



GOBIERNO DE  
MÉXICO

EDUCACIÓN  
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO  
NACIONAL DE MÉXICO



## TECNOLOGICO NACIONAL DE MEXICO INSTITUTO TECNOLOGICO DE CIUDAD MADERO

Carrera: Ingeniería en Sistemas Computacionales.

Materia: Programación Nativa para Móviles.

Maestro: Jorge Peralta Escobar.

Integrantes del equipo:

Karla Denisse Cruz Solís #21070310

Yahir Osvaldo Valero Hernández #21070330

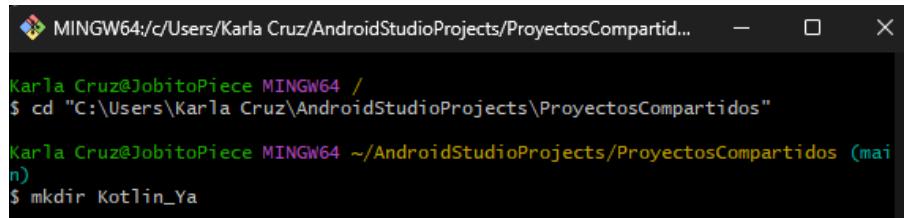
Grupo: A

Hora: 09:00 – 10:00

Semestre: Enero - Junio 2025.

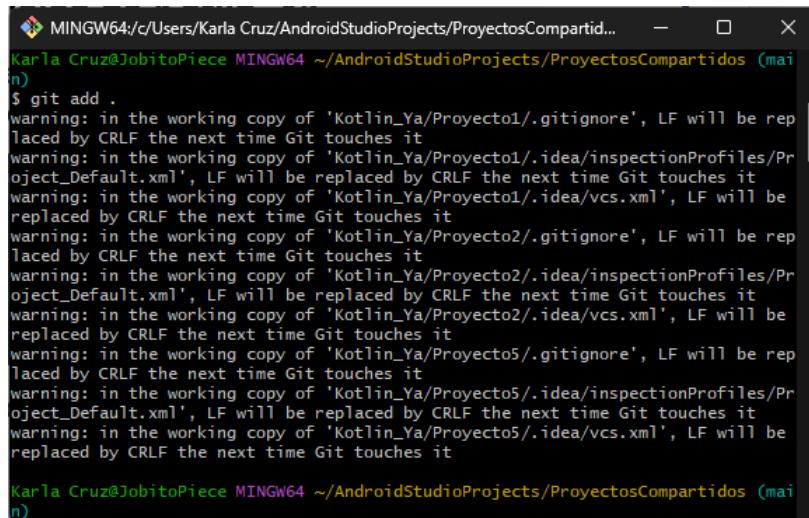
## Documentación de ejercicios de Kotlin\_Ya

Para estos ejercicios fue necesario crear la carpeta correspondiente dentro de nuestro repositorio, para lo cual fue necesario el comando `mkdir Kotlin_Ya`:

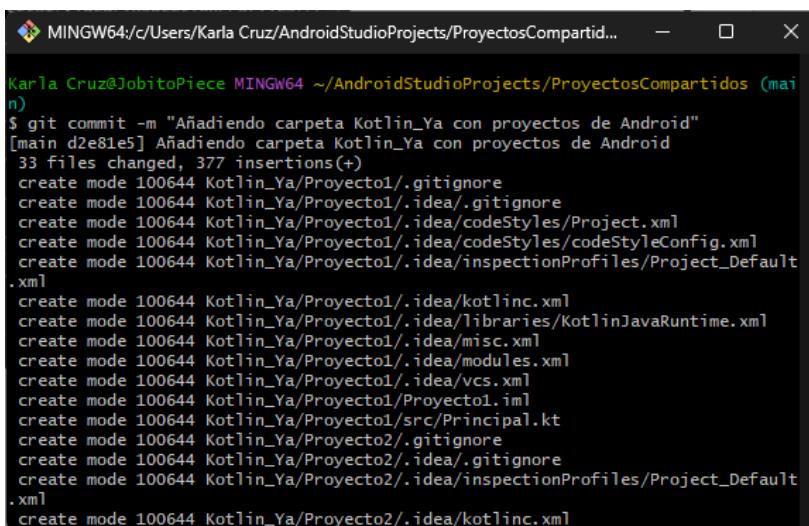


```
MINGW64:/c/Users/Karla Cruz/AndroidStudioProjects/ProyectosCompartidos... - X
Karla Cruz@JobitoPiece MINGW64 /
$ cd "C:\Users\Karla Cruz\AndroidStudioProjects\ProyectosCompartidos"
Karla Cruz@JobitoPiece MINGW64 ~/AndroidStudioProjects/ProyectosCompartidos (main)
$ mkdir Kotlin_Ya
```

Agregamos los proyectos y subimos los cambios:



```
MINGW64:/c/Users/Karla Cruz/AndroidStudioProjects/ProyectosCompartidos... - X
Karla Cruz@JobitoPiece MINGW64 ~/AndroidStudioProjects/ProyectosCompartidos (main)
$ git add .
warning: in the working copy of 'Kotlin_Ya/Proyecto1/.gitignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Kotlin_Ya/Proyecto1/.idea/inspectionProfiles/Project_Default.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Kotlin_Ya/Proyecto1/.idea/vcs.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Kotlin_Ya/Proyecto2/.gitignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Kotlin_Ya/Proyecto2/.idea/inspectionProfiles/Project_Default.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Kotlin_Ya/Proyecto2/.idea/vcs.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Kotlin_Ya/Proyecto5/.gitignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Kotlin_Ya/Proyecto5/.idea/inspectionProfiles/Project_Default.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Kotlin_Ya/Proyecto5/.idea/vcs.xml', LF will be replaced by CRLF the next time Git touches it
Karla Cruz@JobitoPiece MINGW64 ~/AndroidStudioProjects/ProyectosCompartidos (main)
```



```
MINGW64:/c/Users/Karla Cruz/AndroidStudioProjects/ProyectosCompartidos... - X
Karla Cruz@JobitoPiece MINGW64 ~/AndroidStudioProjects/ProyectosCompartidos (main)
$ git commit -m "Añadiendo carpeta Kotlin_Ya con proyectos de Android"
[main d2e81e5] Añadiendo carpeta Kotlin_Ya con proyectos de Android
 33 files changed, 377 insertions(+)
   create mode 100644 Kotlin_Ya/Proyecto1/.gitignore
   create mode 100644 Kotlin_Ya/Proyecto1/.idea/.gitignore
   create mode 100644 Kotlin_Ya/Proyecto1/.idea/codeStyles/Project.xml
   create mode 100644 Kotlin_Ya/Proyecto1/.idea/codeStyles/codeStyleConfig.xml
   create mode 100644 Kotlin_Ya/Proyecto1/.idea/inspectionProfiles/Project_Default.xml
   create mode 100644 Kotlin_Ya/Proyecto1/.idea/kotlinc.xml
   create mode 100644 Kotlin_Ya/Proyecto1/.idea/libraries/KotlinJavaRuntime.xml
   create mode 100644 Kotlin_Ya/Proyecto1/.idea/misc.xml
   create mode 100644 Kotlin_Ya/Proyecto1/.idea/modules.xml
   create mode 100644 Kotlin_Ya/Proyecto1/.idea/vcs.xml
   create mode 100644 Kotlin_Ya/Proyecto1/Proyecto1.iml
   create mode 100644 Kotlin_Ya/Proyecto1/src/Principal.kt
   create mode 100644 Kotlin_Ya/Proyecto2/.gitignore
   create mode 100644 Kotlin_Ya/Proyecto2/.idea/.gitignore
   create mode 100644 Kotlin_Ya/Proyecto2/.idea/inspectionProfiles/Project_Default.xml
   create mode 100644 Kotlin_Ya/Proyecto2/.idea/kotlinc.xml
```

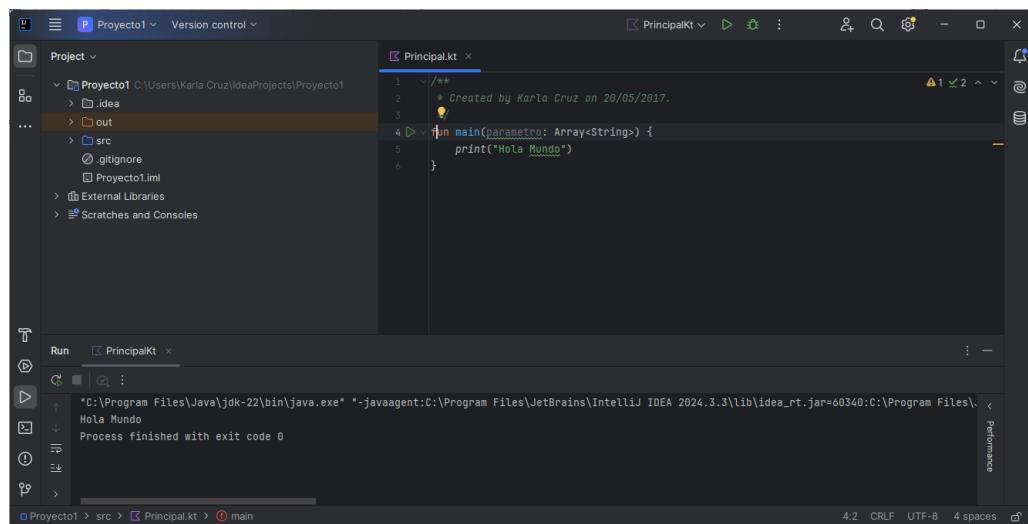
```
MINGW64:/c/Users/Karla Cruz/AndroidStudioProjects/ProyectosCompartidos... - X
Karla Cruz@JobitoPiece MINGW64 ~/AndroidStudioProjects/ProyectosCompartidos (main)
$ git push origin main
Enumerating objects: 33, done.
Counting objects: 100% (33/33), done.
Delta compression using up to 8 threads.
Compressing objects: 100% (28/28), done.
Writing objects: 100% (32/32), 4.78 KiB | 444.00 KiB/s, done.
Total 32 (delta 5), reused 7 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), completed with 1 local object.
To https://github.com/KarlaDenisseCruzSalis/ProyectosCompartidos.git
  38f269f..d2e81e5  main -> main

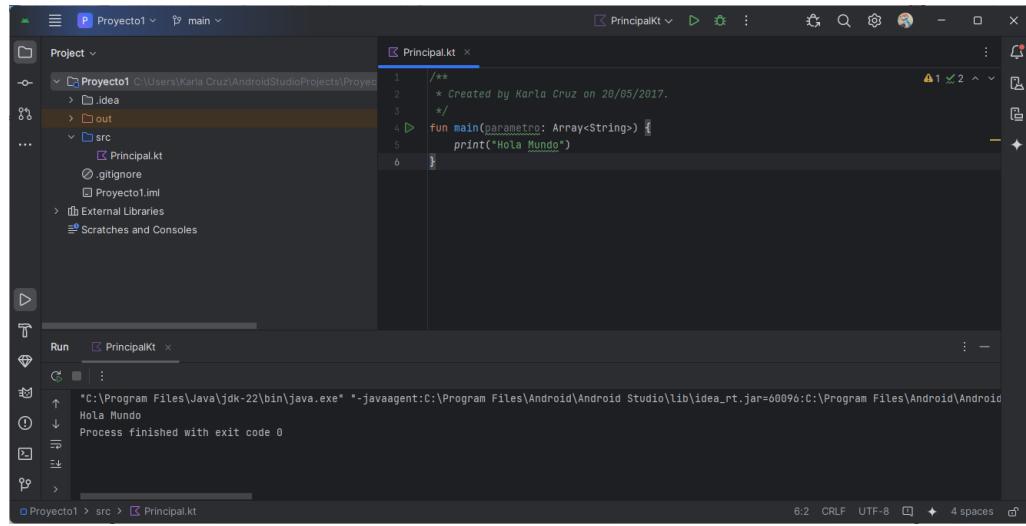
Karla Cruz@JobitoPiece MINGW64 ~/AndroidStudioProjects/ProyectosCompartidos (main)
$ cd "C:\Users\Karla Cruz\AndroidStudioProjects\ProyectosCompartidos\Kotlin_Ya"
Karla Cruz@JobitoPiece MINGW64 ~/AndroidStudioProjects/ProyectosCompartidos/Kotlin_Ya (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.
```

## Capítulo 2

### Problema

Como podemos ver, se desarrollaron las funcionalidades para el primer y segundo proyecto, los cuales se pueden observar a partir de la página 12 del libro:



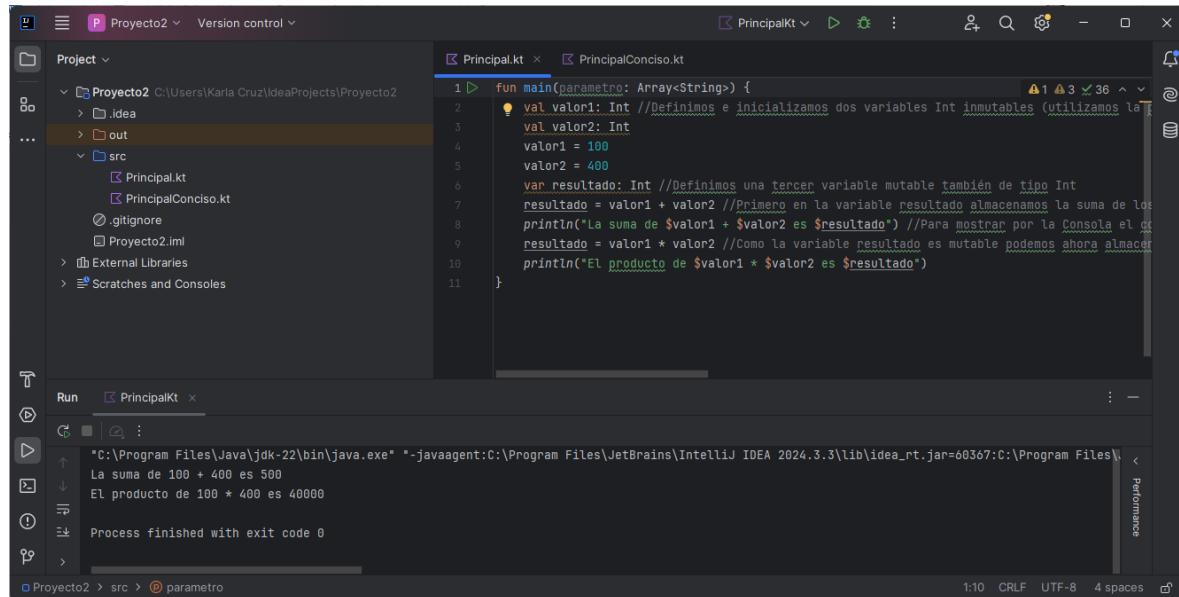


```
/*
 * Created by Karla Cruz on 20/05/2017.
 */
fun main(parametro: Array<String>) {
    println("Hola Mundo")
}
```

Para nuestro primer ejercicio se nos solicitó un Hola Mundo impreso en la consola, mientras que en nuestro segundo ejercicio se nos solicitó realizar operaciones matemáticas básicas (suma y multiplicación) utilizando variables inmutables (val) y mutables (var).

## Capítulo 3

En este segundo proyecto se realizan operaciones matemáticas simples como se mencionaba anteriormente, pero con dos números predefinidos (100 y 400) para luego mostrar los resultados en la consola:



```
fun main(parametro: Array<String>) {
    val valor1: Int = 100
    val valor2: Int = 400
    var resultado: Int = valor1 + valor2
    println("La suma de $valor1 + $valor2 es $resultado")
    resultado *= 2
    println("El producto de $valor1 * $valor2 es $resultado")
}
```

Screenshot of IntelliJ IDEA showing the code in `PrincipalConciso.kt`. The code defines two immutable variables (`val valor1` and `val valor2`) and calculates their sum and product.

```
fun main(parametro: Array<String>) { //Mismo que Principal.kt pero con cambio
    val valor1: Int = 100 // Podemos definir la variable e inmediatamente asignar su valor
    val valor2: Int = 400
    var resultado: Int = valor1 + valor2 //o el contenido de otras variables
    println("La suma de $valor1 + $valor2 es $resultado")
    resultado = valor1 * valor2
    println("El producto de $valor1 * $valor2 es $resultado")
}
```

Screenshot of Android Studio showing the code in `PrincipalKt`. The code defines two immutable variables (`val valor1` and `val valor2`) and calculates their sum and product.

```
fun main(parametro: Array<String>) {
    val valor1: Int //Definimos e inicializamos dos variables Int inmutables (utilizamos la
    val valor2: Int
    valor1 = 100
    valor2 = 400
    var resultado: Int //Definimos una tercera variable mutable también de tipo Int
    resultado = valor1 + valor2 //Primero en la variable resultado almacenamos la suma de los
    resultado = valor1 * valor2 //Como la variable resultado es mutable podemos ahora almacenar
    println("La suma de $valor1 + $valor2 es $resultado") //Para mostrar por la Consola el resultado
    println("El producto de $valor1 * $valor2 es $resultado")
}
```

```
fun main(parametro: Array<String>) { //Mismo que Principal.kt pero con cambio val valor1: Int = 100 // Podemos definir la variable e inmediatamente asignar su valor val valor2: Int = 400 var resultado: Int = valor1 + valor2 //o el contenido de otras variables println("La suma de $valor1 + $valor2 es $resultado") resultado = valor1 * valor2 println("El producto de $valor1 * $valor2 es $resultado") }
```

## Capítulo 4

### Problema 1

En cuanto al proyecto 3 y 4, estos son tomados como propuestas a desarrollar, por lo que el proyecto 5 empieza en el cuarto apartado de datos por teclado en la consola, donde se solicitan dos números al usuario, se realiza una suma y una multiplicación con ellos, y muestra los resultados en consola:

```
fun main(argumento: Array<String>) { // Solicita al usuario que ingrese el primer valor print("Ingrese primer valor:") val valor1 = readLine()!!.toInt() // Lee el valor ingresado por teclado y lo convierte a entero // Solicita al usuario que ingrese el segundo valor print("Ingrese segundo valor:") val valor2 = readLine()!!.toInt() // Lee el valor ingresado por teclado y lo convierte a entero // Calcula la suma de los dos valores val suma = valor1 + valor2 // Imprime el resultado de la suma println("La suma de $valor1 y $valor2 es $suma") }
```

```
fun main(argumento: Array<String>) {  
    // Calcula la suma de los dos valores  
    val suma = valor1 + valor2  
    // Imprime el resultado de la suma  
    println("La suma de $valor1 y $valor2 es $suma")  
  
    // Calcula el producto de los dos valores  
    val producto = valor1 * valor2  
    // Imprime el resultado del producto  
    println("El producto de $valor1 y $valor2 es $producto")  
}  
  
@InlineOnly  
public inline fun println(  
    message: Any?  
) : Unit  
  
Prints the given message and the line  
separator to the standard output stream.  
  
kotlin.io  
ConsoleKt.class
```

The code in Principal.kt performs the following tasks:

- It defines a main function that takes an array of strings as an argument.
- Inside the main function, it declares variables `valor1` and `valor2` and initializes them with the values passed as arguments.
- It calculates the sum of `valor1` and `valor2` and prints the result.
- It calculates the product of `valor1` and `valor2` and prints the result.

The run console shows the output:

```
*C:\Program Files\Java\jdk-22\bin\java.exe* "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\lib\idea_rt.jar=60534:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\bin" -Dfile.encoding=UTF-8  
Ingrese primer valor:24  
Ingrese segundo valor:2  
La suma de 24 y 2 es 26  
El producto de 24 y 2 es 48  
Process finished with exit code 0
```

## Problema 2

Por otra parte, para el proyecto 6 se le pide al usuario que ingrese la medida del lado de un cuadrado, calcula su perímetro y muestra el resultado en consola:

```
fun main(parametro: Array<String>) { // Corrección con tipo String  
    print("Ingrese la medida del lado del cuadrado:")  
    val lado = readLine()!!.toInt()  
    val perimetro = lado * 4  
    println("El perímetro del cuadrado es $perimetro")  
}  
  
// de forma extensa: val lado:Int, lado= readLine()!!.toInt()
```

The code in Principal.kt performs the following tasks:

- It defines a main function that takes an array of strings as an argument.
- Inside the main function, it prompts the user to enter the side length of a square.
- It reads the user input and converts it to an integer.
- It calculates the perimeter of the square (side length multiplied by 4) and prints the result.

The run console shows the output:

```
*C:\Program Files\Java\jdk-22\bin\java.exe* "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\lib\idea_rt.jar=60534:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\bin" -Dfile.encoding=UTF-8  
Ingrese la medida del lado del cuadrado:7  
El perímetro del cuadrado es 28  
Process finished with exit code 0
```

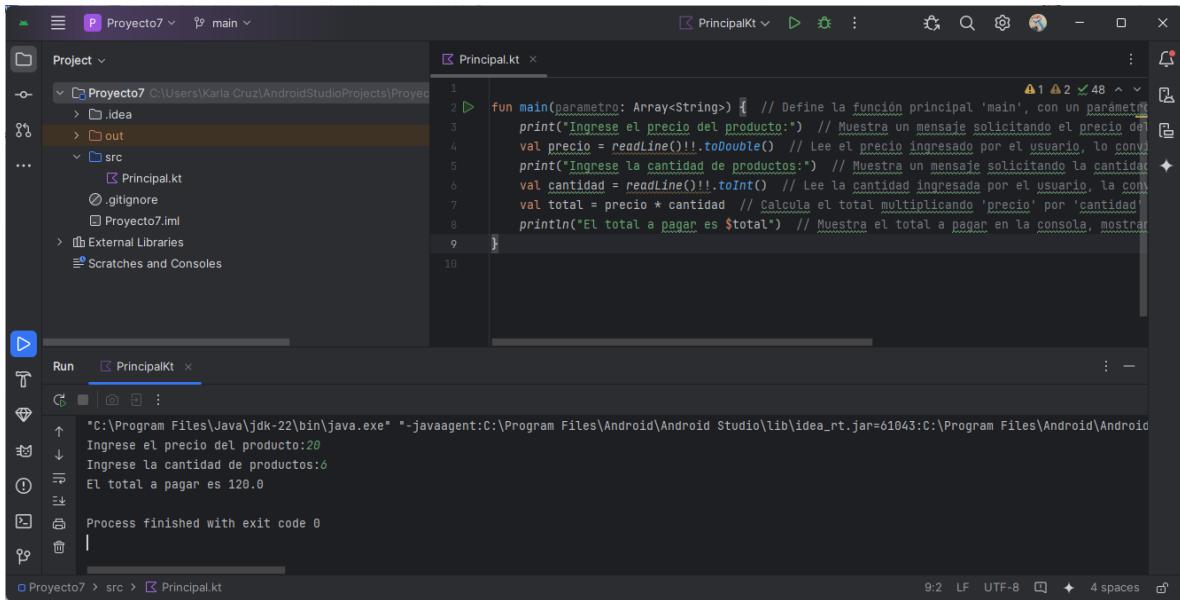
```
fun main(parametros: Array<String>) { // Corrección con tipo String
    print("Ingrese la medida del lado del cuadrado:")
    val lado = readLine()!!.toInt()
    val perimetro = lado * 4
    println("El perimetro del cuadrado es $perimetro")
}

//de forma extensa: val lado:Int, lado= readLine()!!.toInt()
```

### Problema 3

En el proyecto 7 el programa calcula el total a pagar por un producto en función del precio y la cantidad ingresada por el usuario:

```
fun main(parametros: Array<String>) { // Define la función principal 'main', con un parámetro
    print("Ingrese el precio del producto:") // Muestra un mensaje solicitando el precio del producto
    val precio = readLine()!!.toDouble() // Lee el precio ingresado por el usuario, lo convierte a doble y lo guarda en 'precio'
    print("Ingrese la cantidad de productos:") // Muestra un mensaje solicitando la cantidad de productos
    val cantidad = readLine()!!.toInt() // Lee la cantidad ingresada por el usuario, la convierte a entero y lo guarda en 'cantidad'
    val total = precio * cantidad // Calcula el total multiplicando 'precio' por 'cantidad'
    println("El total a pagar es $total") // Muestra el total a pagar en la consola, mostrando el resultado
}
```

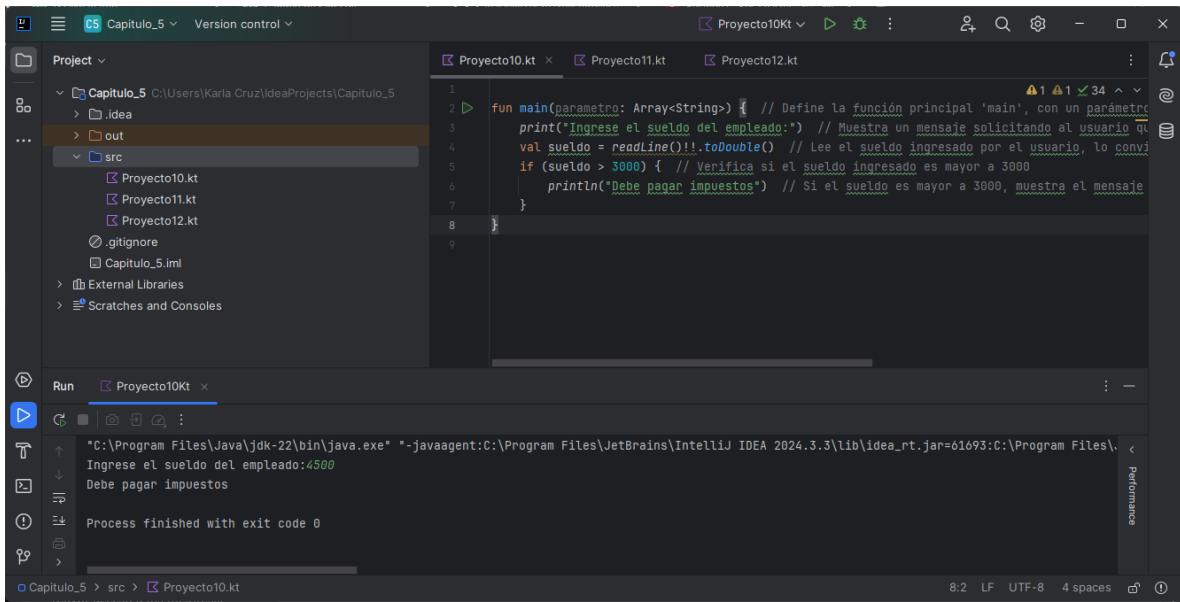


```
fun main(parametro: Array<String>) { // Define la función principal 'main', con un parámetro
    print("Ingrese el precio del producto:") // Muestra un mensaje solicitando el precio del producto
    val precio = readLine()!!.toDouble() // Lee el precio ingresado por el usuario, lo convierte a double y lo guarda en 'precio'
    print("Ingrese la cantidad de productos:") // Muestra un mensaje solicitando la cantidad de productos
    val cantidad = readLine()!!.toInt() // Lee la cantidad ingresada por el usuario, la convierte a int y lo guarda en 'cantidad'
    val total = precio * cantidad // Calcula el total multiplicando 'precio' por 'cantidad'
    println("El total a pagar es $total") // Muestra el total a pagar en la consola, mostrando el resultado
}
```

