lo que debe abonar el comprador.

FUNCIONES

5. Desarrollar un programa con dos funciones. La primera que solicite el ingreso de un entero y muestre el cuadrado de dicho valor. La segunda que solicite la carga de dos valores y muestre el producto de estos. Llamar desde la main a ambas funciones.

6. Desarrollar una función que solicite la carga de tres valores y muestre el menor. Desde la función main del programa llamar 2 veces a dicha función (sin utilizar una estructura repetitiva)

7. En la función main solicitar que se ingrese una clave dos veces por teclado. Desarrollar una función que reciba dos String como parámetros y muestre un mensaje si las dos claves ingresadas son iguales o distintas.

8. Confeccionar una función que reciba tres enteros y los muestre ordenados de menor a mayor. En la función main solicitar la carga de 3 enteros por teclado y proceder a llamar a la primera función definida.

9. Elaborar una función que reciba tres enteros y nos retorne el valor promedio de los mismos. La impresión del promedio se realiza en el main.

10. Hacer una función que reciba una cantidad variable de enteros y retorne su suma. Utiliza vararg para implementarlo.

11. Confeccionar una función que reciba una serie de edades y nos retorne la cantidad que son mayores o iguales a 18 (como mínimo se envía un entero a la función). Utiliza vararg para implementarlo.

FUNCIONES INFIX

12. Crea una función Infix llamada "esMayorQue" que tome dos números como parámetros y devuelva true si el primero es mayor que el segundo, de lo contrario, devuelve false. Puedes utilizarla para comparar dos números.

ARRAYS

13. Cargar un array de 10 elementos de tipo entero y verificar posteriormente si el mismo está ordenado de menor a mayor.

14. Realizar un programa que pida la carga de dos arreglos numéricos enteros de 4 elementos. Obtener la suma de los dos arreglos elemento a elemento, dicho resultado guardarlo en un tercer arreglo del mismo tamaño.

15. Desarrollar un programa que permita ingresar un arreglo de n elementos, ingresar n por teclado. Elaborar dos funciones, una donde se cree y cargue el array y otra que sume todos sus elementos y retorne dicho valor al main donde se imprima.

WHEN

16. Ingresar 10 valores enteros por teclado. Contar cuántos de dichos valores ingresados fueron 0, y cuántos de valores diferentes a 0. Recuerda utilizar la cláusula when.

POO

17. Supongamos que estás creando un sistema para administrar una biblioteca de libros. Vamos a modelar esta situación utilizando POO:

Definición de Clases:

1. Crea una clase llamada Libro con las siguientes propiedades:
    - titulo (String): El título del libro.
    - autor (String): El autor del libro.
    - añoPublicacion (Int): El año de publicación del libro.
    - precio (Double): El precio del libro.
2. Sobreescribe el método toString para mostrar la información del libro.

Definición de Herencia:

1. Crea una subclase llamada LibroDigital que herede de la clase Libro. La clase LibroDigital debe tener una propiedad adicional:
    - formato (String): El formato del libro digital (por ejemplo, "PDF", "EPUB").

Definición de Polimorfismo:

1. Crea una función llamada calcularPrecioDescuento en la clase Libro que calcule el precio con un descuento del 10%. Sobrescribe esta función en la clase LibroDigital para aplicar un descuento del 20% en los libros digitales.

Creación de Objetos:

1. En la función main, crea varios objetos de tipo Libro y LibroDigital. Llama a la función calcularPrecioDescuento en cada uno de ellos y muestra el precio con descuento en la consola.

18. En este ejercicio, modelamos un sistema de manejo de tareas (to-do list) con diferentes categorías de tareas y funcionalidades para agregar, modificar y listar tareas.

Definición de **Enum** y **Data Class**:

1. Crea un enum llamado Prioridad que representa diferentes niveles de prioridad para las tareas (por ejemplo, ALTA, MEDIA, BAJA).

2. Crea un data class llamada "Tarea" con las siguientes propiedades:
   - nombre (String): El nombre de la tarea.
   - descripcion (String): Una descripción opcional de la tarea.
   - prioridad (Prioridad): La prioridad de la tarea.
   - completada (Boolean): Indica si la tarea está completada o no.

Funciones y colecciones:

1. Crea una función llamada "agregarTarea" que toma una lista mutable de tareas y una nueva tarea como parámetros, y agregue la tarea a la lista.

2. Crea una función llamada "modificarTarea" que toma una lista mutable de tareas, el nombre de la tarea que se desea modificar y se modifiquen los detalles de esa tarea.

3. Crea una función llamada "listarTareasPorPrioridad" que tome una lista de tareas y una prioridad como parámetros, y devuelva una lista de tareas con esa prioridad.

Funciones de alcance

1. Utiliza una función de alcance (apply o with) para crear e inicializar una lista mutable de tareas con algunas tareas de ejemplo.

2. Utiliza otra función de alcance para agregar, modificar y listar tareas en la lista que creaste en el paso anterior.