## Capítulo 5

### Problema 1

Como los proyectos 8 y 9 son propuestas pasamos al proyecto 10. En este proyecto se le solicita al usuario que ingrese el sueldo de un empleado y, si el sueldo es mayor a 3000, muestra el mensaje "Debe pagar impuestos":



```
fun main(parametro: Array<String>) { // Define la función principal 'main', con un parámetro
    print("Ingrese el sueldo del empleado:") // Muestra un mensaje solicitando al usuario que ingrese el sueldo
    val sueldo = readLine()!!.toDouble() // Lee el sueldo ingresado por el usuario, lo convierte a double y lo guarda en 'sueldo'
    if (sueldo > 3000) { // Verifica si el sueldo ingresado es mayor a 3000
        println("Debe pagar impuestos") // Si el sueldo es mayor a 3000, muestra el mensaje
    }
}
```

The screenshot shows the Android Studio interface with the project 'Capítulo\_5' selected. In the center, the code editor displays 'Proyecto10.kt' with the following content:

```
1 fun main(parametro: Array<String>) { // Define la función principal 'main', con un parámetro
2     print("Ingrese el sueldo del empleado:") // Muestra un mensaje solicitando al usuario que ingrese el sueldo
3     val sueldo = readLine()!!.toDouble() // Lee el sueldo ingresado por el usuario, lo convierte a un número decimal
4     if (sueldo > 3000) { // Verifica si el sueldo ingresado es mayor a 3000
5         println("Debe pagar impuestos") // Si el sueldo es mayor a 3000, muestra el mensaje
6     }
7 }
```

Below the code editor, the run tab shows the output of running 'Proyecto10Kt':

```
*C:\Program Files\Java\jdk-22\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62013:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8
Ingrese el sueldo del empleado:2500
Process finished with exit code 0
```

## Problema 2

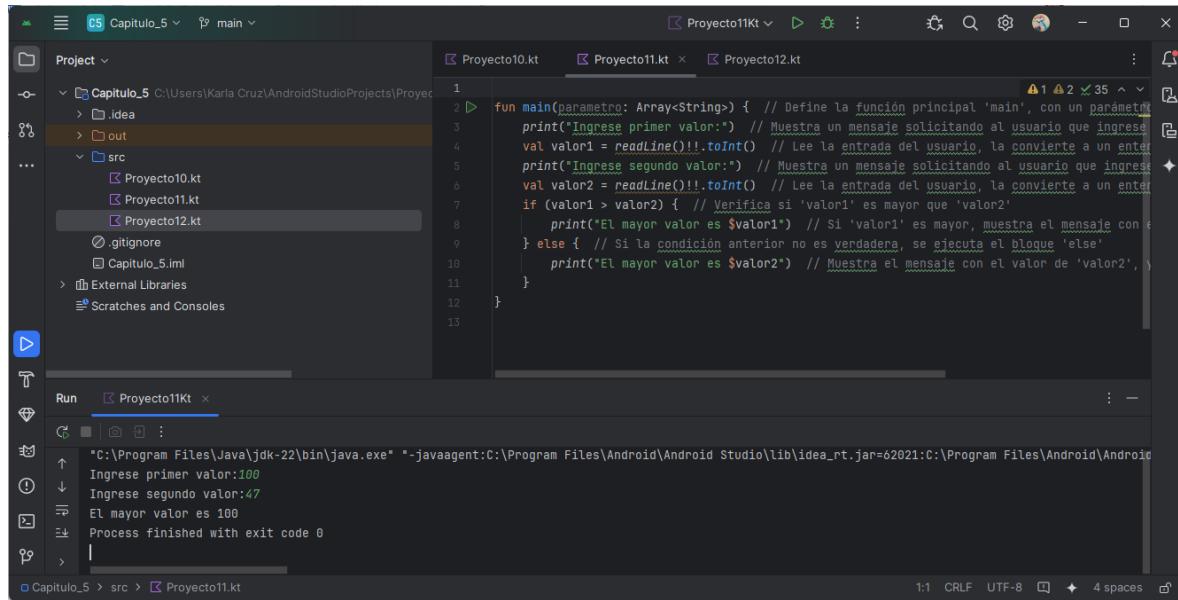
Por otra parte, en el proyecto 11 se comparan dos valores ingresados por el usuario y se muestra cuál de los dos es el mayor:

The screenshot shows the IntelliJ IDEA interface with the project 'Capítulo\_5' selected. In the center, the code editor displays 'Proyecto11.kt' with the following content:

```
1 fun main(parametro: Array<String>) { // Define la función principal 'main', con un parámetro
2     print("Ingrese primer valor:") // Muestra un mensaje solicitando al usuario que ingrese el primer valor
3     val valor1 = readLine()!!.toInt() // Lee la entrada del usuario, la convierte a un entero
4     print("Ingrese segundo valor:") // Muestra un mensaje solicitando al usuario que ingrese el segundo valor
5     val valor2 = readLine()!!.toInt() // Lee la entrada del usuario, la convierte a un entero
6     if (valor1 > valor2) { // Verifica si 'valor1' es mayor que 'valor2'
7         print("El mayor valor es $valor1") // Si 'valor1' es mayor, muestra el mensaje con el valor de 'valor1'
8     } else { // Si la condición anterior no es verdadera, se ejecuta el bloque 'else'
9         print("El mayor valor es $valor2") // Muestra el mensaje con el valor de 'valor2'
10    }
11 }
```

Below the code editor, the run tab shows the output of running 'Proyecto11Kt':

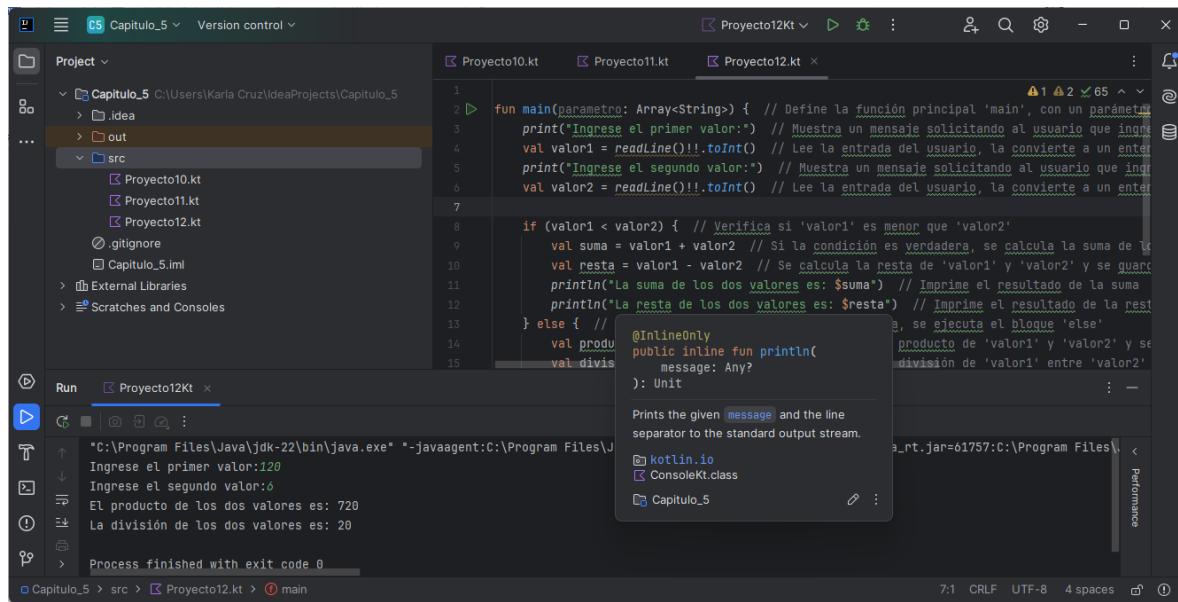
```
*C:\Program Files\Java\jdk-22\bin\java.exe* "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\lib\idea_rt.jar=61749:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\lib\proguard.jar" -Dfile.encoding=UTF-8
Ingrese primer valor:50
Ingrese segundo valor:5
El mayor valor es 50
Process finished with exit code 0
```



```
fun main(parametro: Array<String>) { // Define la función principal 'main', con un parámetro
    print("Ingrese primer valor:") // Muestra un mensaje solicitando al usuario que ingrese un valor
    val valor1 = readLine()!!.toInt() // Lee la entrada del usuario, la convierte a un entero
    print("Ingrese segundo valor:") // Muestra un mensaje solicitando al usuario que ingrese otro valor
    val valor2 = readLine()!!.toInt() // Lee la entrada del usuario, la convierte a un entero
    if (valor1 > valor2) { // Verifica si 'valor1' es mayor que 'valor2'
        print("El mayor valor es $valor1") // Si 'valor1' es mayor, muestra el mensaje con el valor de 'valor1'
    } else { // Si la condición anterior no es verdadera, se ejecuta el bloque 'else'
        print("El mayor valor es $valor2") // Muestra el mensaje con el valor de 'valor2', ya que 'valor1' no es mayor
    }
}
```

### Problema 3

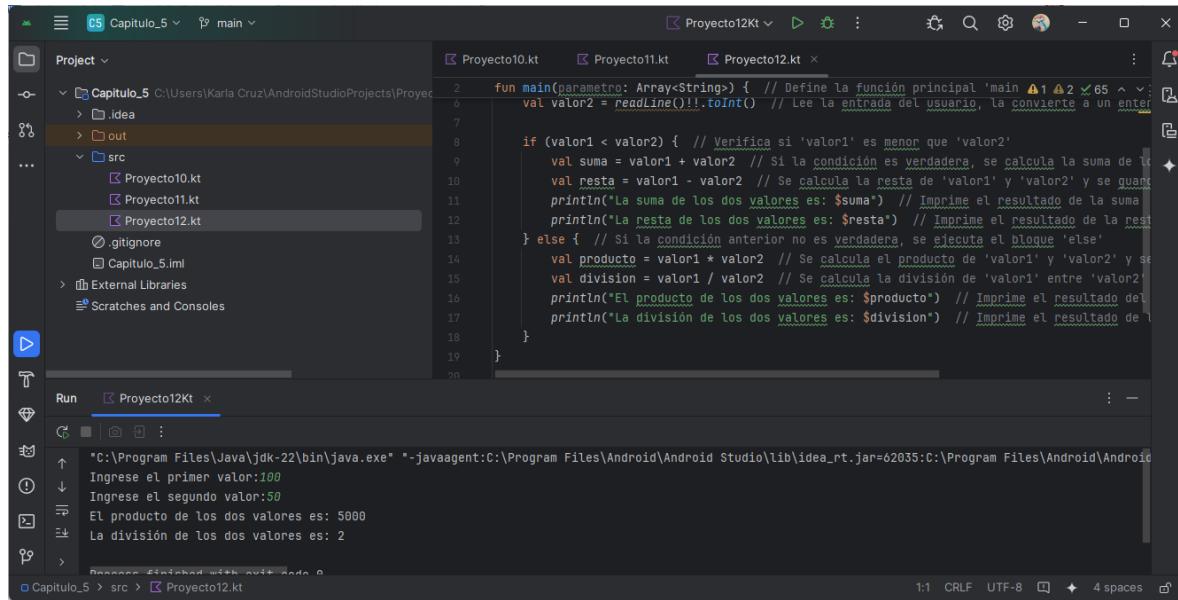
En el proyecto 12 se realizan operaciones matemáticas entre dos valores ingresados por el usuario y las muestra en función de la relación entre esos dos valores:



```
fun main(parametro: Array<String>) { // Define la función principal 'main', con un parámetro
    print("Ingrese el primer valor:") // Muestra un mensaje solicitando al usuario que ingrese un valor
    val valor1 = readLine()!!.toInt() // Lee la entrada del usuario, la convierte a un entero
    print("Ingrese el segundo valor:") // Muestra un mensaje solicitando al usuario que ingrese otro valor
    val valor2 = readLine()!!.toInt() // Lee la entrada del usuario, la convierte a un entero
    if (valor1 < valor2) { // Verifica si 'valor1' es menor que 'valor2'
        val suma = valor1 + valor2 // Si la condición es verdadera, se calcula la suma de los dos valores
        val resta = valor1 - valor2 // Se calcula la resta de 'valor1' y 'valor2' y se guarda en la variable 'resta'
        println("La suma de los dos valores es: $suma") // Imprime el resultado de la suma
        println("La resta de los dos valores es: $resta") // Imprime el resultado de la resta
    } else { // Si la condición es falsa (valor1 ≥ valor2)
        val producto = valor1 * valor2 // Se calcula el producto de 'valor1' y 'valor2' y se guarda en la variable 'producto'
        val divis = valor1 / valor2 // Se calcula la división de 'valor1' entre 'valor2' y se guarda en la variable 'divis'
        println("El producto de los dos valores es: $producto") // Imprime el resultado del producto
        println("La división de los dos valores es: $divis") // Imprime el resultado de la división
    }
}

@InlineOnly
public inline fun println(
    message: Any?
): Unit {
    Prints the given message and the line separator to the standard output stream.
}

```



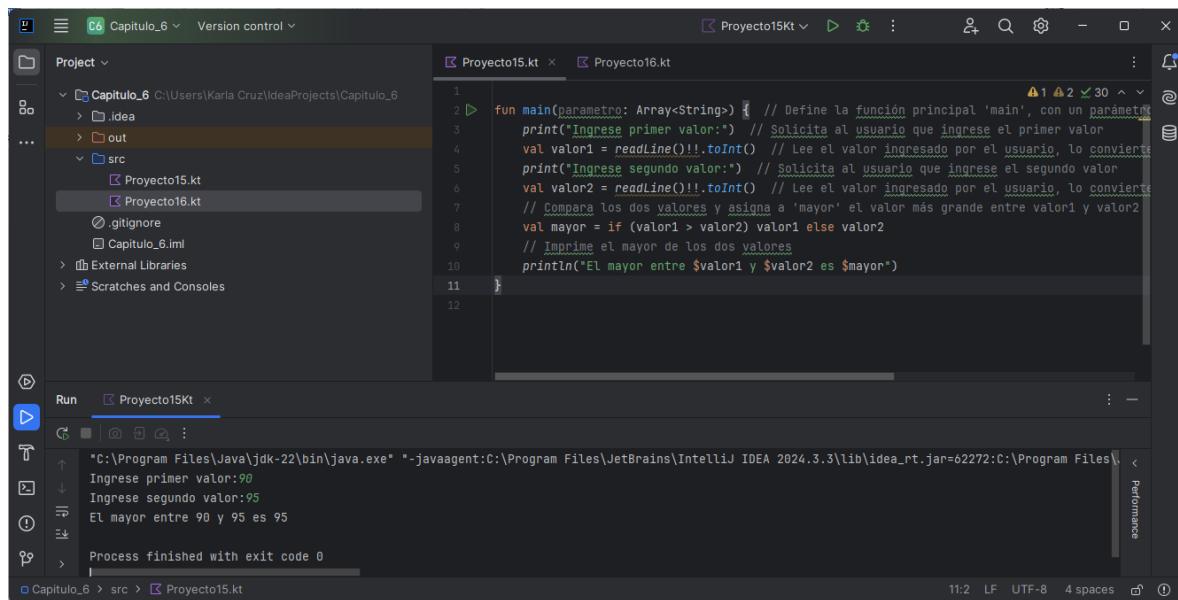
```
fun main(parametro: Array<String>) { // Define la función principal 'main', con un parámetro
    val valor1 = readLine()!!.toInt() // Lee la entrada del usuario, la convierte a un entero
    val valor2 = readLine()!!.toInt() // Lee la entrada del usuario, la convierte a un entero

    if (valor1 < valor2) { // Verifica si 'valor1' es menor que 'valor2'
        val suma = valor1 + valor2 // Si la condición es verdadera, se calcula la suma de los dos valores
        val resta = valor1 - valor2 // Se calcula la resta de 'valor1' y 'valor2' y se guarda en la variable 'resta'
        println("La suma de los dos valores es: $suma") // Imprime el resultado de la suma
        println("La resta de los dos valores es: $resta") // Imprime el resultado de la resta
    } else { // Si la condición anterior no es verdadera, se ejecuta el bloque 'else'
        val producto = valor1 * valor2 // Se calcula el producto de 'valor1' y 'valor2' y se guarda en la variable 'producto'
        val division = valor1 / valor2 // Se calcula la división de 'valor1' entre 'valor2' y se guarda en la variable 'division'
        println("El producto de los dos valores es: $producto") // Imprime el resultado del producto
        println("La división de los dos valores es: $division") // Imprime el resultado de la división
    }
}
```

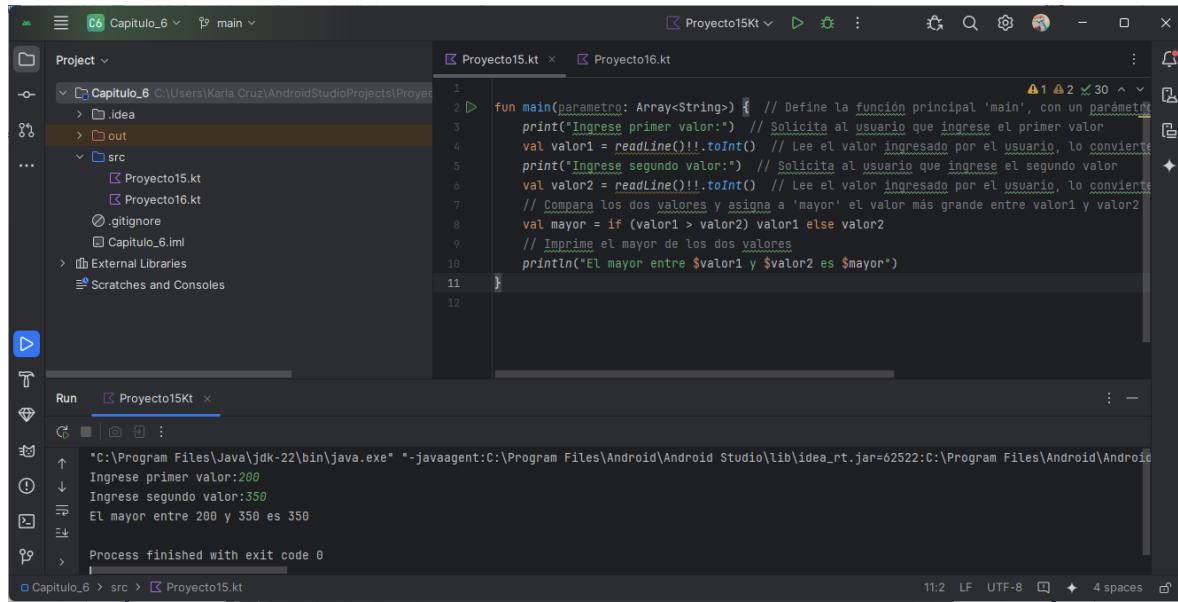
## Capítulo 6

### Problema 1

Los proyectos 13 y 14 son propuestas, pero en cuanto al proyecto 15 se nos solicita pedirle al usuario dos valores, para luego compararlos y determinar cuál es el mayor de los dos. Luego, se imprime el valor mayor.



```
fun main(parametro: Array<String>) { // Define la función principal 'main', con un parámetro
    print("Ingrese primer valor:") // Sigue al usuario que ingrese el primer valor
    val valor1 = readLine()!!.toInt() // Lee el valor ingresado por el usuario, lo convierte a un entero
    print("Ingrese segundo valor:") // Sigue al usuario que ingrese el segundo valor
    val valor2 = readLine()!!.toInt() // Lee el valor ingresado por el usuario, lo convierte a un entero
    // Compara los dos valores y asigna a 'mayor' el valor más grande entre 'valor1' y 'valor2'
    val mayor = if (valor1 > valor2) valor1 else valor2
    // Imprime el mayor de los dos valores
    println("El mayor entre $valor1 y $valor2 es $mayor")
}
```



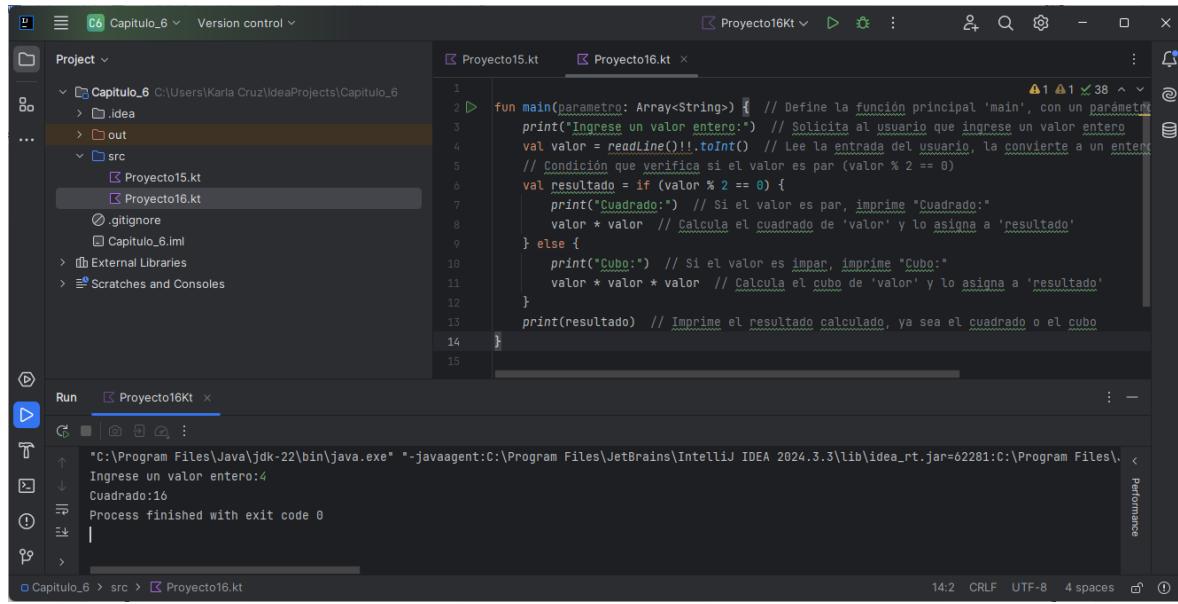
```
fun main(parametro: Array<String>) { // Define la función principal 'main', con un parámetro
    print("Ingrese primer valor:") // Sigue al usuario que ingrese el primer valor
    val valor1 = readLine()!!.toInt() // Lee el valor ingresado por el usuario, lo convierte a un entero
    print("Ingrese segundo valor:") // Sigue al usuario que ingrese el segundo valor
    val valor2 = readLine()!!.toInt() // Lee el valor ingresado por el usuario, lo convierte a un entero
    // Compara los dos valores y asigna a 'mayor' el valor más grande entre valor1 y valor2
    val mayor = if (valor1 > valor2) valor1 else valor2
    // Imprime el mayor de los dos valores
    println("El mayor entre $valor1 y $valor2 es $mayor")
}
```

The Run tab shows the output of the program:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62522:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8
Ingrese primer valor:200
Ingrese segundo valor:350
El mayor entre 200 y 350 es 350
Process finished with exit code 0
```

## Problema 2

En cuanto al proyecto 16 el programa solicita al usuario que ingrese un valor entero, luego verifica si el valor es par o impar. Si el valor es par, calcula su cuadrado; si es impar, calcula su cubo. Finalmente, imprime el resultado:



```
fun main(parametro: Array<String>) { // Define la función principal 'main', con un parámetro
    print("Ingrese un valor entero:") // Sigue al usuario que ingrese un valor entero
    val valor = readLine()!!.toInt() // Lee la entrada del usuario, la convierte a un entero
    // Condición que verifica si el valor es par (valor % 2 == 0)
    val resultado = if (valor % 2 == 0) {
        print("Cuadrado:") // Si el valor es par, imprime "Cuadrado"
        valor * valor // Calcula el cuadrado de 'valor' y lo asigna a 'resultado'
    } else {
        print("Cubo:") // Si el valor es impar, imprime "Cubo"
        valor * valor * valor // Calcula el cubo de 'valor' y lo asigna a 'resultado'
    }
    print(resultado) // Imprime el resultado calculado, ya sea el cuadrado o el cubo
}
```

The Run tab shows the output of the program:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\lib\idea_rt.jar=62281:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\lib\proguard.jar" -Dfile.encoding=UTF-8
Ingrese un valor entero:4
Cuadrado:16
Process finished with exit code 0
```

```
1 fun main(parametro: Array<String>) { // Define la función principal 'main', con un parámetro
2     println("Ingrese un valor entero:") // Solicita al usuario que ingrese un valor entero
3     val valor = readLine()!!.toInt() // Lee la entrada del usuario, la convierte a un entero
4     // Condición que verifica si el valor es par (valor % 2 == 0)
5     val resultado = if (valor % 2 == 0) {
6         println("Cuadrado:") // Si el valor es par, imprime "Cuadrado"
7         valor * valor // Calcula el cuadrado de 'valor' y lo asigna a 'resultado'
8     } else {
9         println("Cubo:") // Si el valor es impar, imprime "Cubo"
10        valor * valor * valor // Calcula el cubo de 'valor' y lo asigna a 'resultado'
11    }
12    print(resultado) // Imprime el resultado calculado, ya sea el cuadrado o el cubo
13 }
14 }
```

## Capítulo 7

En este capítulo aprendimos a utilizar la estructura condicional `if`, que permite ejecutar diferentes bloques de código según si se cumple o no una condición lógica. Esta es una de las herramientas más básicas y poderosas en programación, esencial para la toma de decisiones.

A través de un ejercicio práctico, aplicamos el uso de `if` como expresión, permitiendo asignar el resultado directamente a una variable en función de ciertas condiciones.

### Problema 1

```
1 fun Proyecto18() {
2     println("Capítulo 18: Promedio sentencia")
3     println("Ingrese primer nota:")
4     val nota1 = readLine()!!.toInt()
5     println("Ingrese segunda nota:")
6     val nota2 = readLine()!!.toInt()
7     println("Ingrese tercera nota:")
8     val nota3 = readLine()!!.toInt()
9     val promedio = (nota1 + nota2 + nota3) / 3
10    val estado = if (promedio >= 7) "Promocionado"
11    else if (promedio >= 4) "Regular"
12    else "Libre"
13    println("Estado del alumno $estado")
14 }
```

El programa solicita al usuario tres notas, calcula el promedio y determina el estado académico del alumno según el resultado:

- Promocionado si el promedio es mayor o igual a 7
- Regular si está entre 4 y 6
- Libre si es menor a 4

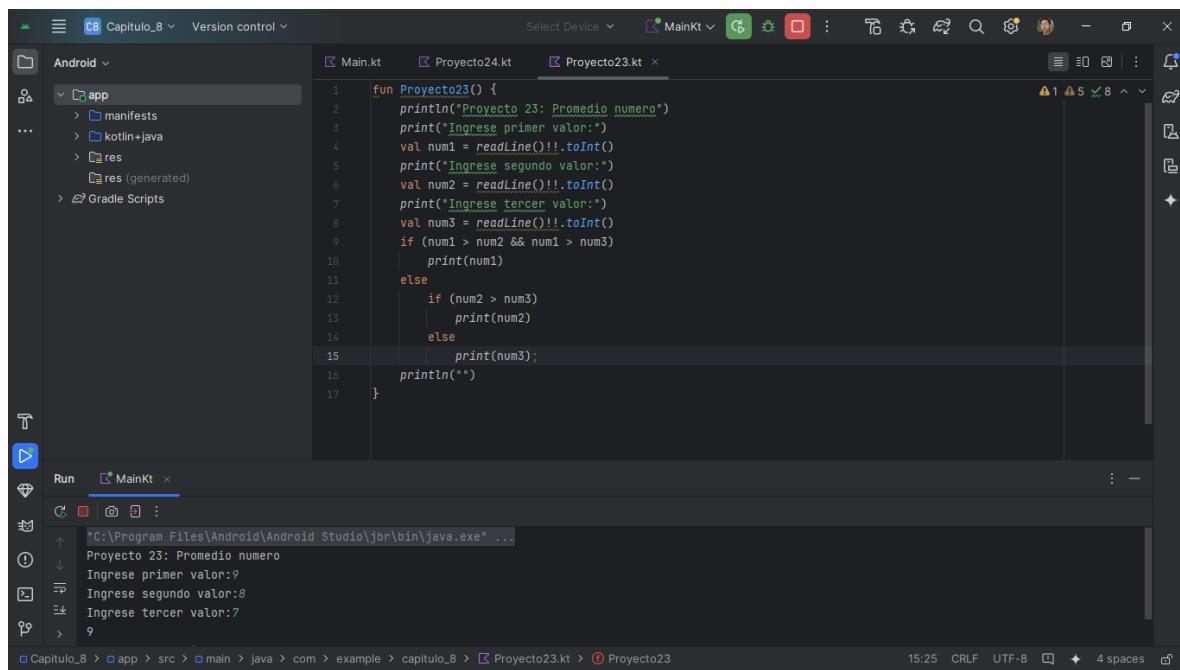
Se utiliza la sentencia if como expresión para asignar directamente el resultado (el estado) a una variable, lo cual hace el código más limpio y conciso.

## Capítulo 8

En este capítulo se abordó el uso de estructuras condicionales anidadas y compuestas. A diferencia del if simple, estas estructuras permiten evaluar múltiples condiciones encadenadas o en combinación lógica, haciendo posible una toma de decisiones más compleja y precisa.

A través de dos ejercicios prácticos, exploramos cómo identificar el mayor de tres números y cómo determinar el trimestre correspondiente a una fecha ingresada.

### Problema 1



The screenshot shows the Android Studio interface with the Main.kt file open in the editor. The code defines a function `Proyecto23()` that prints "Proyecto 23: Promedio numero", reads three integers from the user, and then uses nested if statements to determine and print the largest number. The run tab shows the application output: "Proyecto 23: Promedio numero", followed by three lines where the user inputs "Ingrese primer valor:9", "Ingrese segundo valor:8", and "Ingrese tercer valor:7". The output then shows the largest value entered, which is 9.

```
fun Proyecto23() {
    println("Proyecto 23: Promedio numero")
    print("Ingrese primer valor:")
    val num1 = readLine()!!.toInt()
    print("Ingrese segundo valor:")
    val num2 = readLine()!!.toInt()
    print("Ingrese tercer valor:")
    val num3 = readLine()!!.toInt()
    if (num1 > num2 && num1 > num3)
        println(num1)
    else
        if (num2 > num3)
            print(num2)
        else
            print(num3);
    println("*)"
}
```

Este programa solicita tres números enteros y determina cuál es el mayor entre ellos. Utiliza una serie de condicionales anidados para comparar los valores entre sí:

- Si num1 es mayor que num2 y num3, se imprime num1.

- En caso contrario, se compara num2 con num3, y se imprime el mayor de los dos.

## Problema 2

The screenshot shows the Android Studio interface. On the left, the project structure for 'Capítulo\_8' is visible, with 'app' selected. In the center, the code editor displays 'Main.kt' with the following content:

```

1 fun Proyecto24() {
2     println("Proyecto 24: trimestre")
3     print("Ingrese dia:")
4     val dia = readLine()!!.toInt()
5     print("Ingrese mes:")
6     val mes = readLine()!!.toInt()
7     print("Ingrese Año:")
8     val año = readLine()!!.toInt()
9     if (mes == 1 || mes == 2 || mes == 3)
10         println("Corresponde al primer trimestre");
11 }

```

Below the code editor, the 'Run' tab is selected, showing the run history:

- Proyecto 24: trimestre
- Ingrese dia:26
- Ingrese mes:05
- Ingrese Año:2025
- Process finished with exit code 0

The bottom status bar indicates the file path 'Capítulo\_8 > app > src > main > java > com > example > capítulo\_8 > Main.kt' and encoding 'UTF-8'.

Este ejercicio solicita una **fecha completa** (día, mes y año) y verifica si el mes corresponde al **primer trimestre del año**:

- Si el mes es enero, febrero o marzo (mes == 1 || mes == 2 || mes == 3), se muestra un mensaje indicando que la fecha pertenece al primer trimestre.

No realiza ninguna otra validación (como comprobar si la fecha es válida), ya que el objetivo es practicar condiciones compuestas.

## Capítulo 9

En este capítulo se introdujo la estructura de repetición while, una herramienta fundamental para ejecutar un bloque de código mientras se cumpla una condición lógica. A diferencia de otras estructuras de repetición como for, while es útil cuando no se conoce de antemano cuántas veces se repetirá el ciclo.

A través de los ejercicios prácticos de este capítulo, aprendimos a usar while para realizar conteos, acumular valores, calcular promedios y aplicar condiciones dentro de repeticiones.

## Problema 1

The screenshot shows the Android Studio interface with the code editor open. The project structure on the left shows a file named 'Proyecto31.kt' selected. The code in the editor is:

```
1 fun Proyecto31() {
2     println("Capítulo 31: Del 1 al 100")
3     var x = 1
4     while (x <= 100) {
5         println(x)
6         x = x + 1
7     }
8 }
```

The run tab is selected, showing the output: "Process finished with exit code 0".

Este programa imprime los números del 1 al 100, uno por línea, utilizando un ciclo while. El contador x comienza en 1 y se incrementa en cada iteración hasta llegar a 100.

## Problema 2

The screenshot shows the Android Studio interface with the code editor open. The project structure on the left shows a file named 'Proyecto32.kt' selected. The code in the editor is:

```
1 fun Proyecto32() {
2     println("Capítulo 32: Del 1 hasta donde tu quieras")
3     print("Ingrese un valor:")
4     val valor = readLine()!!.toInt()
5     var x = 1
6     while (x <= valor) {
7         println(x)
8         x = x + 1
9     }
10 }
```

The run tab is selected, showing the output: "C:\Program Files\Android\Android Studio\jbr\bin\java.exe" ... Capítulo 32: Del 1 hasta donde tu quieras Ingrese un valor:10 1 2 3 4 5 6 7 8 9 10 Process finished with exit code 0".

Este ejercicio amplía el anterior permitiendo que el usuario decida hasta qué número contar. Se lee un valor desde consola y se imprime del 1 hasta ese número utilizando while.

### Problema 3

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays a Kotlin script named 'Proyecto33.kt' which contains a function 'Proyecto33()' that reads 10 integers from the user, calculates their sum, and then prints the average.

```
fun Proyecto33() {
    println("Capítulo 33: Promedio")
    var x = 1
    var suma = 0
    while (x <= 10) {
        print("Ingrese un valor:")
        val valor=readLine()!!.toInt()
        suma = suma + valor
        x = x + 1
    }
    println("La suma de los 10 valores ingresados es $suma")
    val promedio = suma / 10
    println("El promedio es $promedio")
}
```

The run log at the bottom shows the user inputting 10 values and the final output:

```
Ingrese un valor:9
Ingrese un valor:8
Ingrese un valor:7
Ingrese un valor:6
Ingrese un valor:5
Ingrese un valor:7
Ingrese un valor:8
Ingrese un valor:9
Ingrese un valor:8
Ingrese un valor:7
La suma de los 10 valores ingresados es 74
El promedio es 7.4
```

Este código solicita al usuario **diez números**, uno por uno, y va acumulando la suma. Luego calcula el **promedio** dividiendo la suma total entre 10. Es una aplicación clara del while para iteraciones controladas por contador.

### Problema 4

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays a Kotlin script named 'Proyecto34.kt' which contains a function 'Proyecto34()' that reads the number of pieces to process and then loops through them, checking if each piece's length is between 1.20 and 1.30 meters.

```
fun Proyecto34() {
    println("Capítulo 34: Piezas aptas")
    print("Cuantas piezas procesara:")
    val n = readLine()!!.toInt()
    var x = 1
    var cantidad = 0
    while (x <= n) {
        print("Ingrese la medida de la pieza:")
        val largo = readLine()!!.toDouble()
        if (largo >= 1.20 && largo <= 1.30)
            cantidad = cantidad + 1
        x = x + 1
    }
    println("La cantidad de piezas aptas son: $cantidad")
}
```

The run log at the bottom shows the user inputting the number of pieces and then individual lengths, followed by the final output:

```
C:\Program Files\Android\Android Studio\jbr\bin\java.exe" ...
Capítulo 34: Piezas aptas
Cuantas piezas procesara:3
Ingrese la medida de la pieza:1.10
Ingrese la medida de la pieza:1.40
Ingrese la medida de la pieza:1.25
La cantidad de piezas aptas son: 1
Process finished with exit code 0
```

Este ejercicio permite al usuario ingresar la **cantidad de piezas que quiere procesar**. Para cada pieza, se pide la medida y se evalúa si está dentro del rango apto (entre 1.20 y 1.30 metros). Al final, muestra cuántas piezas fueron aptas.

## Capítulo 10

En este capítulo conocimos la estructura de control do...while, una variante de los ciclos en la cual el bloque de código se ejecuta al menos una vez, antes de verificar la condición. Esta estructura es útil cuando necesitamos asegurarnos de que una acción se realice al menos una vez, incluso si la condición final ya no se cumple desde el inicio.

Los programas que realizamos nos permitieron aplicar esta lógica a casos prácticos como contar dígitos, calcular promedios, y clasificar piezas según su peso.

### Problema 1

The screenshot shows the Android Studio interface. On the left, the project structure is visible, showing modules like app, manifests, kotlin+java, and com.example.capitulo\_10. The MainKt tab is selected in the top navigation bar. The code editor contains the following Kotlin code:

```
fun Proyecto42() {
    println("Proyecto 42: Cantidad de dígitos")
    do {
        print("Ingrese un valor comprendido entre 0 y 999:")
        val valor = readLine()!!.toInt()
        if (valor < 10)
            println("El valor ingresado tiene un dígito")
        else
            if (valor < 100)
                println("El valor ingresado tiene dos dígitos")
            else
                println("El valor ingresado tiene tres dígitos")
    } while (valor != 0)
}
```

The Run tab is selected, showing the output of the program. The terminal window displays the following text:

```
Proyecto 42: Cantidad de dígitos
Ingrese un valor comprendido entre 0 y 999:246
El valor ingresado tiene tres dígitos
Ingrese un valor comprendido entre 0 y 999:5
El valor ingresado tiene un dígito
Ingrese un valor comprendido entre 0 y 999:24
El valor ingresado tiene dos dígitos
Ingrese un valor comprendido entre 0 y 999:0
El valor ingresado tiene un dígito
Process finished with exit code 0
```

At the bottom, the file path is shown as Capítulo\_10 > app > src > main > java > com > example > capitulo\_10 > Proyecto42.kt, along with other status information like line count (15), character count (15), and encoding (UTF-8).

Este programa permite ingresar varios valores entre 0 y 999 y muestra cuántos dígitos tiene cada número ingresado. El ciclo se repite hasta que se introduce el valor 0, el cual sirve como condición de salida.

## Problema 2

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays a Kotlin script named 'Proyecto43.kt' which calculates the average of input numbers. The run tab shows the output of the program, which asks for values and prints the average.

```
fun Proyecto43() {
    println("Proyecto 43: Promedio")
    var cant = 0
    var suma = 0
    do {
        print("Ingrese un valor (0 para finalizar):")
        val valor = readLine()!!.toInt()
        if (valor != 0) {
            suma += valor
            cant++
        }
    } while (valor != 0)
    if (cant != 0) {
        val promedio = suma / cant
        print("El promedio de los valores ingresados es: $promedio")
    } else
        print("No se ingresaron valores.")
}
```

Run tab output:

```
Proyecto 43: Promedio
Ingrese un valor (0 para finalizar):3
Ingrese un valor (0 para finalizar):2
Ingrese un valor (0 para finalizar):4
Ingrese un valor (0 para finalizar):0
El promedio de los valores ingresados es: 3
Process finished with exit code 0
```

Este ejercicio permite al usuario **ingresar varios números enteros** hasta que escriba 0. Luego **calcula el promedio de todos los números ingresados (excluyendo el cero)**. También maneja el caso en que no se ingresan valores distintos de cero.

## Problema 3

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays a Kotlin script named 'Proyecto44.kt' which counts pieces based on their weight. The run tab shows the output of the program, which asks for piece weights and categorizes them.

```
fun Proyecto44() {
    println("Proyecto 44: Contador de piezas")
    var cant1 = 0
    var cant2 = 0
    var cant3 = 0
    do {
        print("Ingrese el peso de la pieza (0 para finalizar):")
        val peso = readLine()!!.toDouble()
        if (peso > 10.2)
            cant1++
        else
            if (peso > 9.8)
                cant2++
            else
                if (peso > 0)
                    cant3++
    } while(peso != 0.0)
}
```

Run tab output:

```
Proyecto 44: Contador de piezas
Ingrese el peso de la pieza (0 para finalizar):10.3
Ingrese el peso de la pieza (0 para finalizar):9.9
Ingrese el peso de la pieza (0 para finalizar):9.7
Ingrese el peso de la pieza (0 para finalizar):0
Piezas aptas: 1
Piezas con un peso superior a 10.2: 1
Piezas con un peso inferior a 9.8: 1
Cantidad total de piezas procesadas: 3
```

Este programa permite ingresar pesos de piezas industriales y las clasifica en tres grupos:

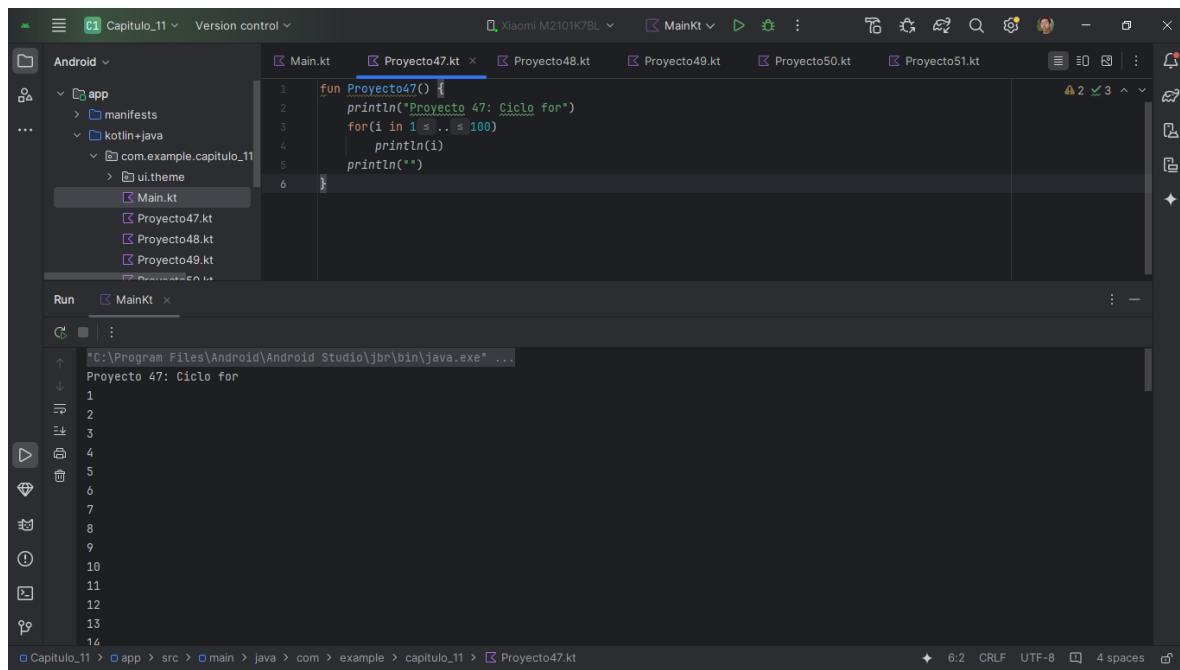
- Aptas: peso entre 9.8 y 10.2 kg.
- ↑ Peso superior a 10.2 kg.
- ↓ Peso inferior a 9.8 kg.

El ingreso de datos continúa hasta que se introduce un peso de **0**, y al final muestra el conteo de cada grupo y la cantidad total de piezas procesadas.

## Capítulo 11

En este capítulo exploramos el uso de estructuras repetitivas utilizando el ciclo for en el lenguaje de programación Kotlin. A través de diversos ejercicios prácticos, aplicamos esta estructura para resolver problemas básicos de iteración, acumulación de valores, conteo condicional y análisis de datos. El objetivo fue familiarizarse con el ciclo for, su sintaxis y su aplicación en problemas reales, permitiendo al estudiante automatizar tareas repetitivas y manipular secuencias numéricas o de entrada de forma eficiente.

### Problema 1



The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "Capítulo\_11". It contains an "app" module with "manifests", "kotlin+java", and "com.example.capítulo\_11" packages. Inside "com.example.capítulo\_11", there is a "ui.theme" folder and several files: Main.kt, Proyecto47.kt, Proyecto48.kt, and Proyecto49.kt.
- Main.kt File Content:**

```
fun Proyecto47() {
    println("Proyecto 47: Ciclo for")
    for(i in 1 .. 100)
        println(i)
    println("")
```
- Run Tab:** The "MainKt" tab is selected. The run output shows the following text:  
"C:\Program Files\Android\Android Studio\jbr\bin\java.exe" ...  
Proyecto 47: Ciclo for  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14
- Status Bar:** Shows file path: Capítulo\_11 > app > src > main > java > com > example > capítulo\_11 > Proyecto47.kt, and other settings: 6:2, CRLF, UTF-8, 4 spaces.

Este programa utiliza un ciclo for para imprimir en pantalla todos los números del 1 al 100, uno por línea. El ciclo se escribe con la sintaxis `for(i in 1..100)`, lo que significa que i tomará los valores del 1 al 100 de forma secuencial.

## Problema 2

The screenshot shows the Android Studio interface with the code editor open. The code in `Projecto48.kt` is as follows:

```
fun Projecto48() {
    println("Proyecto 48: Suma y Promedio con ciclo for")
    var suma = 0
    for(i in 1 .. 10) {
        print("Ingrese un valor:")
        val valor = readLine()!!.toInt()
        suma += valor
    }
    println("La suma de los valores ingresados es $suma")
    val promedio = suma / 10
    println("Su promedio es $promedio")
    println("")
}
```

In the Run tab, the output is:

```
*C:\Program Files\Android\Android Studio\jbr\bin\java.exe* ...
Proyecto 48: Suma y Promedio con ciclo for
Ingrese un valor:3
Ingrese un valor:5
Ingrese un valor:4
Ingrese un valor:9
Ingrese un valor:7
Ingrese un valor:6
Ingrese un valor:5
Ingrese un valor:2
Ingrese un valor:7
La suma de los valores ingresados es 56
Su promedio es 5
```

Este programa pide al usuario que ingrese 10 valores numéricos. Cada uno se suma acumulativamente en la variable suma. Al finalizar, se imprime el total de la suma y el promedio de los valores. El promedio se obtiene dividiendo suma / 10.

## Problema 3

The screenshot shows the Android Studio interface with the code editor open. The code in `Projecto49.kt` is as follows:

```
fun Projecto49() {
    var aprobados = 0
    var reprobados = 0
    for(i in 1 .. 10) {
        print("Ingrese nota:")
        val nota = readLine()!!.toInt()
        if (nota >= 7)
            aprobados++
        else
            reprobados++
    }
    println("Cantidad de alumnos con notas mayores o iguales a 7: $aprobados")
    println("Cantidad de alumnos con notas menores a 7: $reprobados")
}
```

In the Run tab, the output is:

```
Proyecto 49: Aprobados y Reprobados
Ingrese nota:6
Ingrese nota:7
Ingrese nota:8
Ingrese nota:9
Ingrese nota:5
Ingrese nota:9
Ingrese nota:8
Ingrese nota:7
Ingrese nota:5
Cantidad de alumnos con notas mayores o iguales a 7: 6
Cantidad de alumnos con notas menores a 7: 4
```

El programa solicita 10 notas de estudiantes. Si la nota ingresada es mayor o igual a 7, se considera aprobado y se incrementa el contador aprobados. De lo contrario, se incrementa el contador reprobados. Al final, muestra cuántos alumnos aprobaron y cuántos reprobaron.

## Problema 4

The screenshot shows the Android Studio interface with the code for Problem 4 in the main editor. The code is a Kotlin function named `Proyecto50()` that prints "Proyecto 50: Multiples", initializes variables `mult3`, `mult5`, and `mult9` to 0, and then loops through numbers from 1 to 10,000. For each number, it checks if it is divisible by 3, 5, or 8 and increments the respective counter. Finally, it prints the counts for each divisor. The run output shows the results: Cantidad de múltiplos de 3: 3333, Cantidad de múltiplos de 5: 2000, and Cantidad de múltiplos de 9: 1250.

```
fun Proyecto50() {
    println("Proyecto 50: Multiples")
    var mult3 = 0
    var mult5 = 0
    var mult9 = 0
    for(i in 1 .. 10000) {
        if (i % 3 == 0)
            mult3++
        if (i % 5 == 0)
            mult5++
        if (i % 8 == 0)
            mult9++
    }
    println("Cantidad de múltiplos de 3: $mult3")
    println("Cantidad de múltiplos de 5: $mult5")
    println("Cantidad de múltiplos de 9: $mult9")
    println("")
}
```

Cantidad de múltiplos de 3: 3333  
Cantidad de múltiplos de 5: 2000  
Cantidad de múltiplos de 9: 1250

Process finished with exit code 0

Este programa analiza los números del 1 al 10,000 y cuenta cuántos son múltiplos de 3, de 5 y de 8. Utiliza el operador módulo (%) para comprobar si un número es divisible por otro. Al final, muestra la cantidad total de múltiplos encontrados para cada número.

## Problema 5

The screenshot shows the Android Studio interface with the code for Problem 5 in the main editor. The code is a Kotlin function named `Proyecto51()` that prints "Proyecto 51: Contador de pares", asks for the number of values to analyze, reads the values one by one, and checks if each value is even (using `valor % 2 == 0`). If it is, it increments a counter `cant`. Finally, it prints the total count of even numbers. The run output shows the interaction with the user: asking for the number of values, then prompting for each value and checking if it's even, and finally printing the total count.

```
fun Proyecto51() {
    println("Proyecto 51: Contador de pares")
    var cant = 0
    print("Cuantos valores ingresará para analizar:")
    val cantidad = readLine()!!.toInt()
    for(i in 1 .. cantidad) {
        print("Ingrese valor:")
        val valor = readLine()!!.toInt()
        if (valor % 2 == 0)
            cant++
    }
    println("Cantidad de pares: $cant")
    println("")
}
```

Proyecto 51: Contador de pares  
Cuantos valores ingresará para analizar:  
Ingrese valor:8  
Ingrese valor:9  
Ingrese valor:7  
Cantidad de pares: 1

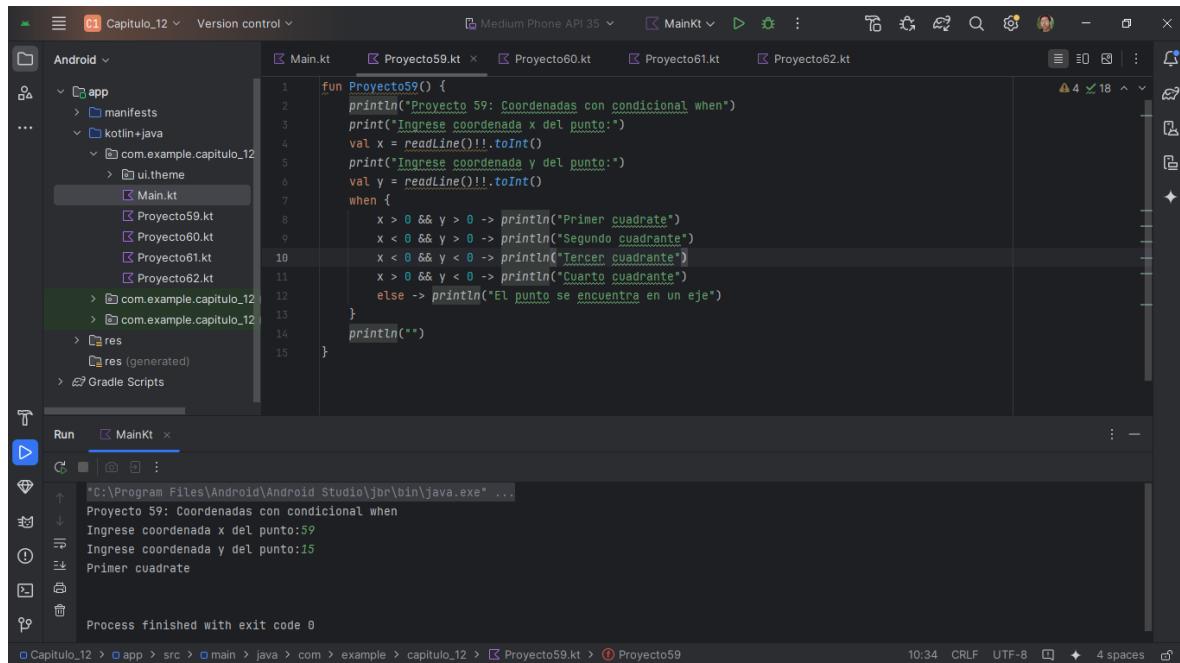
Process finished with exit code 0

Este programa primero pregunta al usuario cuántos números desea ingresar. Luego, en un ciclo, solicita esos valores y verifica si cada uno es par (usando `valor % 2 == 0`). Si lo es, incrementa el contador `cant`. Finalmente, imprime cuántos de los valores ingresados fueron pares.

## Capítulo 12

En este capítulo trabajamos con la estructura condicional `when`, una alternativa más clara y organizada al uso de múltiples sentencias `if-else`. La instrucción `when` permite realizar múltiples comparaciones de forma eficiente y legible. A lo largo de los ejercicios, utilizamos esta estructura para tomar decisiones basadas en coordenadas, promedios, rangos de pesos y rangos salariales. Aprendimos a usar `when` tanto como instrucción (para ejecutar bloques de código) como expresión (para devolver un valor directamente).

### Problema 1



The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar shows the project tree under "Android". The "app" module contains files like Main.kt, Proyecto59.kt, Proyecto60.kt, Proyecto61.kt, and Proyecto62.kt.
- Main.kt Code:** The code block below shows a function named `Proyecto59()` which prints a message, reads two integers from the user, and then uses a `when` expression to determine the quadrant of a point based on its coordinates `x` and `y`. The output of the `when` expression is then printed.
- Run Tab:** The bottom tab bar shows the "Run" tab is selected, and the "MainKt" configuration is active.
- Output Log:** The bottom pane displays the terminal output of the application's execution. It shows the program prompting for coordinates, receiving input, and then printing the result: "Primer cuadrante".
- Status Bar:** The bottom right shows the status bar with "10:34", "CRLF", "UTF-8", and other system information.

```
fun Proyecto59() {
    println("Proyecto 59: Coordenadas con condicional when")
    print("Ingrese coordenada x del punto:")
    val x = readLine()!!.toInt()
    print("Ingrese coordenada y del punto:")
    val y = readLine()!!.toInt()
    when {
        x > 0 && y > 0 -> println("Primer cuadrante")
        x < 0 && y > 0 -> println("Segundo cuadrante")
        x < 0 && y < 0 -> println("Tercer cuadrante")
        x > 0 && y < 0 -> println("Cuarto cuadrante")
        else -> println("El punto se encuentra en un eje")
    }
    println("")
}
```

Este programa solicita al usuario las coordenadas  $x$  e  $y$  de un punto y luego determina en qué cuadrante del plano cartesiano se encuentra. Si el punto no está en ningún cuadrante (es decir, si está sobre uno de los ejes), lo indica también. Utiliza la estructura `when` con condiciones booleanas para evaluar los rangos.

## Problema 2

The screenshot shows the Android Studio interface with the Main.kt file open. The code calculates the average of three grades and prints the result. It uses a conditional when block to determine the grade based on the average.

```
fun Proyecto60() {
    println("Proyecto 60: Promedio con condicional when")
    print("Ingrese primer nota:")
    val nota1 = readLine()!!.toInt()
    print("Ingrese segunda nota:")
    val nota2 = readLine()!!.toInt()
    print("Ingrese tercera nota:")
    val nota3 = readLine()!!.toInt()
    val promedio = (nota1 + nota2 + nota3) / 3
    when {
        promedio >= 7 -> print("Promocionado")
        promedio >= 4 -> print("Regular")
        else -> print("Libre")
    }
    println()
}
```

The run tab shows the output of the program:

```
[C:\Program Files\Android\Android Studio\jbr\bin\java.exe" ...]
Proyecto 60: Promedio con condicional when
Ingrese primer nota:9
Ingrese segunda nota:8
Ingrese tercera nota:7
Promocionado
Process finished with exit code 0
```

Este programa calcula el **promedio de tres notas** y luego clasifica al estudiante en una de tres categorías:

- **Promocionado** si el promedio es mayor o igual a 7.
- **Regular** si el promedio es mayor o igual a 4.
- **Libre** si es menor a 4.

Se usa la estructura when para tomar decisiones en función del valor del promedio.

## Problema 3

The screenshot shows the Android Studio interface with the Main.kt file open. The code counts the number of pieces based on their weight. It uses a conditional when block to count pieces in three categories: aptas (10.2 kg or more), superiores (9.8 kg or more), and inferiores (9.8 kg or less).

```
fun Proyecto61() {
    println("Proyecto 61: Contador con condicional when")
    var cant1 = 0
    var cant2 = 0
    var cant3 = 0
    do {
        print("Ingrese el peso de la pieza (0 para finalizar):")
        val peso = readLine()!!.toDouble()
        when {
            peso > 10.2 -> cant1++
            peso >= 9.8 -> cant2++
            peso > 0 -> cant3++
        }
    } while(peso != 0.0)
    println("Piezas aptas: $cant2")
    println("Piezas con un peso superior a 10.2: $cant1")
    println("Piezas con un peso inferior a 9.8: $cant3")
}
```

The run tab shows the output of the program:

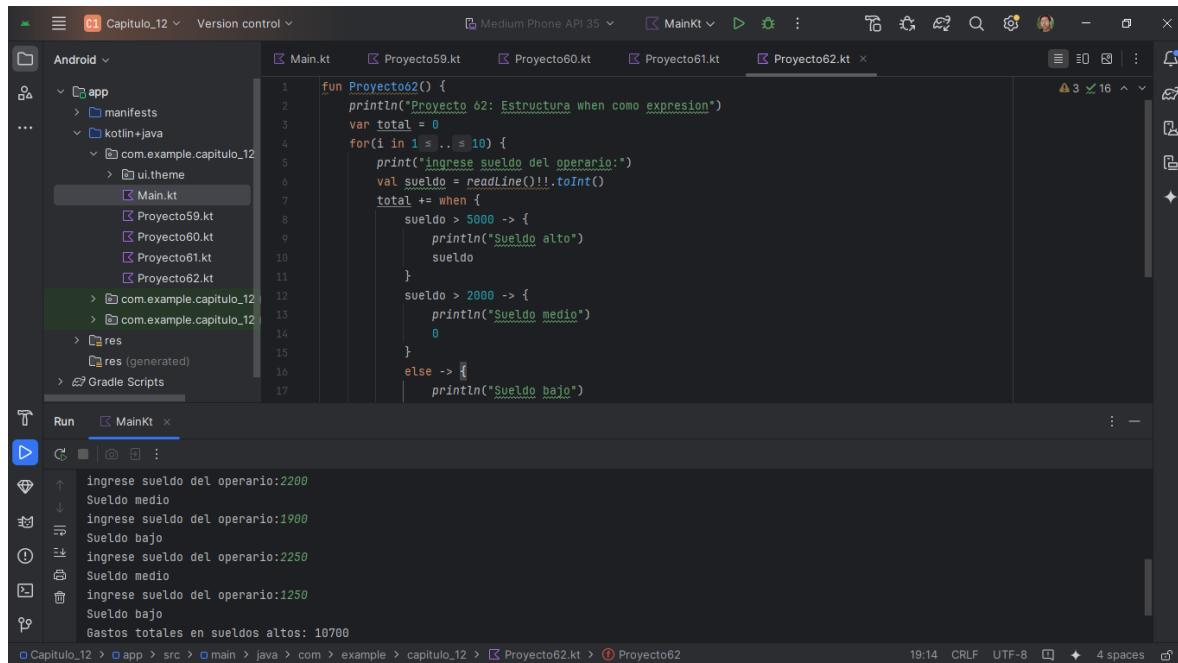
```
Proyecto 61: Contador con condicional when
Ingrese el peso de la pieza (0 para finalizar):10.3
Ingrese el peso de la pieza (0 para finalizar):9.9
Ingrese el peso de la pieza (0 para finalizar):9.7
Ingrese el peso de la pieza (0 para finalizar):0
Piezas aptas: 1
Piezas con un peso superior a 10.2: 1
Piezas con un peso inferior a 9.8: 1
Cantidad total de piezas procesadas: 3
```

Este programa analiza una serie de **pesos de piezas** ingresadas por el usuario. Cada pieza se clasifica según su peso:

- **Superior a 10.2** → cant1
- **Entre 9.8 y 10.2 inclusive** → cant2 (piezas "aptas")
- **Menor a 9.8** → cant3

El ingreso de datos continúa hasta que el usuario introduce el valor 0, lo cual finaliza el ciclo. Al final, se muestra un resumen con el conteo de piezas de cada categoría y el total procesado.

## Problema 4



The screenshot shows the Android Studio interface with the Main.kt file open. The code defines a function `Proyecto62()` that prints a welcome message, initializes a total sum to 0, and then loops from 1 to 10, asking for a salary input. It uses a `when` expression to add the salary to `total` if it's greater than 5000, and prints a message for each case: "Sueldo alto" for salaries above 5000, "Sueldo medio" for salaries between 2000 and 5000, and "Sueldo bajo" for salaries of 2000 or less. The run tab shows the console output with five entries of salary inputs (2200, 1900, 2250, 1250, 1250) and their corresponding classification messages. The bottom status bar shows the current time as 19:14 and file encoding as UTF-8.

```
fun Proyecto62() {
    println("Proyecto 62: Estructura when como expresión")
    var total = 0
    for(i in 1 .. 10) {
        println("Ingrese sueldo del operario:")
        val sueldo = readLine()!!.toInt()
        total += when {
            sueldo > 5000 -> {
                println("Sueldo alto")
                sueldo
            }
            sueldo > 2000 -> {
                println("Sueldo medio")
                0
            }
            else -> {
                println("Sueldo bajo")
            }
        }
    }
}
```

Este programa solicita los **sueldos de 10 operarios** y clasifica cada uno como:

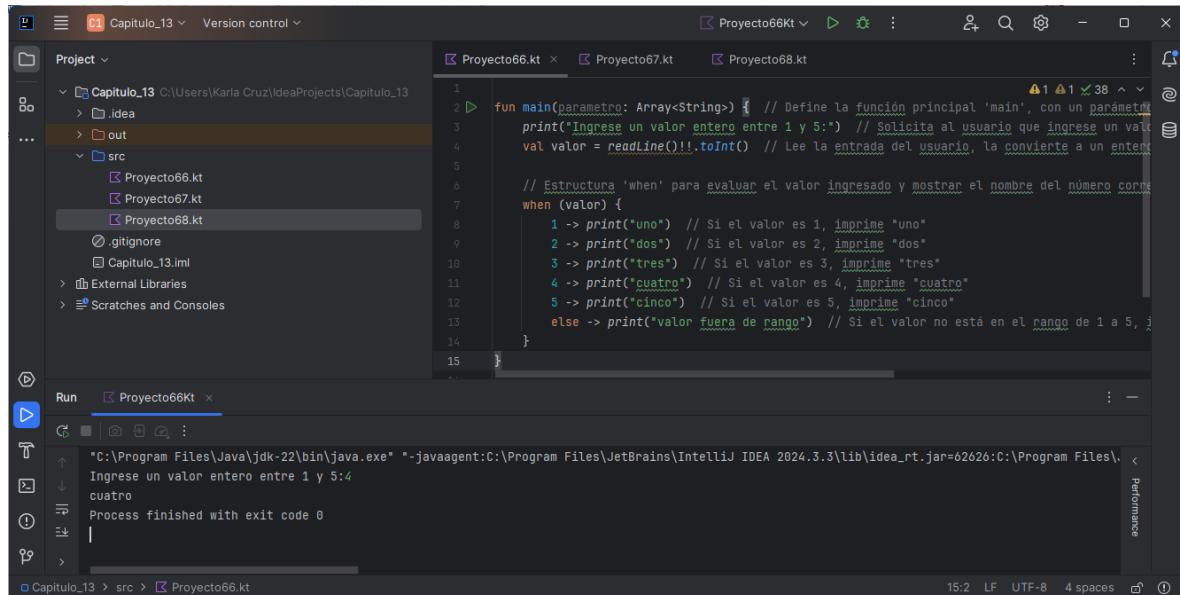
- **Alto** (>5000)
- **Medio** (entre 2001 y 5000)
- **Bajo** (2000 o menos)

Solo los sueldos **altos** se acumulan en la variable total. La estructura `when` se usa como **expresión**, ya que retorna directamente un valor (el sueldo o cero), lo que permite sumar condicionalmente solo los sueldos altos.

# Capítulo 13

## Problema 1

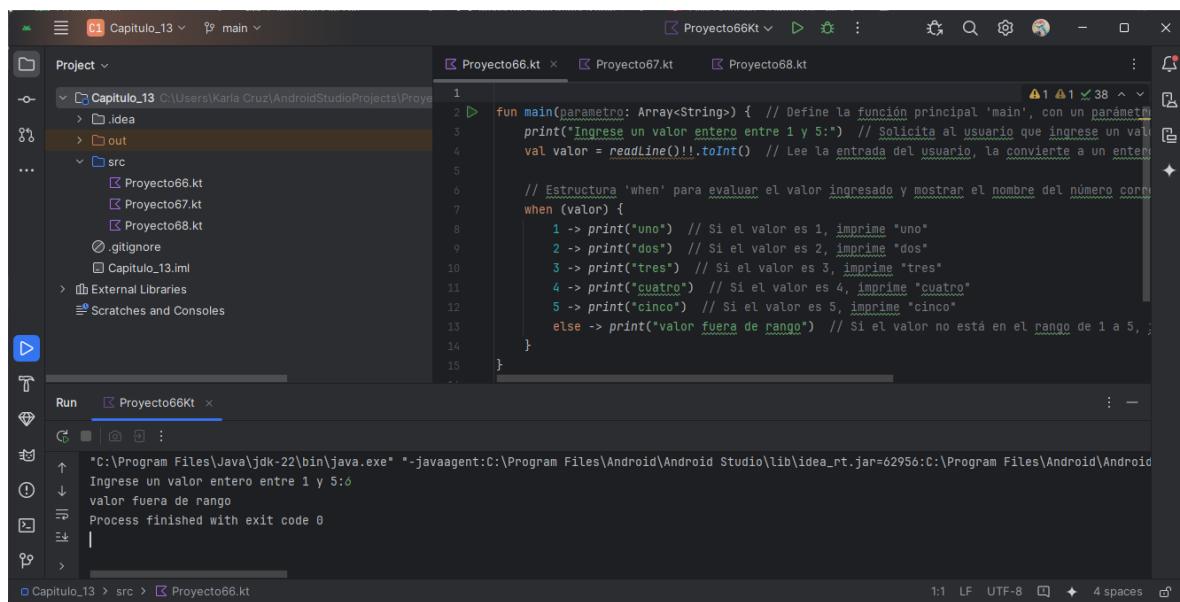
Para el proyecto 66 se solicita al usuario que ingrese un valor entero entre 1 y 5. Dependiendo del valor ingresado, el programa imprime el nombre del número correspondiente. Si el valor no está dentro del rango especificado (1-5), imprime un mensaje de error indicando que el valor está fuera de rango:



The screenshot shows the IntelliJ IDEA interface with the project 'Capítulo\_13' open. The 'src' directory contains three files: 'Proyecto66.kt', 'Proyecto67.kt', and 'Proyecto68.kt'. The 'Proyecto66.kt' file is selected and shown in the editor. The code defines a main function that prints a value between 1 and 5 and then uses a when expression to print the corresponding number name ('uno', 'dos', 'tres', 'cuatro', or 'cinco'). If the value is outside the range, it prints 'valor fuera de rango'. The run output shows the program was run with input '4', resulting in the output 'cuatro'.

```
fun main(parametro: Array<String>) { // Define la función principal 'main', con un parámetro
    print("Ingrese un valor entero entre 1 y 5:") // solicita al usuario que ingrese un valor
    val valor = readLine()!!.toInt() // Lee la entrada del usuario, la convierte a un entero

    // Estructura 'when' para evaluar el valor ingresado y mostrar el nombre del número correspondiente
    when (valor) {
        1 -> print("uno") // Si el valor es 1, imprime "uno"
        2 -> print("dos") // Si el valor es 2, imprime "dos"
        3 -> print("tres") // Si el valor es 3, imprime "tres"
        4 -> print("cuatro") // Si el valor es 4, imprime "cuatro"
        5 -> print("cinco") // Si el valor es 5, imprime "cinco"
        else -> print("valor fuera de rango") // Si el valor no está en el rango de 1 a 5, imprime "valor fuera de rango"
    }
}
```



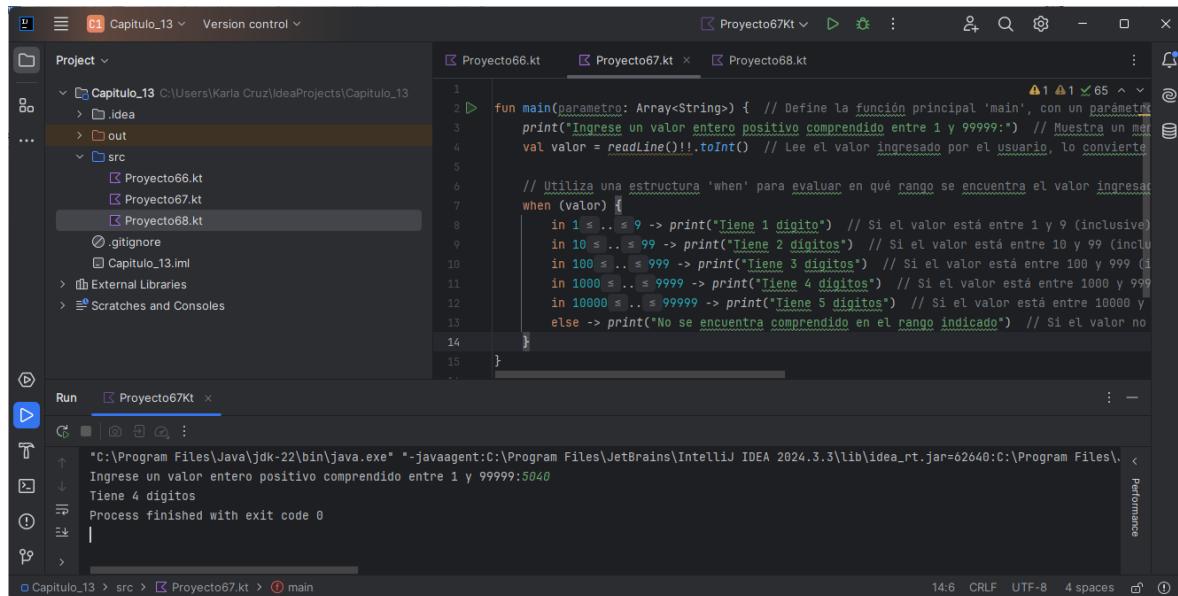
The screenshot shows the Android Studio interface with the project 'Capítulo\_13' open. The 'src' directory contains three files: 'Proyecto66.kt', 'Proyecto67.kt', and 'Proyecto68.kt'. The 'Proyecto66.kt' file is selected and shown in the editor. The code is identical to the one in IntelliJ IDEA, defining a main function that prints a value between 1 and 5 and then uses a when expression to print the corresponding number name ('uno', 'dos', 'tres', 'cuatro', or 'cinco'). If the value is outside the range, it prints 'valor fuera de rango'. The run output shows the program was run with input '4', resulting in the output 'valor fuera de rango'.

```
fun main(parametro: Array<String>) { // Define la función principal 'main', con un parámetro
    print("Ingrese un valor entero entre 1 y 5:") // solicita al usuario que ingrese un valor
    val valor = readLine()!!.toInt() // Lee la entrada del usuario, la convierte a un entero

    // Estructura 'when' para evaluar el valor ingresado y mostrar el nombre del número correspondiente
    when (valor) {
        1 -> print("uno") // Si el valor es 1, imprime "uno"
        2 -> print("dos") // Si el valor es 2, imprime "dos"
        3 -> print("tres") // Si el valor es 3, imprime "tres"
        4 -> print("cuatro") // Si el valor es 4, imprime "cuatro"
        5 -> print("cinco") // Si el valor es 5, imprime "cinco"
        else -> print("valor fuera de rango") // Si el valor no está en el rango de 1 a 5, imprime "valor fuera de rango"
    }
}
```

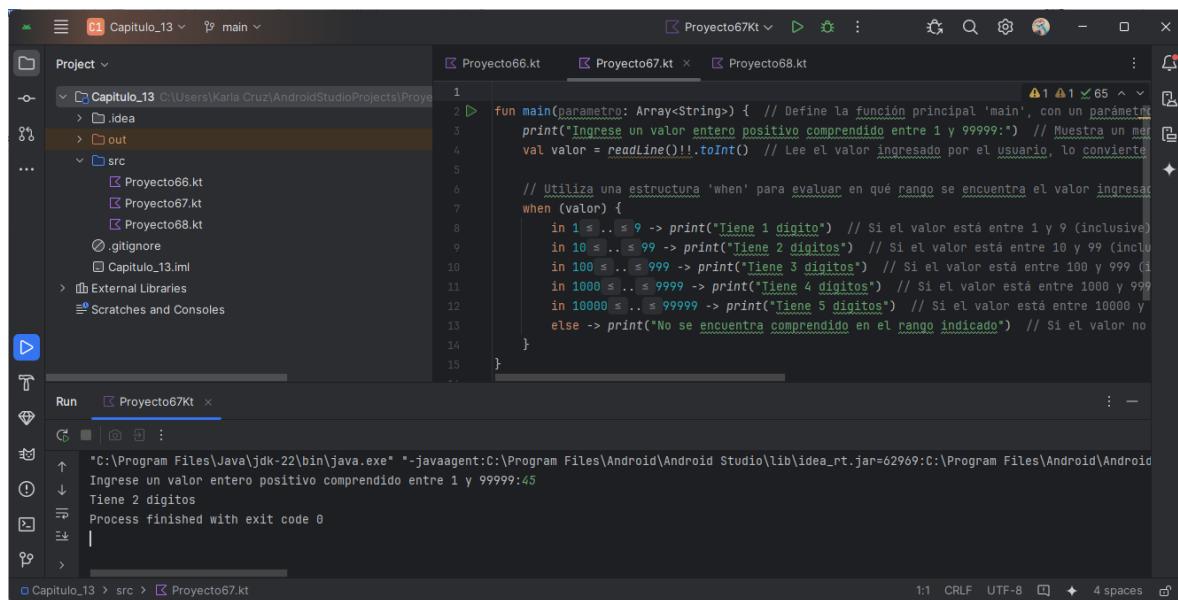
## Problema 2

En cambio, para el proyecto 67 se solicita al usuario ingresar un número entero positivo entre 1 y 99999 y luego determina cuántos dígitos tiene el número ingresado. Utiliza la estructura when para evaluar el valor y verificar en qué rango se encuentra:



```
fun main(parametro: Array<String>) { // Define la función principal 'main', con un parámetro
    print("Ingrese un valor entero positivo comprendido entre 1 y 99999: ") // Muestra un mensaje de solicitud
    val valor = readLine()!!.toInt() // Lee el valor ingresado por el usuario, lo convierte a entero

    // Utiliza una estructura 'when' para evaluar en qué rango se encuentra el valor ingresado
    when (valor) {
        in 1 .. 9 -> print("Tiene 1 dígito") // Si el valor está entre 1 y 9 (inclusive)
        in 10 .. 99 -> print("Tiene 2 dígitos") // Si el valor está entre 10 y 99 (inclusive)
        in 100 .. 999 -> print("Tiene 3 dígitos") // Si el valor está entre 100 y 999 (inclusive)
        in 1000 .. 9999 -> print("Tiene 4 dígitos") // Si el valor está entre 1000 y 9999 (inclusive)
        in 10000 .. 99999 -> print("Tiene 5 dígitos") // Si el valor está entre 10000 y 99999
        else -> print("No se encuentra comprendido en el rango indicado") // Si el valor no cumple con ninguno de los rangos anteriores
    }
}
```



```
fun main(parametro: Array<String>) { // Define la función principal 'main', con un parámetro
    print("Ingrese un valor entero positivo comprendido entre 1 y 99999: ") // Muestra un mensaje de solicitud
    val valor = readLine()!!.toInt() // Lee el valor ingresado por el usuario, lo convierte a entero

    // Utiliza una estructura 'when' para evaluar en qué rango se encuentra el valor ingresado
    when (valor) {
        in 1 .. 9 -> print("Tiene 1 dígito") // Si el valor está entre 1 y 9 (inclusive)
        in 10 .. 99 -> print("Tiene 2 dígitos") // Si el valor está entre 10 y 99 (inclusive)
        in 100 .. 999 -> print("Tiene 3 dígitos") // Si el valor está entre 100 y 999 (inclusive)
        in 1000 .. 9999 -> print("Tiene 4 dígitos") // Si el valor está entre 1000 y 9999 (inclusive)
        in 10000 .. 99999 -> print("Tiene 5 dígitos") // Si el valor está entre 10000 y 99999
        else -> print("No se encuentra comprendido en el rango indicado") // Si el valor no cumple con ninguno de los rangos anteriores
    }
}
```

## Problema 3

Por último, el proyecto 68 solicita al usuario que ingrese un valor entero 10 veces y, según el valor ingresado, incrementa contadores específicos para ciertos valores. Luego, muestra la cantidad de veces que se ingresaron ciertos números:

Project

Capítulo\_13 C:\Users\Karia Cruz\IdeaProjects\Capítulo\_13

.idea

out

src

Proyecto66.kt

Proyecto67.kt

Run

Proyecto68Kt

Projecto66.kt

Projecto67.kt

Projecto68.kt

```
fun main(parametro: Array<String>) { // Define la función principal 'main'
    // Muestra la cantidad de veces que se ingresó el número 0
    println("Cantidad de números 0 ingresados: $cant1")
    // Muestra la cantidad de veces que se ingresaron los números 1, 5 o 10
    println("Cantidad de números 1,5 o 10 ingresados: $cant2")
}
```

"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\lib\idea\_rt.jar=62681:C:\Program Files\

Ingrese un valor entero:5  
Ingrese un valor entero:10  
Ingrese un valor entero:1  
Ingrese un valor entero:0  
Ingrese un valor entero:0  
Ingrese un valor entero:1  
Ingrese un valor entero:5  
Ingrese un valor entero:10  
Ingrese un valor entero:0  
Ingrese un valor entero:0  
Cantidad de números 0 ingresados: 4  
Cantidad de números 1,5 o 10 ingresados: 6

Process finished with exit code 0

16:6 CRLF UTF-8 4 spaces

Project

Capítulo\_13 C:\Users\Karia Cruz\AndroidStudioProjects\Proyecto68

.idea

out

src

Proyecto66.kt

Proyecto67.kt

Proyecto68.kt

Run

Proyecto68Kt

Projecto66.kt

Projecto67.kt

Projecto68.kt

```
fun main(parametro: Array<String>) { // Define la función principal 'main', con un parámetro
    var cant1 = 0 // Declara la variable 'cant1' que llevará la cuenta de los números 0 ingresados
    var cant2 = 0 // Declara la variable 'cant2' que llevará la cuenta de los números 1, 5 o 10 ingresados

    // Inicia un bucle 'for' que se repite 10 veces (de 1 a 10)
    for(i in 1 .. 10) {
        print("Ingrese un valor entero:") // Muestra un mensaje solicitando al usuario que ingrese un valor entero
        val linea = readLine()!! // Lee el valor ingresado por el usuario, lo convierte a entero y lo guarda en 'linea'
        linea.toInt() // Convierte el valor de tipo String a entero y lo guarda en 'numero'
        if(numero == 0) {
            cant1++
        } else if(numero == 1 || numero == 5 || numero == 10) {
            cant2++
        }
    }
}
```

"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea\_rt.jar=62979:C:\Program Files\Android\Android

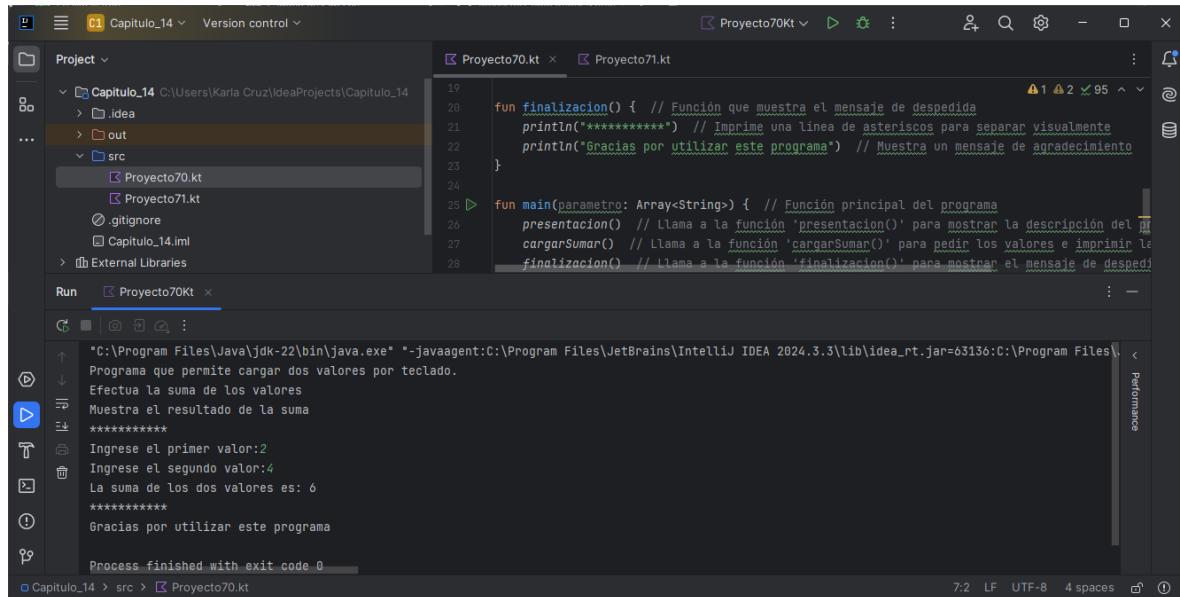
Ingrese un valor entero:10  
Ingrese un valor entero:0  
Ingrese un valor entero:5  
Ingrese un valor entero:1  
Ingrese un valor entero:0  
Ingrese un valor entero:5  
Ingrese un valor entero:10  
Ingrese un valor entero:0  
Ingrese un valor entero:0  
Cantidad de números 0 ingresados: 4  
Cantidad de números 1,5 o 10 ingresados: 6

1:1 CRLF UTF-8 4 spaces

# Capítulo 14

## Problema 1

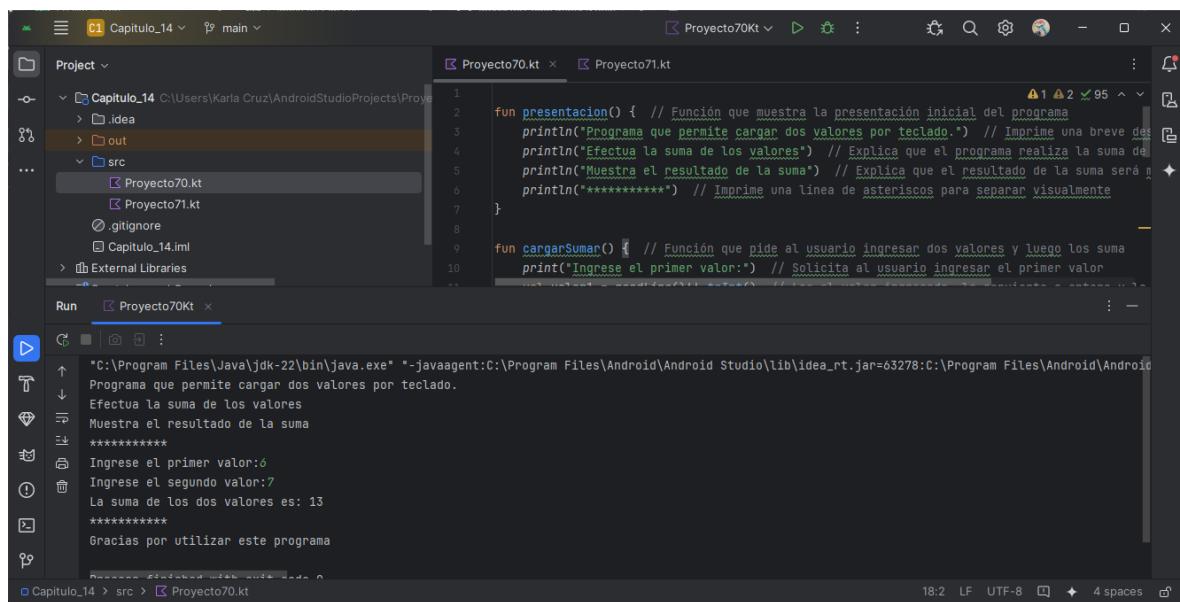
El proyecto 69 se propone en el capítulo anterior, pero para el proyecto 70 el programa está estructurado en varias funciones: presentacion(), cargarSumar(), y finalizacion(), que se encargan de diferentes tareas. La función principal main() organiza la ejecución de estas funciones:



```
fun finalizacion() { // Función que muestra el mensaje de despedida
    println("*****") // Imprime una línea de asteriscos para separar visualmente
    println("Gracias por utilizar este programa") // Muestra un mensaje de agradecimiento
}

fun main(parametro: Array<String>) { // Función principal del programa
    presentacion() // Llama a la función 'presentacion()' para mostrar la descripción del
    cargarSumar() // Llama a la función 'cargarSumar()' para pedir los valores e imprimir la
    finalizacion() // Llama a la función 'finalizacion()' para mostrar el mensaje de despedida
}

fun cargarSumar() {
    println("Programa que permite cargar dos valores por teclado.")
    println("Efectúa la suma de los valores")
    println("Muestra el resultado de la suma")
    println("*****")
    println("Ingrese el primer valor:2")
    println("Ingrese el segundo valor:4")
    println("La suma de los dos valores es: 6")
    println("*****")
    println("Gracias por utilizar este programa")
}
```

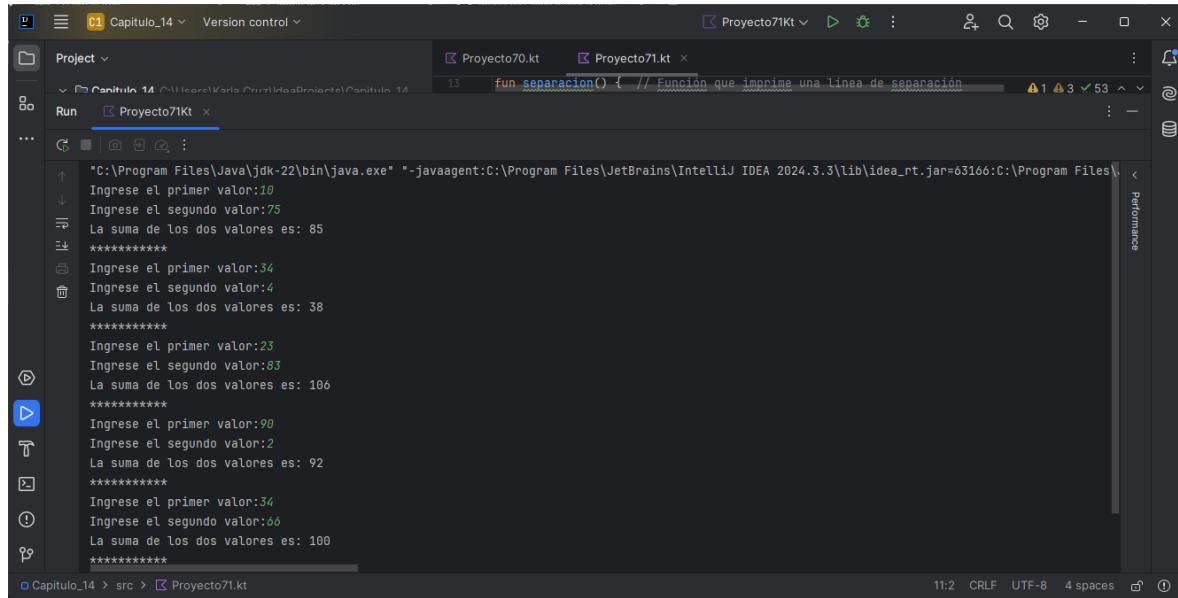


```
fun presentacion() { // Función que muestra la presentación inicial del programa
    println("Programa que permite cargar dos valores por teclado.") // Imprime una breve descripción del programa
    println("Efectúa la suma de los valores") // Explica que el programa realiza la suma de los valores
    println("Muestra el resultado de la suma") // Explica que el resultado de la suma será mostrado en la consola
    println("*****") // Imprime una línea de asteriscos para separar visualmente
}

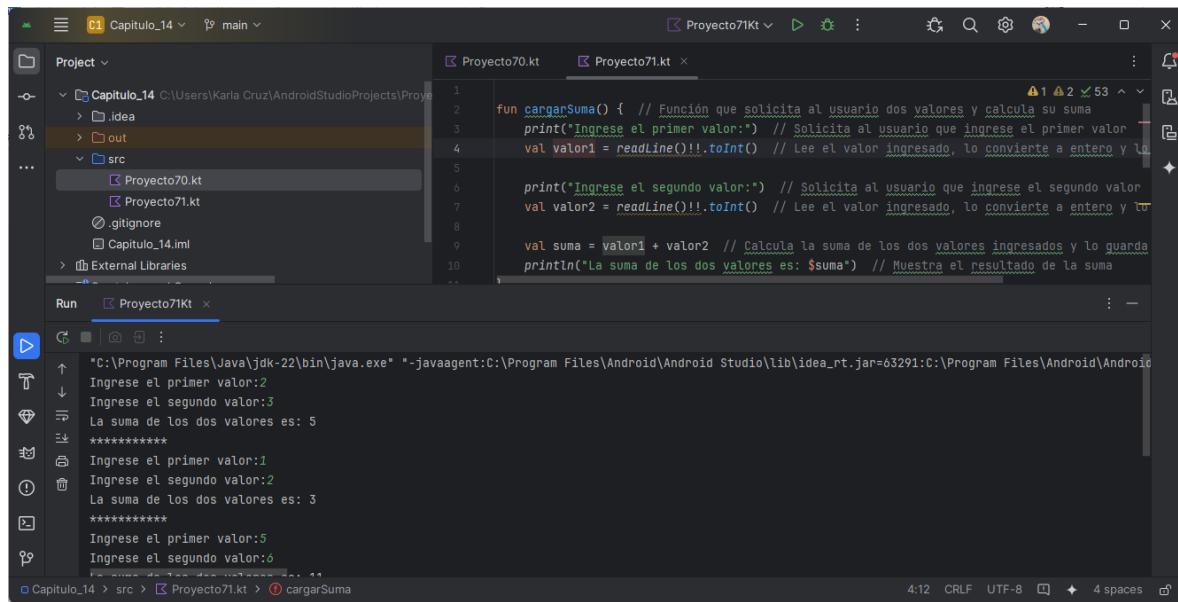
fun cargarSumar() {
    println("Programa que pide al usuario ingresar dos valores y luego los suma")
    print("Ingrese el primer valor:") // Solicita al usuario ingresar el primer valor
    val num1 = readLine() // Lee el primer valor ingresado por el usuario
    print("Ingrese el segundo valor:") // Solicita al usuario ingresar el segundo valor
    val num2 = readLine() // Lee el segundo valor ingresado por el usuario
    val resultado = num1.toInt() + num2.toInt() // Calcula la suma de los dos valores
    println("La suma de los dos valores es: $resultado")
    println("*****")
    println("Gracias por utilizar este programa")
}
```

## Problema 2

Para el proyecto 71 el programa contiene dos funciones principales: cargarSuma() y separacion(), además de un ciclo for dentro de la función principal main() que ejecuta estas funciones varias veces:



```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\lib\idea_rt.jar=63166:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\bin" Capitulo_14\src\main\kotlin\Capitulo_14\Proyecto71Kt.kt
Ingrese el primer valor:10
Ingrese el segundo valor:75
La suma de los dos valores es: 85
*****
Ingrese el primer valor:34
Ingrese el segundo valor:4
La suma de los dos valores es: 38
*****
Ingrese el primer valor:23
Ingrese el segundo valor:83
La suma de los dos valores es: 106
*****
Ingrese el primer valor:99
Ingrese el segundo valor:2
La suma de los dos valores es: 92
*****
Ingrese el primer valor:34
Ingrese el segundo valor:66
La suma de los dos valores es: 100
*****
```

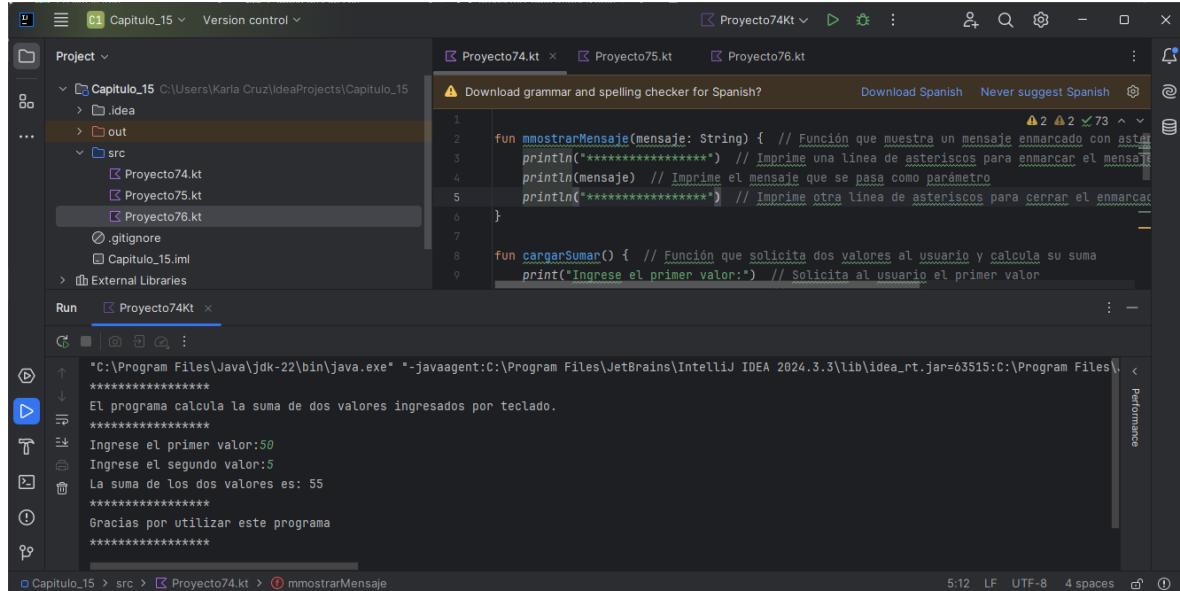


```
fun cargarSuma() { // Función que solicita al usuario dos valores y calcula su suma
    print("Ingrese el primer valor:") // Solicita al usuario que ingrese el primer valor
    val valor1 = readLine()!!.toInt() // Lee el valor ingresado, lo convierte a entero y lo guarda
    print("Ingrese el segundo valor:") // Solicita al usuario que ingrese el segundo valor
    val valor2 = readLine()!!.toInt() // Lee el valor ingresado, lo convierte a entero y lo guarda
    val suma = valor1 + valor2 // Calcula la suma de los dos valores ingresados y lo guarda
    println("La suma de los dos valores es: $suma") // Muestra el resultado de la suma
}
```

# Capítulo 15

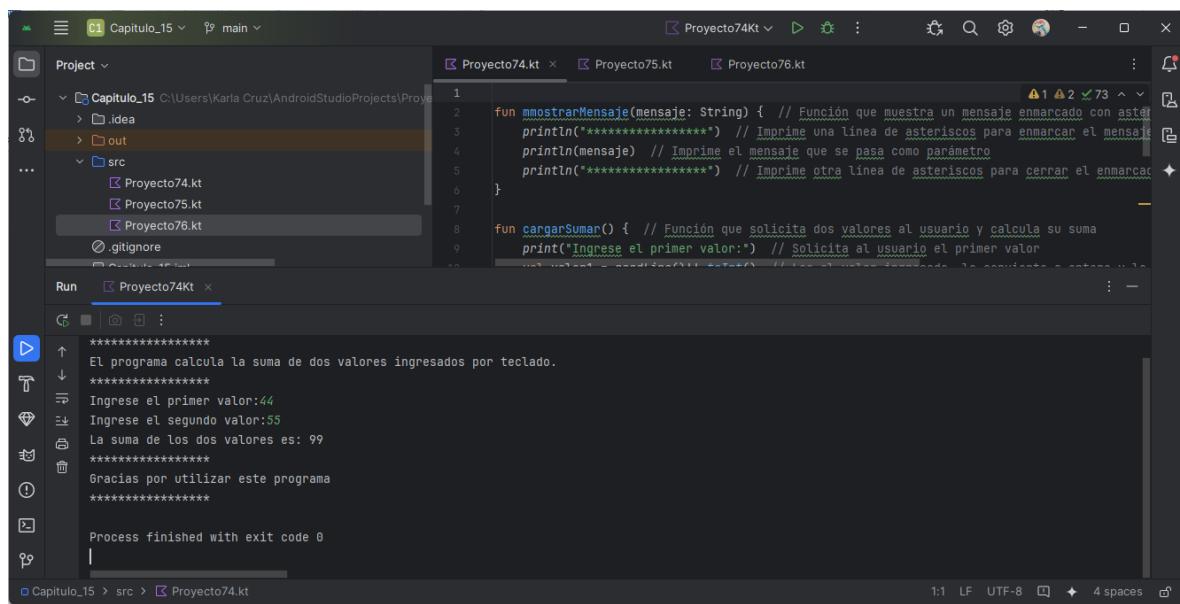
## Problema 1

Los proyectos 72 y 73 son propuestas. En el proyecto 74 el programa define dos funciones principales: mostrarMensaje() y cargarSumar(), donde la función principal main() las utiliza para mostrar mensajes y realizar una operación de suma:



```
fun mostrarMensaje(mensaje: String) { // Función que muestra un mensaje enmarcado con asteriscos
    println("*****") // Imprime una línea de asteriscos para enmarcar el mensaje
    println(mensaje) // Imprime el mensaje que se pasa como parámetro
    println("*****") // Imprime otra línea de asteriscos para cerrar el enmarcado
}

fun cargarSumar() { // Función que solicita dos valores al usuario y calcula su suma
    print("Ingrese el primer valor:") // solicita al usuario el primer valor
    val num1 = readLine() // Lee el primer valor ingresado por teclado
    print("Ingrese el segundo valor:") // solicita al usuario el segundo valor
    val num2 = readLine() // Lee el segundo valor ingresado por teclado
    val resultado = num1.toInt() + num2.toInt() // Calcula la suma de los dos valores
    println("La suma de los dos valores es: $resultado")
}
```



```
fun mostrarMensaje(mensaje: String) { // Función que muestra un mensaje enmarcado con asteriscos
    println("*****") // Imprime una línea de asteriscos para enmarcar el mensaje
    println(mensaje) // Imprime el mensaje que se pasa como parámetro
    println("*****") // Imprime otra línea de asteriscos para cerrar el enmarcado
}

fun cargarSumar() { // Función que solicita dos valores al usuario y calcula su suma
    print("Ingrese el primer valor:") // solicita al usuario el primer valor
    val num1 = readLine() // Lee el primer valor ingresado por teclado
    print("Ingrese el segundo valor:") // solicita al usuario el segundo valor
    val num2 = readLine() // Lee el segundo valor ingresado por teclado
    val resultado = num1.toInt() + num2.toInt() // Calcula la suma de los dos valores
    println("La suma de los dos valores es: $resultado")
}
```

## Problema 2

En el proyecto 75 el programa está diseñado para pedir tres valores al usuario y luego determinar cuál de esos tres es el mayor. La función mostrarMayor() realiza la comparación, y la función principal main() solicita los valores al usuario y llama a mostrarMayor():

The screenshot shows the IntelliJ IDEA interface with the project 'Capítulo\_15' open. The 'src' directory contains three files: 'Proyecto74.kt', 'Proyecto75.kt', and 'Proyecto76.kt'. The 'Run' tool window is active, showing the output of running 'Proyecto75Kt'. The terminal pane displays:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\lib\idea_rt.jar=63526:C:\Program Files\  
Ingrese primer valor:646  
Ingrese segundo valor:34  
Ingrese tercer valor:23  
Mayor:646  
Process finished with exit code 0
```

The status bar at the bottom right indicates: 12:1 CRLF UTF-8 4 spaces.

```
2 fun mostrarMayor(v1: Int, v2: Int, v3: Int) { // Función que recibe tres  
3     if (v2 > v3) // Si 'v2' es mayor que 'v3'  
4         print(v2) // Muestra 'v2' como el mayor  
5     else // Si ninguna de las condiciones anteriores se cumple  
6         print(v3) // Muestra 'v3' como el mayor  
7 }  
8  
13 fun main(parametro: Array<String>) { // Función principal del programa  
14     print("Ingrese primer valor:") // Solicita al usuario el primer valor  
15     val valor1 = readLine()!!.toInt() // Lee el valor ingresado, lo convierte a entero y lo
```

The screenshot shows the Android Studio interface with the project 'Capítulo\_15' open. The 'src' directory contains three files: 'Proyecto74.kt', 'Proyecto75.kt', and 'Proyecto76.kt'. The 'Run' tool window is active, showing the output of running 'Proyecto75Kt'. The terminal pane displays:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=63681:C:\Program Files\Android\Android  
Ingrese primer valor:83  
Ingrese segundo valor:11  
Ingrese tercer valor:20  
Mayor:83  
Process finished with exit code 0
```

The status bar at the bottom right indicates: 1:1 CRLF UTF-8 4 spaces.

```
1 fun mostrarMayor(v1: Int, v2: Int, v3: Int) { // Función que recibe tres valores enteros  
2     print("Mayor:") // Imprime la palabra "Mayor:" antes de mostrar el número mayor  
3     if (v1 > v2 && v1 > v3) // Si 'v1' es mayor que 'v2' y 'v3'  
4         println(v1) // Muestra 'v1' como el mayor  
5     else // Si la condición anterior no es verdadera  
6         if (v2 > v3) // Si 'v2' es mayor que 'v3'  
7             print(v2) // Muestra 'v2' como el mayor  
8         else // Si ninguna de las condiciones anteriores se cumple
```

### Problema 3

En el proyecto 76 se calcula y muestra el perímetro o la superficie de un cuadrado, según lo que el usuario elija:

The screenshot shows the IntelliJ IDEA interface with the project structure on the left and the code editor on the right. The code in `Proyecto76.kt` is as follows:

```
fun main(parametro: Array<String>) { // Función principal del programa
    val la = readLine()!!.toInt() // Lee el valor ingresado y lo convierte a un entero (Int)
    print("Quiere calcular el perímetro o la superficie[ingresar texto: perímetro/superficie]")
    var respuesta = readLine()!! // Lee la respuesta del usuario (perímetro o superficie) y la convierte a una cadena
    when (respuesta) { // Utiliza una expresión 'when' para comparar la respuesta del usuario
        "perímetro" -> mostrarPerimetro(la) // Si el usuario ingresa "perímetro", llama a la función mostrarPerimetro con el lado
        "superficie" -> mostrarSuperficie(la) // Si el usuario ingresa "superficie", llama a la función mostrarSuperficie con el lado
    }
}
```

The run output shows the program prompting the user for input and then printing the result:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\lib\idea_rt.jar=63534:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.3\bin"
Ingresé el valor del lado de un cuadrado:8
Quiere calcular el perímetro o la superficie[ingresar texto: perímetro/superficie]perímetro
El perímetro es 32
Process finished with exit code 0
```

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code in `Proyecto76.kt` is as follows:

```
fun mostrarPerimetro(lado: Int) { // Función que calcula y muestra el perímetro de un cuadrado
    val perimetro = lado * 4 // Calcula el perímetro multiplicando el lado por 4
    println("El perímetro es $perimetro") // Imprime el resultado del perímetro
}

fun mostrarSuperficie(lado: Int) { // Función que calcula y muestra la superficie de un cuadrado
    val superficie = lado * lado // Calcula la superficie (área) multiplicando el lado por sí mismo
    println("La superficie es $superficie") // Imprime el resultado de la superficie
}

fun main(parametro: Array<String>) { // Función principal del programa
    val lado: Int = parametro[0].toInt()
    if (parametro.size > 1) {
        if (parametro[1] == "perímetro") {
            mostrarPerimetro(lado)
        } else if (parametro[1] == "superficie") {
            mostrarSuperficie(lado)
        }
    } else {
        println("No se ingresó la opción")
    }
}
```

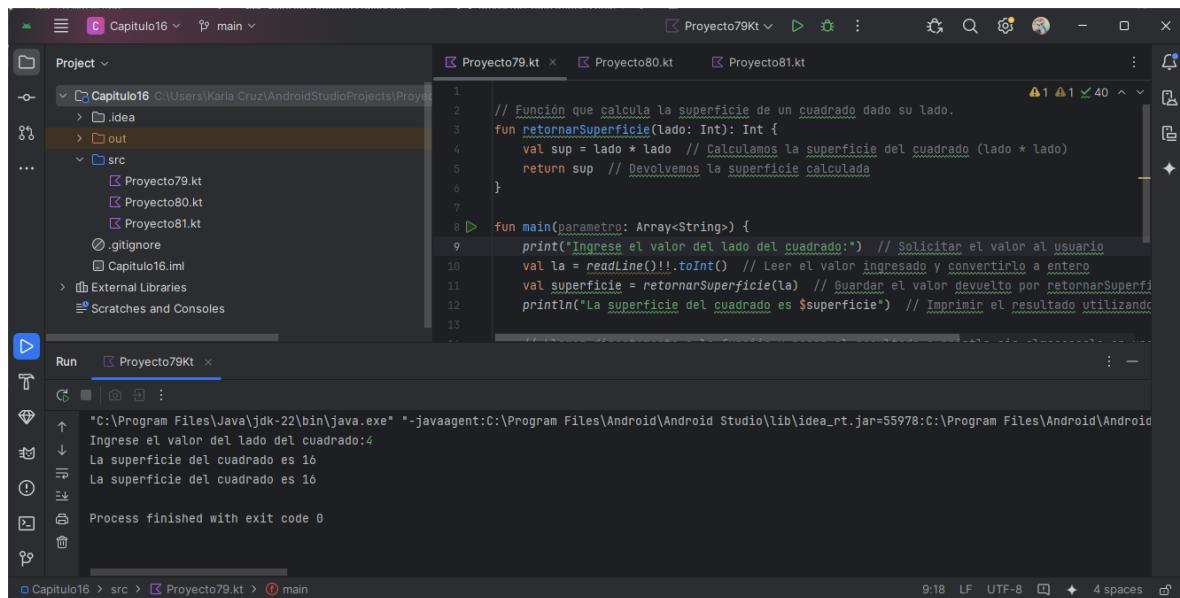
The run output shows the program prompting the user for input and then printing the result:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=63686:C:\Program Files\Android\Android Studio
Ingresé el valor del lado de un cuadrado:2
Quiere calcular el perímetro o la superficie[ingresar texto: perímetro/superficie]superficie
La superficie es 4
Process finished with exit code 0
```

## Capítulo 16

### Problema 1

En el proyecto 79 se calcula la superficie de un cuadrado a partir del valor ingresado por el usuario. Se define la función `retornarSuperficie`, que recibe un parámetro `lado` de tipo `Int` y retorna un entero calculado como `lado * lado`, utilizando la palabra clave `return`. Esta función encapsula el cálculo y su variable interna `sup` no es accesible fuera de ella. En `main`, se solicita al usuario el valor del lado, se convierte a entero y se pasa a la función. El resultado se almacena en la variable `superficie` y se muestra por pantalla. También se muestra directamente llamando a la función dentro de `println` con `${...}`, aprovechando la interpolación de cadenas de Kotlin.



The screenshot shows the Android Studio interface with the project 'Capítulo16' open. The code editor displays 'Proyecto79Kt.kt' with the following content:

```
1 // Función que calcula la superficie de un cuadrado dado su lado.
2 fun retornarSuperficie(lado: Int): Int {
3     val sup = lado * lado // Calculamos la superficie del cuadrado (lado * lado)
4     return sup // Devolvemos la superficie calculada
5 }
6
7
8 fun main(parametro: Array<String>) {
9     print("Ingrese el valor del lado del cuadrado:") // Solicitar el valor al usuario
10    val la = readLine()!!.toInt() // Leer el valor ingresado y convertirlo a entero
11    val superficie = retornarSuperficie(la) // Guardar el valor devuelto por retornarSuperficie
12    println("La superficie del cuadrado es $superficie") // Imprimir el resultado utilizando la interpolación de cadenas
13 }
```

The run tab shows the output of the program:

```
*C:\Program Files\Java\jdk-22\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=55978:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8
Ingresese el valor del lado del cuadrado:4
La superficie del cuadrado es 16
La superficie del cuadrado es 16
Process finished with exit code 0
```

### Problema 2

En el proyecto 80 se define una función llamada `retornarMayor` que recibe dos números enteros como parámetros y retorna el mayor de ellos. Utiliza una estructura condicional `if-else` para comparar ambos valores: si `v1` es mayor que `v2`, devuelve `v1`; de lo contrario, devuelve `v2`. La palabra clave `return` se usa para indicar el valor que la función debe retornar, y una vez ejecutada, la función finaliza y regresa al punto desde donde fue llamada. En la función `main`, se solicita al usuario que ingrese dos números enteros, que se leen desde la consola y se convierten a enteros. Luego, se llama a la función `retornarMayor` con esos dos valores, y el resultado se muestra utilizando interpolación de cadenas.

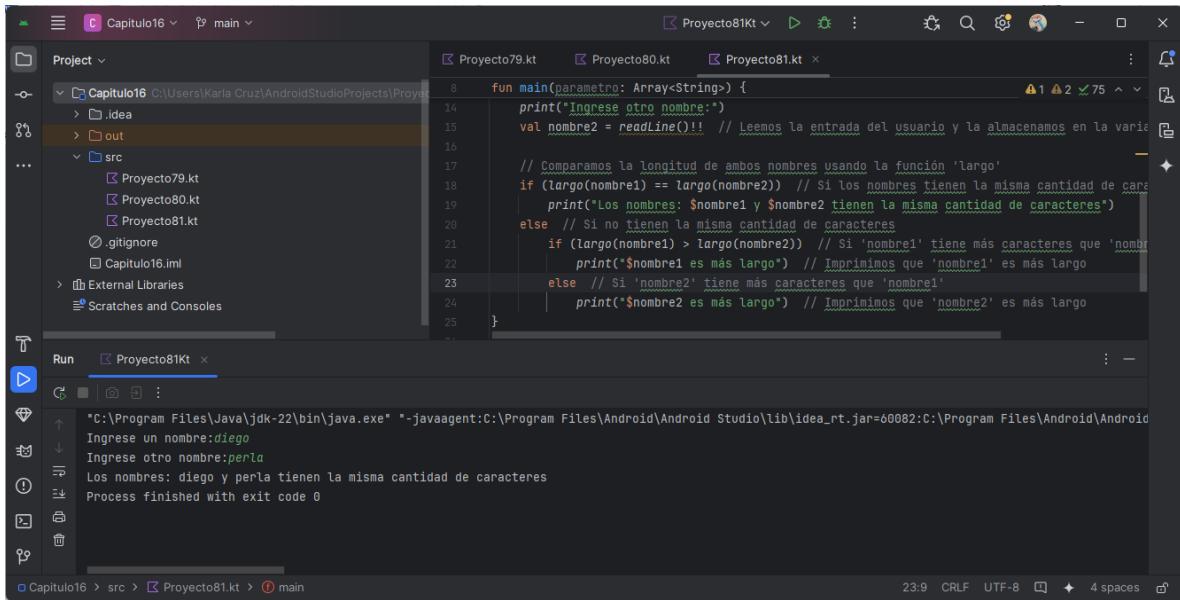
```
// Definimos la función 'retornarMayor' que toma dos parámetros enteros (v1 y v2) y devuelve el mayor
fun retornarMayor(v1: Int, v2: Int): Int {
    if (v1 > v2) // Comprobamos si el primer valor es mayor que el segundo
        return v1 // Si v1 es mayor, devolvemos v1
    else
        return v2 // Si no, devolvemos v2 (es decir, v2 es mayor o igual a v1)
}

fun main(parametro: Array<String>) {
    // Solicitar al usuario que ingrese el primer valor
    print("Ingrese el primer valor:")
    val valor1 = readLine()!!.toInt() // Leemos la entrada del usuario y la convertimos a entero
    // Solicitar al usuario que ingrese el segundo valor
    print("Ingrese el segundo valor:")
    val valor2 = readLine()!!.toInt() // Leemos la entrada del usuario y la convertimos a entero
    // Comparar los valores
    if (valor1 > valor2)
        println("El mayor entre $valor1 y $valor2 es $valor1")
    else
        println("El mayor entre $valor1 y $valor2 es $valor2")
}
```

### Problema 3

En cuanto al proyecto 81 se define una función llamada largo que recibe como parámetro un String y devuelve un valor entero que representa la cantidad de caracteres de esa cadena. Para ello, utiliza la propiedad length, propia de la clase String en Kotlin, la cual permite saber cuántos caracteres tiene una palabra o frase. Esta función retorna ese valor utilizando la palabra clave return, lo que significa que cuando se llama a la función, se detiene su ejecución y se envía el valor calculado al lugar donde fue llamada.

En la función main, el programa solicita al usuario que ingrese dos nombres. Ambos se almacenan como cadenas de texto en las variables nombre1 y nombre2. Luego, se llama a la función largo dos veces, una con cada nombre, para calcular la longitud de cada uno. Posteriormente, se comparan los valores devueltos por la función para determinar cuál nombre tiene más caracteres. Si ambos nombres tienen la misma longitud, se muestra un mensaje indicando que tienen igual cantidad de caracteres. Si uno es más largo que el otro, se imprime cuál es el más largo.



```
fun main(parametro: Array<String>) {
    print("Ingrese otro nombre:")
    val nombre2 = readLine()!! // Leemos la entrada del usuario y la almacenamos en la variable nombre2
    // Comparamos la longitud de ambos nombres usando la función 'largo'
    if (largo(nombre1) == largo(nombre2)) // Si los nombres tienen la misma cantidad de caracteres
        print("Los nombres: $nombre1 y $nombre2 tienen la misma cantidad de caracteres")
    else // Si no tienen la misma cantidad de caracteres
        if (largo(nombre1) > largo(nombre2)) // Si 'nombre1' tiene más caracteres que 'nombre2'
            print("$nombre1 es más largo") // Imprimimos que 'nombre1' es más largo
        else // Si 'nombre2' tiene más caracteres que 'nombre1'
            print("$nombre2 es más largo") // Imprimimos que 'nombre2' es más largo
}
```

## Capítulo 17

### Problema 1

En el proyecto 85 se define una función llamada `retornarSuperficie` que calcula y devuelve la superficie de un cuadrado, recibiendo como parámetro el valor de uno de sus lados. Como la función tiene una sola expresión, se puede escribir en una sola línea con el operador `=` en lugar de usar llaves y la palabra `return`.

Además, no es necesario indicar explícitamente el tipo de dato que retorna la función, ya que el compilador de Kotlin puede deducirlo automáticamente a partir de la operación `lado * lado`, que da como resultado un `Int`. Sin embargo, si se desea, también se puede indicar el tipo de retorno de manera explícita escribiendo `: Int`.

En la función `main`, se solicita al usuario que ingrese el valor del lado del cuadrado, se convierte a entero, y se pasa como argumento a la función `retornarSuperficie`. El resultado se muestra directamente utilizando interpolación de cadenas, insertando el valor devuelto dentro del mensaje que se imprime en pantalla.

The screenshot shows the Android Studio interface with the Project tool window on the left and the Editor tool window on the right. The Editor window displays the code for `Proyecto85.kt`. The code defines a function `retornarSuperficie` that takes an integer parameter `lado` and returns its square. It then reads a value from the user and prints the result.

```
// Definimos una función que recibe un parámetro de tipo Int (el valor del lado del cuadrado)
// y retorna la superficie (el área del cuadrado) que se calcula como lado * lado.
fun retornarSuperficie(lado: Int) = lado * lado // Sintaxis concisa de una función de una sola línea

fun main(parametro: Array<String>) {
    // Solicitar al usuario que ingrese el valor del lado del cuadrado.
    print("Ingrese el valor del lado del cuadrado:")
    val la = readLine()!!.toInt() // Leemos la entrada del usuario y la convertimos a un entero.

    // Llamamos a la función 'retornarSuperficie' y mostramos el resultado en pantalla.
    // El valor devuelto por la función se inserta en el String mediante la interpolación de cadenas.
    println("La superficie del cuadrado es ${retornarSuperficie(la)}")
}
```

## Problema 2

En el proyecto 86 se define una función llamada “`retornarMayor`” que recibe dos enteros y devuelve el mayor usando una expresión `if` en una sola línea, lo que hace el código más conciso. En `main`, se solicitan dos valores al usuario, se llama a la función con esos datos y se imprime cuál es el mayor. Esta forma simplificada es equivalente a usar `if` con `return`, pero más directa.

The screenshot shows the Android Studio interface with the Project tool window on the left and the Editor tool window on the right. The Editor window displays the code for `Proyecto86.kt`. The code defines a function `retornarMayor` that takes two integers `v1` and `v2` and returns the greater one using an expression `if`. It then reads two values from the user and prints the result.

```
// Definimos una función que recibe dos parámetros enteros y retorna el mayor de los dos.
// La función utiliza una expresión if para decidir cuál es el mayor.
fun retornarMayor(v1: Int, v2: Int) = if (v1 > v2) v1 else v2 // Sintaxis concisa con una sola línea

fun main(parametro: Array<String>) {
    // Solicitar al usuario que ingrese el primer valor.
    print("Ingrese el primer valor:")
    val valor1 = readLine()!!.toInt() // Leemos la entrada del usuario y la convertimos a un entero.

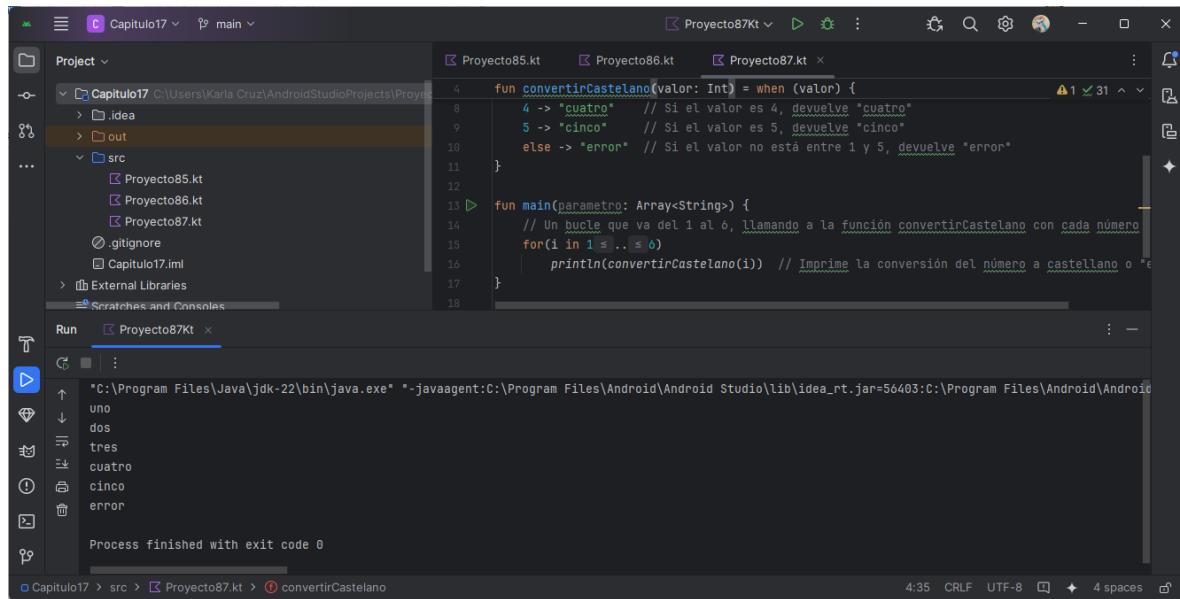
    // Solicitar al usuario que ingrese el segundo valor.
    print("Ingrese el segundo valor:")
    val valor2 = readLine()!!.toInt() // Leemos la entrada del usuario y la convertimos a un entero.

    // Llamamos a la función retornarMayor y mostramos el valor mayor entre los dos números.
    println("El mayor entre $valor1 y $valor2 es ${retornarMayor(valor1, valor2)}")
}
```

## Problema 3

Por su parte, en el proyecto 87 se define una función llamada “`convertirCastellano`” que toma un número entero y devuelve su equivalente en castellano si está entre 1 y 5. Si el número no está en ese rango, devuelve "error". En la función `main`, se

utiliza un bucle que recorre los números del 1 al 6 y muestra en pantalla su conversión llamando a la función. Así, los primeros cinco números se traducen y el sexto genera un mensaje de error.



The screenshot shows the Android Studio interface with the following details:

- Project View:** Shows the project structure under "Capítulo17".
- Main Activity (Projeto87Kt):** Contains the following Kotlin code:

```
fun convertirCastellano(valor: Int) = when (valor) {  
    4 -> "cuatro" // Si el valor es 4, devuelve "cuatro"  
    5 -> "cinco" // Si el valor es 5, devuelve "cinco"  
    else -> "error" // Si el valor no está entre 1 y 5, devuelve "error"  
}  
  
fun main(parametro: Array<String>) {  
    // Un bucle que va del 1 al 6, llamando a la función convertirCastellano con cada número  
    for(i in 1 .. 6)  
        println(convertirCastellano(i)) // Imprime la conversión del número a castellano o "error"  
}
```
- Run Tab:** Set to "Projeto87Kt".  
Output window shows:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=56403:C:\Program Files\Android\Android Studio" Capítulo17  
uno  
dos  
tres  
cuatro  
cinco  
error  
Process finished with exit code 0
```
- Status Bar:** Shows the time as 4:35, file encoding as CRLF, and character set as UTF-8.

## Capítulo 18

### Problema 1

En el proyecto 92 se define una función llamada “`tituloSubrayado`” que permite imprimir un texto como título y subrayarlo con un carácter específico. La función recibe dos parámetros: el primero es un `String` obligatorio que representa el título a mostrar, y el segundo es un `String` opcional que indica el carácter con el que se subrayará dicho título. Este segundo parámetro tiene un valor por defecto de “`*`”, lo cual significa que, si no se especifica un carácter en la llamada a la función, se usará automáticamente el asterisco como subrayado.

Dentro de la función, primero se imprime el título recibido. Luego, utilizando un bucle `for`, se imprime el carácter de subrayado tantas veces como la cantidad de letras del título. Finalmente, se imprime un salto de línea para separar visualmente el resultado de cualquier contenido que se muestre después.

En la función `main`, que es el punto de entrada del programa, se hacen dos llamadas a `tituloSubrayado`. La primera utiliza solo un argumento: “Sistema de Administración”, por lo que se aplica el carácter por defecto “`*`” para el subrayado. En la segunda llamada, se proporciona un segundo argumento “`-`”, por lo que el subrayado se realiza con guiones.

```
fun tituloSubrayado(titulo: String, caracter: String = "*") {
    // Imprime un salto de linea para separar visualmente el titulo del siguiente contenido
    println()
}

// Función principal donde se ejecuta el programa
fun main(parametro: Array<String>) {
    // Llama a la función 'tituloSubrayado' con el título "Sistema de Administración"
    // Como no se proporciona un segundo parámetro, usa el valor por defecto "*"
    tituloSubrayado( titulo: "Sistema de Administración")

    // Llama a la función 'tituloSubrayado' con el título "Ventas"
    // En este caso, el subrayado se hará con el carácter "-" en lugar de "*"
}
```

## Capítulo 19

### Problema 1

En el proyecto 94 se define una función llamada “calcularSueldo”, que recibe el nombre de un empleado, el pago por hora y la cantidad de horas trabajadas. Con estos datos, calcula el sueldo y muestra un mensaje con toda la información.

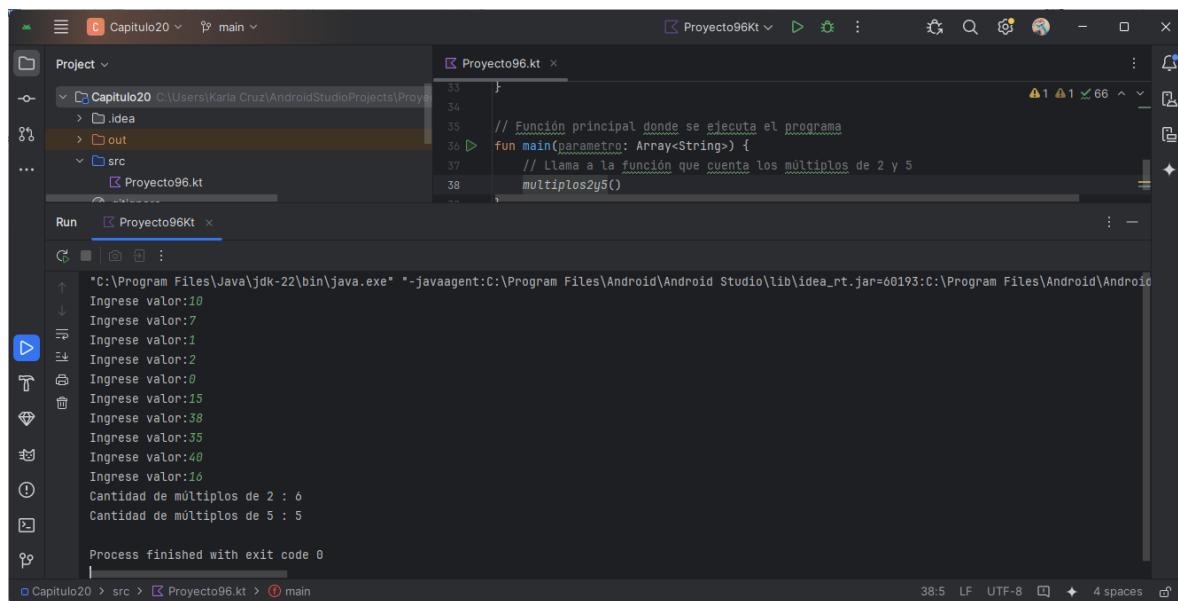
En el main, se llama a esta función de tres formas: primero con los argumentos en orden tradicional, y luego utilizando argumentos nombrados, lo que permite pasarlo en cualquier orden.

## Capítulo 20

## Problema 1

En el proyecto 96 se define una función llamada “multiplos2y5”, que permite ingresar 10 números desde el teclado. Por cada número ingresado, verifica si es múltiplo de 2 o de 5 utilizando una función interna llamada multiplo, la cual retorna true si el número es divisible exactamente por otro.

Durante el proceso, el programa mantiene dos contadores: uno para los múltiplos de 2 y otro para los múltiplos de 5. Al finalizar las 10 entradas, imprime cuántos números ingresados son múltiplos de 2 y cuántos de 5. La función multiplo está definida dentro de multiplos2y5, por lo tanto, solo puede usarse ahí, y no desde otras funciones como main. Esta estructura ayuda a encapsular la lógica que no se necesita fuera de su contexto local.



```
Project Capítulo20 main Proyecto96Kt
Capítulo20 C:\Users\Karla Cruz\AndroidStudioProjects\Proyecto96Kt
src Proyecto96.kt
Run Proyecto96Kt
*C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=60193:C:\Program Files\Android\Android Studio
Ingrese valor:10
Ingrese valor:7
Ingrese valor:1
Ingrese valor:2
Ingrese valor:0
Ingrese valor:15
Ingrese valor:38
Ingrese valor:35
Ingrese valor:40
Ingrese valor:16
Cantidad de múltiplos de 2 : 6
Cantidad de múltiplos de 5 : 5
Process finished with exit code 0
38:5 LF UTF-8 4 spaces
```

## **Capítulo 21**

### Problema 1

Dentro del proyecto 98 se almacenan los sueldos de 5 operarios en un arreglo IntArray. Primero, se declara un arreglo con tamaño 5 usando IntArray(5). Luego, mediante un bucle for, se solicita al usuario ingresar los sueldos, que se almacenan en el arreglo en las posiciones correspondientes. Finalmente, otro bucle for imprime cada sueldo almacenado. El uso de arreglos permite manejar múltiples valores bajo un solo nombre y acceder a ellos mediante índices.

```
// Función principal donde se ejecuta el programa
fun main(parametro: Array<String>) {
    // Declaración de un arreglo de enteros llamado 'sueldos'
    val sueldos: IntArray
        // Inicialización del arreglo con tamaño 5
        sueldos = IntArray( size=5 )
```

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=60237:C:\Program Files\Android\Android Studio\bin" Capítulo21
Ingrese sueldo:2500
Ingrese sueldo:3000
Ingrese sueldo:100
Ingrese sueldo:1500
2500
3000
100
1000
1500

Process finished with exit code 0
```

## Problema 2

En el proyecto 99 se almacenan las alturas de 5 personas en un arreglo `FloatArray`. Luego se calcula el promedio de las alturas sumando todos los valores y dividiendo entre el número de personas. Posteriormente, se cuentan cuántas personas son más altas y cuántas son más bajas que el promedio. Los contadores altos y bajos se actualizan en un segundo bucle al comparar cada altura con el promedio. Finalmente, se imprime el promedio y la cantidad de personas más altas y más bajas que dicho promedio.

```
// Función principal donde se ejecuta el programa
fun main(parametro: Array<String>) {
    // Declaración e inicialización de un arreglo de tipo Float con 5 elementos
    val alturas = FloatArray( size=5 )
    // Variable para almacenar
    var suma = 0f
```

```
Type: In word 'usuario'
Replace with 'usurious' Alt+Mayús+Intro More actions... Alt+Intro
```

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=60345:C:\Program Files\Android\Android Studio\bin" Capítulo21
Ingrese la altura:150
Ingrese la altura:170
Ingrese la altura:155
Ingrese la altura:180
Ingrese la altura:190
Altura promedio: 169.0
Cantidad de personas más altas que el promedio: 3
Cantidad de personas más bajas que el promedio: 2

Process finished with exit code 0
```

## Problema 3

En el proyecto 100 se carga un arreglo de 10 elementos enteros desde el teclado y verifica si están ordenados de menor a mayor. Primero, el usuario ingresa los valores uno por uno en el arreglo. Luego, el programa recorre el arreglo y compara cada elemento con el siguiente. Si encuentra que un elemento es mayor que el siguiente, cambia la variable ordenada a false. Finalmente, imprime si los elementos están ordenados o no según el valor de la variable ordenado.

```
// Función principal donde se ejecuta el programa
fun main(parametro: Array<String>) {
    // Declaración e inicialización de un arreglo de enteros con 10 elementos
    val arreglo = IntArray( size: 10 )
    // Bucle para solicitar al usuario que ingrese los 10 elementos del arreglo
    for(i in 0 .. < arreglo.size-1 ) {
        "C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=60390:C:\Program Files\Android\Android Studio\bin" Capítulo21
        Ingrese elemento:23
        Ingrese elemento:12
        Ingrese elemento:40
        Ingrese elemento:5
        Ingrese elemento:78
        Ingrese elemento:8
        Ingrese elemento:65
        Ingrese elemento:44
        Ingrese elemento:76
        Los elementos no están ordenados de menor a mayor
        Process finished with exit code 0
```

## Problema 4

En el proyecto 101 se solicita al usuario que ingrese 10 valores enteros, que luego se almacenan en un arreglo. Para recorrer el arreglo, se utiliza la propiedad índices, que es más conveniente y legible que utilizar el rango explícito 0..arreglo.size-1. Después de ingresar los valores, el programa imprime el primer elemento del arreglo, accediendo a él mediante arreglo[0], y también imprime el último elemento utilizando la propiedad lastIndex, que devuelve el índice del último valor válido del arreglo. De esta forma, el código es más limpio y eficiente al acceder a los elementos del arreglo sin necesidad de calcular el último índice manualmente.

```
// Función principal donde se ejecuta el programa
fun main(parametro: Array<String>) {
    // Declaración e inicialización de un arreglo de enteros con 10 elementos
    val arreglo = IntArray( size: 10 )
    // Bucle para solicitar al usuario que ingrese los 10 elementos del arreglo
    for (i in 0..9) {
        Ingrese elemento:$i
        arreglo[i] = i + 1
    }
    // Imprimir los 10 elementos del arreglo
    for (i in 0..9) {
        println(arreglo[i])
    }
}
```

Ingrese elemento:1  
Ingrese elemento:2  
Ingrese elemento:3  
Ingrese elemento:4  
Ingrese elemento:5  
Ingrese elemento:6  
Ingrese elemento:7  
Ingrese elemento:8  
Ingrese elemento:9  
Ingrese elemento:10  
Primera componente del arreglo 1  
Última componente del arreglo 10

Process finished with exit code 0

## Problema 5

En el proyecto 102 se solicita al usuario que ingrese 10 números enteros, los cuales se almacenan en un arreglo. Para ingresar los elementos, se utiliza un bucle for que recorre el arreglo con la propiedad índices, lo que permite acceder a cada posición del arreglo de manera simple y eficiente. Después de cargar los valores, otro bucle for recorre el arreglo y se encarga de imprimir cada uno de los elementos. En cada iteración, la variable elemento contiene el valor del arreglo en la posición actual. Además, si se quisiera imprimir el índice junto con el valor de cada elemento, se podría utilizar el método withIndex() de Kotlin para obtener tanto el índice como el valor en cada iteración.

```
// Función principal donde se ejecuta el programa
fun main(parametro: Array<String>) {
    // Declaración e inicialización de un arreglo de enteros con 10 elementos
    val arreglo = IntArray( size: 10 )
    // Bucle para solicitar al usuario que ingrese los 10 elementos del arreglo
    for (i in 0..9) {
        Ingrese elemento:$i
        arreglo[i] = i + 48
    }
    // Imprimir los 10 elementos del arreglo
    for (i in 0..9) {
        println(arreglo[i])
    }
}
```

Ingrese elemento:48  
Ingrese elemento:49  
Ingrese elemento:87  
Ingrese elemento:31  
Ingrese elemento:5  
Ingrese elemento:3  
Ingrese elemento:2  
Ingrese elemento:57  
Ingrese elemento:43  
48  
90  
87  
31  
5  
3  
2  
57  
89

```
// Función principal donde se ejecuta el programa
fun main(parametro: Array<String>) {
    println("Ingresar elemento:0")
    Ingrese elemento:5
    Ingrese elemento:3
    Ingrese elemento:2
    Ingrese elemento:7
    Ingrese elemento:43
    48
    90
    87
    31
    5
    3
    2
    57
    89
    43
}
```

Process finished with exit code 0

## Capítulo 22

### Problema 1

En cuanto al proyecto 105 este está diseñado para cargar y mostrar un arreglo de enteros de 5 elementos. La función principal (main) crea un arreglo llamado arre con 5 elementos. Luego, se llama a la función cargar que recibe como parámetro el arreglo y, utilizando un bucle for, solicita al usuario que ingrese un valor para cada posición del arreglo. El valor ingresado se convierte a entero y se almacena en el arreglo. Posteriormente, se llama a la función imprimir, la cual recorre el arreglo e imprime cada uno de los valores almacenados. Aunque el parámetro de la función cargar se llama arreglo y la variable en main se llama arre, esto no genera ningún problema ya que ambos son del tipo IntArray. Las funciones están estructuradas para separar la lógica de cargar y mostrar los elementos, lo que permite modularizar el código y hacer que sea más claro y reutilizable.

The screenshot shows the Android Studio interface with the project 'Capítulo22' selected. The left sidebar shows the project structure with files 'Proyecto105.kt' and 'Proyecto106.kt' in the 'src' directory. The right side has two tabs: 'Proyecto105.kt' and 'Proyecto106.kt'. The 'Proyecto105.kt' tab contains the following code:

```
20 // Función principal donde se ejecuta el programa
21 fun main(parametro: Array<String>) {
22     // Declaración e inicialización de un arreglo de enteros con 5 elementos
23     val arre = IntArray( size: 5 )
24     // Llamada a la función 'cargar' para llenar el arreglo con valores ingresados por el usuario
25     cargar(arre)
26     // Imprimir los 5 sueldos almacenados en el arreglo
27     imprimirSueldos(arre)
}
```

The 'Run' tab shows the program's output:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=60569:C:\Program Files\Android\Android Studio\bin" -Dfile.encoding=UTF-8 Proyecto105Kt
Ingrese elemento:23
Ingrese elemento:12
Ingrese elemento:19
Ingrese elemento:36
Ingrese elemento:39
23
12
19
36
39

Process finished with exit code 0
```

## Problema 2

Por su parte, el proyecto 106 este está diseñado para almacenar y mostrar los sueldos de operarios. Primero, la función cargar solicita al usuario la cantidad de sueldos que desea ingresar. Luego, crea un arreglo de enteros del tamaño especificado por el usuario. Posteriormente, mediante un bucle for, se solicita al usuario que ingrese cada sueldo, el cual se almacena en el arreglo. Una vez cargados los sueldos, la función cargar retorna el arreglo sueldos.

La función imprimirSueldos recibe el arreglo de sueldos como parámetro y recorre este arreglo para imprimir cada sueldo almacenado. Esta función es llamada en la función main, luego de que el arreglo de sueldos es cargado por la función cargar. En el programa, se observa cómo la función cargar retorna el arreglo creado, y en la función main se le asigna a una variable sueldos, que es del tipo IntArray, gracias a la inferencia de tipos en Kotlin.

```
// Función principal donde se ejecuta el programa
fun main(parametro: Array<String>) {
    // Llamada a la función 'cargar' para obtener el arreglo con los sueldos ingresados por teclado
    val sueldos = cargar()
}

Process finished with exit code 0
```

## Capítulo 23

### Problema 1

En el proyecto 109 se presenta una clase llamada “Persona”, que sirve como modelo para representar a una persona con dos atributos: nombre, de tipo String, y edad, de tipo Int. Estos atributos se inicializan con valores por defecto: una cadena vacía y cero, respectivamente. La clase incluye tres métodos que permiten operar sobre sus datos: inicializar, imprimir y esMayorEdad.

El método inicializar recibe dos parámetros (nombre y edad) y los asigna a los atributos de la clase utilizando la palabra clave this para diferenciarlos de los parámetros con el mismo nombre. El método imprimir muestra en pantalla el nombre y la edad de la persona, y esMayorEdad evalúa si la persona tiene 18 años o más, mostrando un mensaje correspondiente según el caso.

En la función principal del programa (main), se crean dos objetos de la clase Persona: uno con el nombre "Juan" y edad 12, y otro con el nombre "Ana" y edad 50. Luego se llaman los métodos definidos en la clase para cada objeto, permitiendo visualizar sus datos y comprobar si son mayores de edad.

```
Capítulo23 Capítulo23
Projecto109.kt
29
30
31 // Función principal donde se ejecuta el programa
32 fun main(parametro: Array<String>) {
33
34     // Declaración de una variable de tipo Persona
35     val personal: Persona
36
37     // Creación de un objeto de la clase Persona
38     personal1 = Persona()
39
40     // Se inicializan los datos de personal con nombre "Juan" y edad 12
41 }
```

```
*C:\Program Files\Java\jdk-22\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=60846:C:\Program Files\Android\Android Studio\lib\proguard.jar=60846:lib\proguard.jar" Capítulo23
Nombre: Juan y tiene una edad de 12
No es mayor de edad Juan
Nombre: Ana y tiene una edad de 50
Es mayor de edad Ana
Process finished with exit code 0
```

## Problema 2

En el proyecto 110 se define una clase llamada Triangulo, que permite trabajar con los tres lados de un triángulo ingresados por el usuario. A través del método inicializar, el programa solicita al usuario los valores de cada lado. Luego, con el método ladoMayor, identifica e imprime cuál de los tres lados es el más largo utilizando la estructura when. Finalmente, el método esEquilatero evalúa si los tres lados son iguales y determina si el triángulo es equilátero o no. Todo esto se ejecuta en la función principal main, donde se crea un objeto de la clase Triangulo y se llaman sus métodos para mostrar los resultados.

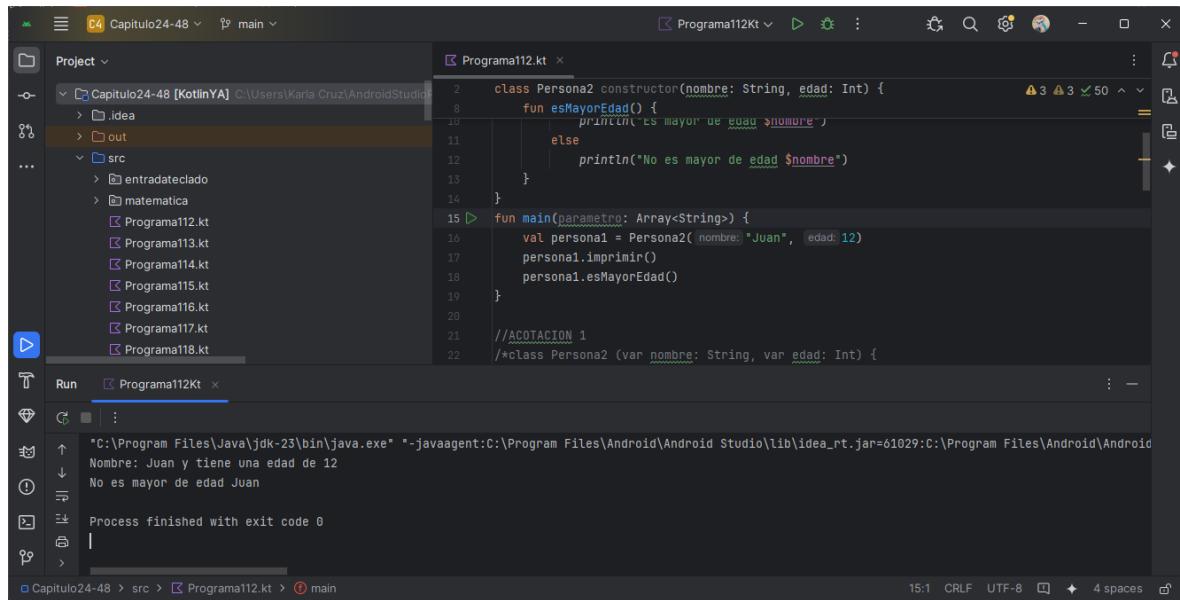
```
Capítulo23 Capítulo23
Projecto110.kt
2
3 class Triangulo {
4     fun esEquilatero() {
5
6     }
7 }
8
9 // Función principal donde se ejecuta el programa
10 fun main(parametro: Array<String>) {
11     // Se crea una instancia de la clase Triangulo
12     val triangulo1 = Triangulo()
13
14     // Se inicializan los valores de los lados del triángulo
15     triangulo1.initialize()
16 }
```

```
*C:\Program Files\Java\jdk-22\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=60902:C:\Program Files\Android\Android Studio\lib\proguard.jar=60902:lib\proguard.jar" Capítulo23
Ingrese lado 1:23
Ingrese lado 2:67
Ingrese lado 3:50
Lado mayor:67
No es un triángulo equilátero
Process finished with exit code 0
```

## Capítulo 24

## Problema 1

En el proyecto 112 la versión original de la clase “Persona2” utiliza un constructor explícito con la palabra clave constructor, y dentro del cuerpo de la clase se asignan manualmente los parámetros recibidos a las propiedades nombre y edad. Aunque esta forma es válida, resulta más extensa de lo necesario. En la primera acotación, se mejora la declaración utilizando una sintaxis más concisa propia de Kotlin, en la cual las propiedades se definen directamente en el encabezado del constructor sin necesidad de asignarlas dentro del cuerpo, eliminando así la redundancia y simplificando el código. La segunda acotación mantiene esa forma concisa, pero añade un bloque init, que permite ejecutar código justo después del constructor. En este caso, se valida que la edad no sea negativa y, si lo es, se corrige automáticamente a cero, lo cual hace el programa más robusto al prevenir datos inválidos desde el momento en que se crea el objeto.

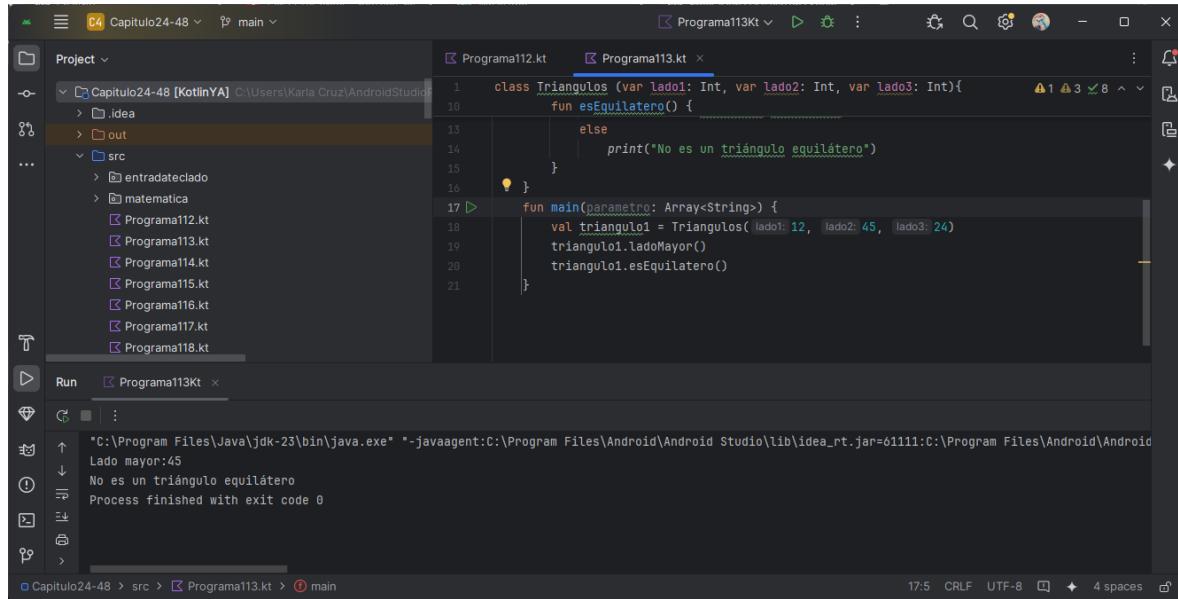


```
2  class Persona2 constructor(nombre: String, edad: Int) {
4      fun esMayorEdad() {
5          if (edad > 0) {
6              println("Es mayor de edad $nombre")
7          } else {
8              println("No es mayor de edad $nombre")
9          }
10     }
11
12     fun main(parametros: Array<String>) {
13         val persona1 = Persona2("Juan", 12)
14         persona1.imprimir()
15         persona1.esMayorEdad()
16     }
17
18     //ACOTACION 1
19     /*class Persona2 (var nombre: String, var edad: Int) {
20         fun imprimir() {
21             println("Nombre: $nombre y tiene una edad de $edad")
22         }
23
24         fun esMayorEdad() {
25             if (edad > 0) {
26                 println("Es mayor de edad $nombre")
27             } else {
28                 println("No es mayor de edad $nombre")
29             }
30         }
31     }*/
32 }
```

## Problema 2

En el proyecto 113 se define una clase llamada Triangulo (y en una segunda versión, Triangulos) que representa un triángulo con tres lados. Las propiedades lado1, lado2 y lado3 se declaran directamente en el constructor principal, lo que permite inicializarlas fácilmente al crear un objeto de la clase. Dentro de la clase, el método ladoMayor determina cuál de los tres lados es el mayor utilizando una estructura when, e imprime el resultado. El segundo método, esEquilatero, verifica si los tres lados son iguales y, en ese caso, informa que el triángulo es equilátero; de lo contrario, indica que no lo es. En la función main, se instancia un objeto triangulo1 con valores específicos para los tres lados, y luego se llaman ambos métodos para mostrar el lado mayor y comprobar si el triángulo es equilátero. La segunda versión del código es prácticamente idéntica, con la única diferencia del nombre de la clase

que cambia a Triangulos, manteniendo exactamente la misma lógica y estructura, por lo que no representa una modificación funcional sino únicamente nominal.



```
1  class Triangulos (var lado1: Int, var lado2: Int, var lado3: Int){  
2      fun esEquilatero(): Boolean {  
3          if (lado1 == lado2 & lado2 == lado3) return true  
4          else print("No es un triángulo equilátero")  
5      }  
6      fun main(parametros: Array<String>) {  
7          val triangulo1 = Triangulos(12, 12, 12)  
8          triangulo1.ladoMayor()  
9          triangulo1.esEquilatero()  
10     }  
11 }
```

Output from Programa113.kt:

```
Lado mayor:45  
No es un triángulo equilátero  
Process finished with exit code 0
```

### Problema 3

En el proyecto 114 se define una clase llamada Triangulo (y su variante Triangulo2) que representa un triángulo con tres lados. El constructor principal recibe tres parámetros (lado1, lado2, lado3) y se utiliza para inicializar las propiedades de la clase. Además, se incluye un segundo constructor sin parámetros que pide al usuario ingresar los valores de los tres lados del triángulo a través de la consola. Este constructor usa `readLine()` para leer los valores ingresados y luego los convierte a enteros. Los métodos `ladoMayor` y `esEquilatero` funcionan de la misma manera que en los ejemplos anteriores, determinando cuál es el lado más largo y verificando si el triángulo es equilátero.

En el `main`, se crean dos instancias de la clase `Triangulo`. La primera instancia usa el constructor sin parámetros, permitiendo que el usuario ingrese los valores de los lados del triángulo desde la consola. La segunda instancia usa el constructor con parámetros, donde se pasan directamente los valores de los tres lados (6, 6, 6). Posteriormente, se llaman los métodos para mostrar el lado mayor y verificar si el triángulo es equilátero en ambas instancias.

La variante `Triangulo2` tiene exactamente la misma funcionalidad, pero el nombre de la clase ha cambiado, manteniendo la misma lógica y estructura, permitiendo la entrada de valores desde el teclado o la asignación directa de valores a los lados del triángulo.

The screenshot shows the Android Studio interface with the project 'Capítulo24-48' open. The code editor displays 'Programa114.kt' containing the following Kotlin code:

```
1  class Triangulo2 (var lado1: Int, var lado2: Int, var lado3: Int){  
2      fun esEquilatero(): Boolean {  
3          if (lado1 == lado2 && lado2 == lado3) return true  
4          else return false  
5      }  
6      fun ladoMayor(): Int {  
7          if (lado1 > lado2 && lado1 > lado3) return lado1  
8          else if (lado2 > lado1 && lado2 > lado3) return lado2  
9          else return lado3  
10     }  
11     fun esIsosceles(): Boolean {  
12         if (lado1 == lado2 || lado2 == lado3) return true  
13         else return false  
14     }  
15     fun esEscaleno(): Boolean {  
16         if (lado1 != lado2 && lado2 != lado3) return true  
17         else return false  
18     }  
19     fun imprimir(): Unit {  
20         println("Lados: $lado1, $lado2, $lado3")  
21         println("Lado mayor: ${ladoMayor()}")  
22         if (esEquilatero())  
23             println("Es un triángulo equilátero")  
24         else if (esIsosceles())  
25             println("Es un triángulo isósceles")  
26         else if (esEscaleno())  
27             println("Es un triángulo escaleno")  
28     }  
29 }  
30  
31 fun main(parametro: Array<String>) {  
32     val triangulo1 = Triangulo2()  
33     triangulo1.imprimir()  
34 }
```

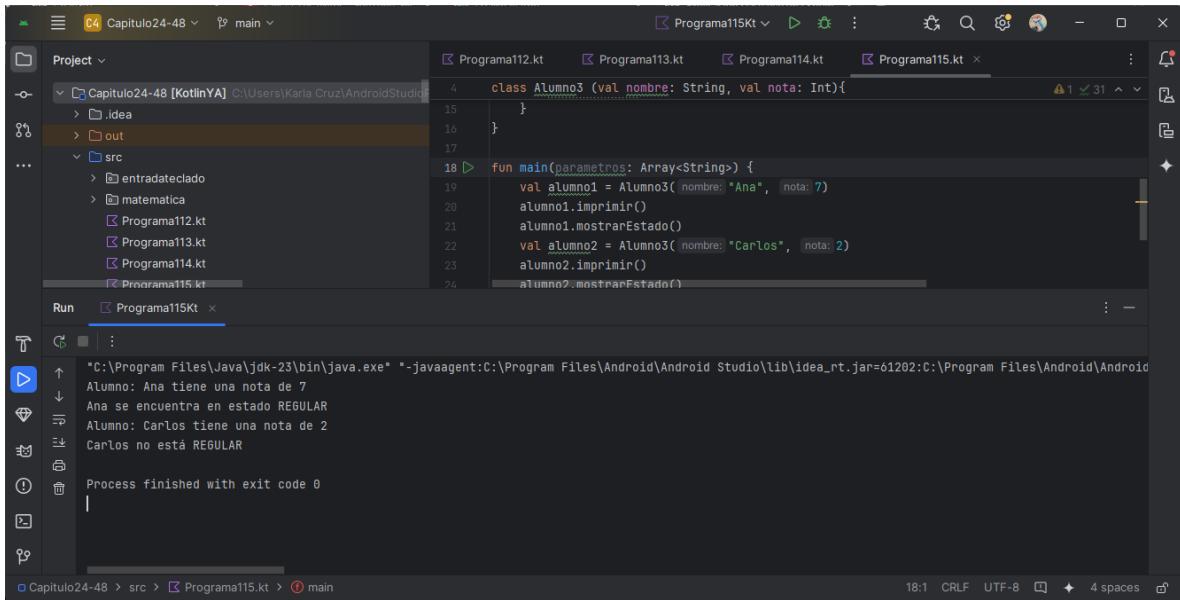
The run output window shows the following terminal output:

```
*C:\Program Files\Java\jdk-23\bin\java.exe* -javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=61157:C:\Program Files\Android\Android Studio\bin  
↑ Ingrese primer lado:23  
↓ Ingrese segundo lado:13  
→ Ingrese tercer lado:45  
Lado mayor:45  
No es un triángulo equilátero  
Lado mayor:6  
Es un triángulo equilátero  
Process finished with exit code 0
```

## Problema 4

En cuanto al proyecto 115 se define una clase llamada Alumno3 con dos propiedades: nombre y nota. El constructor de la clase recibe estos dos valores, que son inicializados al crear un objeto de la clase. La clase incluye dos métodos: imprimir y mostrarEstado. El método imprimir muestra por consola el nombre del alumno y su nota. El método mostrarEstado evalúa la nota del alumno: si la nota es mayor o igual a 4, imprime un mensaje indicando que el alumno se encuentra en estado "REGULAR". Si la nota es menor a 4, imprime que el alumno "no está REGULAR".

En la función main, se crean dos instancias de la clase Alumno3: la primera, alumno1, con el nombre "Ana" y una nota de 7; y la segunda, alumno2, con el nombre "Carlos" y una nota de 2. Para cada uno de estos objetos, se llaman los métodos imprimir y mostrarEstado para mostrar los datos del alumno y su estado de acuerdo a la nota.



```
class Alumno3 (val nombre: String, val nota: Int){  
    }  
  
fun main(parametros: Array<String>){  
    val alumno1 = Alumno3("Ana", 7)  
    alumno1.imprimir()  
    alumno1.mostrarEstado()  
    val alumno2 = Alumno3("Carlos", 2)  
    alumno2.imprimir()  
    alumno2.mostrarEstado()  
}
```

## Problema 5

En cuanto al proyecto 116 se implementa una clase llamada Punto, que representa un punto en el plano cartesiano, definido por sus coordenadas x e y. El constructor de la clase recibe estos valores como parámetros para inicializar las propiedades del objeto.

La clase incluye un método llamado retornarCuadrante, que evalúa las coordenadas x e y del punto para determinar en qué cuadrante se encuentra. Utiliza una estructura when para realizar la comprobación:

- ✓ Si  $x > 0$  y  $y > 0$ , el punto está en el primer cuadrante.
- ✓ Si  $x < 0$  y  $y > 0$ , está en el segundo cuadrante.
- ✓ Si  $x < 0$  y  $y < 0$ , está en el tercer cuadrante.
- ✓ Si  $x > 0$  y  $y < 0$ , está en el cuarto cuadrante.
- ✓ Si cualquiera de las coordenadas es cero, el punto se encuentra en un eje.

En la función main, se crean cinco objetos de la clase Punto, cada uno en un cuadrante diferente o en un eje:

1. punto1 está en el primer cuadrante (12, 3).
2. punto2 está en el segundo cuadrante (-4, 3).
3. punto3 está en el tercer cuadrante (-2, -2).
4. punto4 está en el cuarto cuadrante (12, -5).
5. punto5 está en un eje (0, -5).

Para cada uno de estos puntos, se imprime la coordenada y el cuadrante correspondiente en el que se encuentra utilizando el método retornarCuadrante.

```
fun main(parametro: Array<String>) {
    val punto1 = Punto(12, 3)
    println("La coordenada ${punto1.x}, ${punto1.y} se encuentra en ${punto1.retornarCuadrante}")
    val punto2 = Punto(-4, 3)
    println("La coordenada ${punto2.x}, ${punto2.y} se encuentra en ${punto2.retornarCuadrante}")
    val punto3 = Punto(-2, -2)
    println("La coordenada ${punto3.x}, ${punto3.y} se encuentra en ${punto3.retornarCuadrante}")
    val punto4 = Punto(12, -5)
    println("La coordenada ${punto4.x}, ${punto4.y} se encuentra en ${punto4.retornarCuadrante}")
    val punto5 = Punto(0, -5)
}
```

## Capítulo 25

### Problema 1

En el proyecto 117 se define una clase llamada “Operaciones”, la cual tiene dos propiedades de tipo Int, valor1 y valor2, que representan los números sobre los que se realizarán operaciones matemáticas. Dentro de la clase, se encuentra un método llamado cargar() que solicita al usuario ingresar estos dos valores, luego invoca dos métodos adicionales, sumar() y restar(), que realizan las operaciones de suma y resta respectivamente, mostrando los resultados por la consola.

El método sumar() calcula la suma de los dos valores y lo imprime, mientras que el método restar() realiza la resta de los dos valores y también imprime el resultado. Ambos métodos son llamados dentro del método cargar(), que es el encargado de coordinar la entrada de datos y la ejecución de las operaciones.

La función main() es el punto de inicio del programa. En ella, se crea un objeto de la clase Operaciones, denominado operaciones1. Posteriormente, se llama al método cargar() a través de este objeto, lo que activa el proceso de captura de los valores desde la consola y la ejecución de las operaciones de suma y resta. Finalmente, los resultados de estas operaciones se muestran al usuario en la consola.

The screenshot shows the Android Studio interface with the following details:

- Project View:** Shows the project structure under "Capítulo24-48 [KotlinYA]". The "src" folder contains files: Entradadeclado.kt, matematica.kt, Programa112.kt, Programa113.kt, Programa114.kt, Programa115.kt, Programa116.kt, and Programa117.kt.
- Code Editor:** The file "Programa117.kt" is open, displaying the following Kotlin code:

```
class Operaciones {
    val resta = valor1 - valor2
    println("La resta de $valor1 y $valor2 es $resta")
}

fun main(parametro: Array<String>) {
    val operaciones1 = Operaciones()
    operaciones1.cargar()
}
```
- Run Tab:** The "Programa117Kt" tab is selected. The output window shows the program's execution:

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=61321:C:\Program Files\Android\Android Studio\bin" -Dfile.encoding=UTF-8
Ingrese primer valor:24
Ingrese segundo valor:19
La suma de 24 y 19 es 43
La resta de 24 y 19 es 5

Process finished with exit code 0
```

## Problema 2

Por otra parte, en el proyecto 118 se define una clase llamada “Hijos”, que tiene como propiedad un arreglo de enteros edades, el cual puede almacenar las edades de cinco personas. Dentro de la clase, se incluyen tres métodos importantes: cargar(), mayorEdad() y promedio().

El método cargar() permite que el usuario ingrese las edades de las cinco personas. Para ello, el método recorre el arreglo de edades utilizando un ciclo for, solicitando al usuario que ingrese una edad en cada iteración y guardándola en la posición correspondiente del arreglo. Una vez que todas las edades han sido ingresadas, el método llama a los otros dos métodos, mayorEdad() y promedio(), para calcular y mostrar los resultados.

El método mayorEdad() busca la edad más alta en el arreglo. Inicia asignando el primer valor del arreglo como la mayor edad y luego recorre todas las posiciones del arreglo, actualizando la variable mayor si encuentra una edad mayor. Finalmente, imprime la edad más alta.

Por otro lado, el método `promedio()` calcula el promedio de las edades. Primero suma todas las edades en el arreglo y luego divide la suma entre el número total de elementos en el arreglo (en este caso, 5). Finalmente, imprime el promedio de las edades.

En la función main(), se crea un objeto de la clase Hijos, denominado hijos1, y se llama al método cargar() de este objeto, lo que inicia el proceso de ingreso de edades y cálculo de la mayor edad y el promedio.

The screenshot shows the Android Studio interface with the following details:

- Project View:** Shows the project structure with files like Capítulo24-48, .idea, out, src, entradateclado, matematica, Programa112.kt, Programa113.kt, and Programa114.kt.
- Code Editor:** Displays a Kotlin file named Programa118.kt containing the following code:

```
class Hijos {
    fun promedio() {
    }
}

fun main(parametros: Array<String>) {
    val hijos1 = Hijos()
    hijos1.cargar()
}
```
- Run Tab:** Shows the output of the program execution:

```
*C:\Program Files\Java\jdk-23\bin\java.exe* -javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=61361:C:\Program Files\Android\Android Studio\bin\bootstraps\lib\rt.jar
Ingrese edad:20
Ingrese edad:15
Ingrese edad:30
Ingrese edad:45
Ingrese edad:57
El hijo con mayor edad es 57
La edad promedio de los hijos es 33
Process finished with exit code 0
```
- Status Bar:** Shows the file path Capítulo24-48 > src > Programa118.kt > main, and the encoding settings 34:1 CRLF UTF-8, and 4 spaces.

## Capítulo 26

### Problema 1

En el proyecto 119 se definen dos clases: Cliente y Banco. La clase Cliente tiene como responsabilidad manejar la información de cada cliente, que incluye su nombre y el monto de dinero que tiene depositado. El constructor de esta clase recibe el nombre del cliente y su monto inicial, y dentro de la clase se definen tres métodos clave. El primero es depositar(monto: Float), que suma una cantidad al saldo del cliente. El segundo es extraer(monto: Float), que resta una cantidad del saldo del cliente. Finalmente, el método imprimir() muestra el nombre del cliente y el monto que tiene depositado.

Por otro lado, la clase Banco gestiona tres instancias de la clase Cliente, que representan a tres clientes diferentes: Juan, Ana y Luis. En esta clase se encuentran definidos dos métodos importantes. El primero, operar(), realiza una serie de operaciones de depósito y extracción sobre los clientes. En esta operación, Juan deposita 100, Ana deposita 150, y Luis deposita 200 pero luego extrae 150. El segundo método, depositosTotales(), calcula el total de dinero depositado en el banco sumando los montos de los tres clientes y luego imprime el total, llamando también al método imprimir() de cada cliente para mostrar su saldo.

En la función principal main(), se crea un objeto banco1 de la clase Banco y se ejecutan los métodos operar() y depositosTotales(). Esto hace que se realicen las operaciones de depósito y extracción, y luego se calcule y muestre el total de dinero en el banco, junto con la información individual de cada cliente.

The screenshot shows the Android Studio interface with the following details:

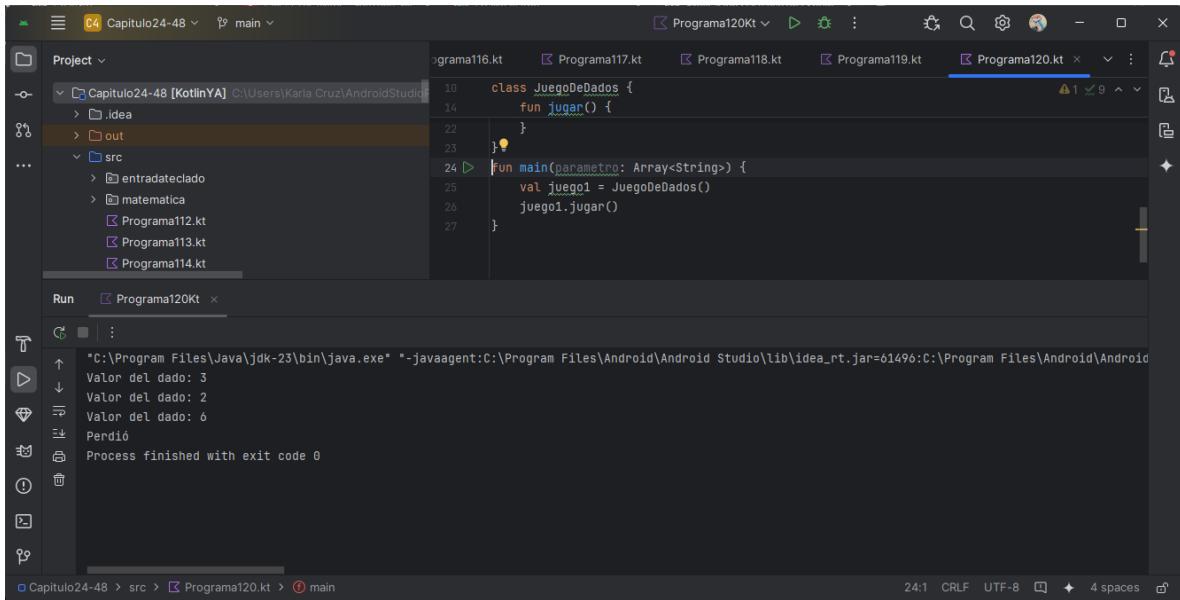
- Project View:** Shows the project structure under "Capítulo24-48 [KotlinYA]".
- Code Editor:** Displays the code for `Programa119.kt`. The code defines a class `Banco` with a main function that initializes a bank account, performs an operation, and prints total deposits.
- Run Tab:** Shows the output of the program execution. The console output includes:
  - "C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea\_rt.jar=61441:C:\Program Files\Android\Android Studio\bin" -Dfile.encoding=UTF-8 main
  - El total de dinero del banco es: 300.0
  - Juan tiene depositado la suma de 100.0
  - Ana tiene depositado la suma de 150.0
  - Luis tiene depositado la suma de 50.0
- Bottom Status Bar:** Shows file paths (Capítulo24-48 > src > Programa119.kt > main), encoding (UTF-8), and code style settings (30:1, CRLF, 4 spaces).

## Problema 2

En cuanto al proyecto 120 el código está compuesto por dos clases principales: Dado y JuegoDeDados. La clase Dado tiene una propiedad llamada valor, que representa el valor actual del dado. Dentro de esta clase, el método tirar() asigna un valor aleatorio entre 1 y 6 al dado, utilizando la función Math.random() para generar un número decimal aleatorio, el cual luego se convierte a un número entero. Después de asignar el nuevo valor, se llama al método imprimir(), que simplemente muestra el valor del dado en la consola.

La clase JuegoDeDados crea tres objetos de la clase Dado (dado1, dado2 y dado3), cada uno con un valor inicial de 1. El método jugar() es el que simula el lanzamiento de los tres dados. En este método, se invoca el método tirar() para cada dado, lo que actualiza su valor con un nuevo número aleatorio entre 1 y 6. Después de tirar los dados, el programa verifica si los tres dados tienen el mismo valor. Si es así, imprime "Ganó", lo que indica que los tres dados muestran el mismo número, de lo contrario, imprime "Perdió", ya que los valores de los dados son diferentes.

En la función main(), se crea un objeto de la clase JuegoDeDados y se llama al método jugar(), lo que simula el lanzamiento de los tres dados y muestra el resultado del juego, ya sea ganando o perdiendo, según si los tres dados tienen el mismo valor o no.

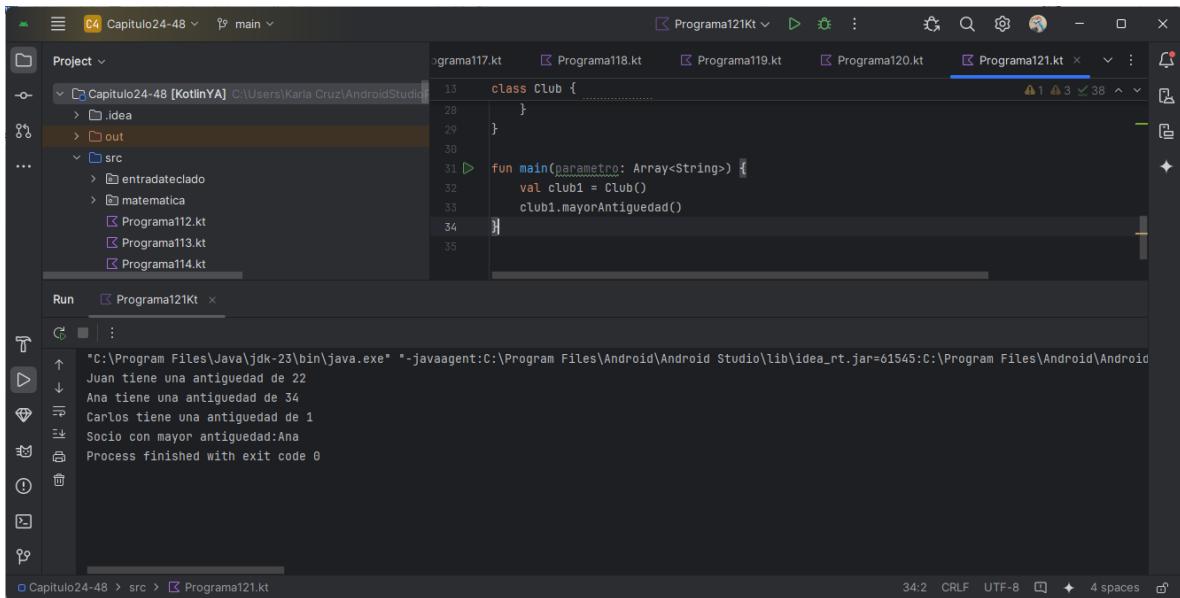


```
class JuegoDeDados {
    fun jugar() {
    }
}

fun main(parametro: Array<String>) {
    val juego1 = JuegoDeDados()
    juego1.jugar()
}
```

### Problema 3

En el proyecto 121 se definen dos clases: Socio y Club. La clase Socio tiene propiedades para almacenar el nombre y la antigüedad en el club de un socio, y un método imprimir() para mostrar esta información. La clase Club contiene tres objetos de tipo Socio y un método mayorAntiguedad() que compara las antigüedades de los socios y muestra el nombre del socio con más años en el club. En la función main(), se crea un objeto de Club y se llama al método mayorAntiguedad() para imprimir el nombre del socio con mayor antigüedad.



```
class Club {
}

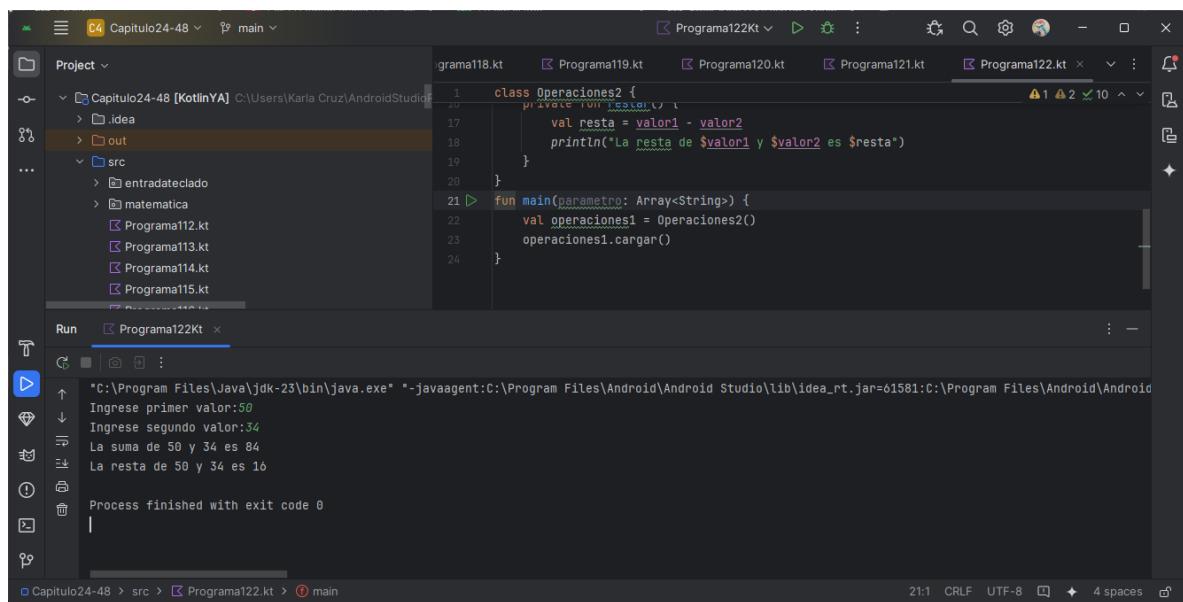
fun main(parametro: Array<String>) {
    val club1 = Club()
    club1.mayorAntiguedad()
}
```

## Capítulo 27

### Problema 1

En el proyecto 122 se define una clase “Operaciones2” que realiza operaciones aritméticas básicas (suma y resta) con dos valores enteros. Los valores son solicitados al usuario mediante la función cargar(), que lee dos números y luego llama a los métodos privados sumar() y restar() para realizar las operaciones. Los métodos sumar() y restar() son privados, lo que significa que solo pueden ser llamados dentro de la misma clase. En el método main(), se crea un objeto de la clase Operaciones2 y se invoca la función cargar() para realizar las operaciones y mostrar los resultados.

La clase usa modificadores de acceso private para ocultar las propiedades valor1 y valor2, así como los métodos de operaciones, lo que significa que no se pueden acceder directamente desde fuera de la clase.



```
class Operaciones2 {
    private fun sumar() {
        val suma = valor1 + valor2
        println("La suma de $valor1 y $valor2 es $suma")
    }
    private fun restar() {
        val resta = valor1 - valor2
        println("La resta de $valor1 y $valor2 es $resta")
    }
}

fun main(parametro: Array<String>) {
    val operaciones1 = Operaciones2()
    operaciones1.cargar()
}
```

## Problema 2

En cuanto al proyecto 123 se define una clase “Dado2” que simula un dado con un valor aleatorio entre 1 y 6. La propiedad valor es privada, lo que significa que no puede ser accedida directamente desde fuera de la clase. Tiene dos métodos públicos: tirar() y imprimir(). El método tirar() genera un número aleatorio entre 1 y 6 y lo asigna a la propiedad valor. El método imprimir() muestra el valor del dado, utilizando el método privado separador() para imprimir una línea de separación antes y después del valor del dado.

La función main() crea un objeto de la clase Dado2, llama al método tirar() para asignar un valor aleatorio al dado y luego imprime el valor usando el método imprimir(). Los métodos tirar() e imprimir() son accesibles desde fuera de la clase, mientras que el método separador() y la propiedad valor están encapsulados y son privados.

```
class Dado2{  
    fun imprimir() {  
        println("Valor del dado: $valor")  
        separador()  
    }  
    private fun separador() = println("*****")  
}  
fun main(parametro: Array<String>) {  
    val dado1 = Dado2()  
    dado1.tirar()  
    dado1.imprimir()  
}
```

\*C:\Program Files\Java\jdk-23\bin\java.exe\* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea\_rt.jar=61585:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8  
\*\*\*\*\*  
Valor del dado: 5  
\*\*\*\*\*  
Process finished with exit code 0

### Problema 3

En el proyecto 124 se define una clase Vector que contiene una propiedad privada vec, un arreglo de 5 enteros. En su bloque init, se invoca un método privado llamado cargar(), que asigna valores aleatorios entre 0 y 10 a cada elemento del arreglo. La clase tiene tres métodos públicos: imprimir(), que muestra todos los elementos del arreglo; mostrarMayor(), que encuentra y muestra el mayor valor en el arreglo; y mostrarMenor(), que encuentra y muestra el valor más pequeño. En la función main(), se crea una instancia de la clase Vector y se llaman estos métodos para imprimir el arreglo y mostrar tanto el mayor como el menor valor. La propiedad vec y el método cargar() son privados, por lo que no pueden ser accedidos directamente desde fuera de la clase, pero los métodos públicos permiten interactuar con ellos de manera controlada.

The screenshot shows the Android Studio interface with the project 'Capítulo24-48' open. The code editor displays 'Programa124.kt' containing the following Kotlin code:

```
class Vector {
    fun mostrarMenor() {
        println("El elemento menor del arreglo es $menor")
    }
}

fun main(parametro: Array<String>) {
    val vector1 = Vector()
    vector1.imprimir()
    vector1.mostrarMayor()
    vector1.mostrarMenor()
}
```

The run tab shows the output of the program:

```
*C:\Program Files\Java\jdk-23\bin\java.exe* -javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=61593:C:\Program Files\Android\Android Studio\lib\proguard.jar C:\Program Files\Android\Android Studio\projects\Capítulo24-48\app\src\main\kotlin\Programa124.kt
Listado completo del arreglo
2 - 5 - 6 - 3 - 1 -
El elemento mayor del arreglo es 6
El elemento menor del arreglo es 1
Process finished with exit code 0
```

## Capítulo 28

### Problema 1

En el proyecto 125 se define una clase “Persona” que tiene dos propiedades: nombre de tipo String y edad de tipo Int. Ambas propiedades utilizan métodos set personalizados para modificar sus valores. El setter de nombre convierte el valor ingresado a mayúsculas antes de asignarlo, mientras que el getter devuelve el nombre entre paréntesis. La propiedad edad tiene un setter que valida si el valor es negativo. Si lo es, asigna 0, y si no, asigna el valor proporcionado. En el bloque main, se crea un objeto de la clase Persona, se asignan valores a sus propiedades, y luego se imprimen los resultados. Cuando se asigna un valor negativo a edad, el setter asegura que el valor sea 0. También se incluye una versión mejorada de la clase utilizando uppercase(Locale.getDefault()) para asegurar la conversión a mayúsculas de manera compatible con el idioma del sistema.

The screenshot shows the Android Studio interface with the project 'Capítulo24-48' open. The 'Programa125.kt' file is selected in the Project tool window. The code editor displays the following Kotlin code:

```
class Persona3 {
    var edad: Int = 0
    set(value) {
        if (value < 0) edad = 0 else edad = value
    }
}

fun main(parametro: Array<String>) {
    val personal1 = Persona3()
    personal1.nombre = "JUAN"
    personal1.edad = 23
    println(personal1.nombre) // Se imprime: (JUAN)
    println(personal1.edad) // Se imprime: 23
    personal1.edad = -50
}
```

The Run tool window shows the output of the program:

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=61626:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8 Programa125Kt
(JUAN)
23
0
Process finished with exit code 0
```

The status bar at the bottom indicates the file path 'Capítulo24-48 > src > Programa125.kt > main' and the time '20:30'.

## Problema 2

En el proyecto 126 se define una clase llamada “Empleado” que representa a un empleado con dos propiedades: nombre, que se asigna directamente desde el constructor, y sueldo, que tiene una lógica de validación. Para evitar que se asignen valores negativos al sueldo, se implementa un setter personalizado que verifica si el valor proporcionado es menor que cero; en ese caso, asigna 0.0 como sueldo. De lo contrario, asigna el valor recibido. Esta lógica se aplica dentro del bloque init, que se ejecuta automáticamente al crear un nuevo objeto. Además, la clase incluye un método imprimir que muestra el nombre del empleado junto con su sueldo. En la función main, se crean dos objetos de la clase: uno con un sueldo positivo y otro con un sueldo negativo. Al imprimir sus datos, se muestra que el primer empleado conserva el sueldo ingresado, mientras que al segundo se le asigna un sueldo de 0.0 debido a la validación implementada.

```
Capítulo24-48 main
```

```
Programa122.kt Programa123.kt Programa124.kt Programa125.kt Programa126.kt
```

Project

```
Programa123.kt  
Programa124.kt  
Programa125.kt  
Programa126.kt  
Programa127.kt  
Programa128.kt  
Programa129.kt  
Programa130.kt  
Programa131.kt  
Programa132.kt  
Programa133.kt
```

Run

```
Programa126Kt
```

Syntax highlighting has been temporarily turned off in file Programa126.kt because of an internal error

```
5     class Empleado(var nombre: String, sueldo: Double) {  
6         }  
7     }  
8     }  
9     }  
10    }  
11    }  
12    }  
13    }  
14    }  
15    }  
16    }  
17    }  
18    }  
19    }  
20    }  
21    }  
22    }  
23    }  
24    fun main(parametro: Array<String>) {  
25        val empleado01 = Empleado(nombre: "Juan", sueldo: 12000.5)  
26        empleado01.imprimir()  
27        val empleado02 = Empleado(nombre: "Ana", sueldo: -1200.0)  
28        empleado02.imprimir()  
29    }  
30    }
```

\*C:\Program Files\Java\jdk-23\bin\java.exe\* -javaagent:C:\Program Files\Android\Android Studio\lib\idea\_rt.jar=61636:C:\Program Files\Android\Android Studio\lib\proguard.jar -Dfile.encoding=UTF-8  
Juan tiene un sueldo de 12000.5  
Ana tiene un sueldo de 0.0  
Process finished with exit code 0

24:1 CRLF UTF-8 4 spaces

### Problema 3

En el proyecto 127 se define una clase llamada “Dado4” que representa un dado con una propiedad llamada valor. Esta propiedad solo acepta valores comprendidos entre 1 y 6 gracias a un setter personalizado que verifica si el número asignado está dentro del rango permitido. Si se intenta asignar un valor fuera de ese rango (como 0 o 7), automáticamente se establece el valor en 1. El constructor de la clase recibe un valor inicial, el cual pasa por esta validación a través del bloque init. Además, la clase implementa dos métodos: tirar, que genera un número aleatorio entre 1 y 6 simulando una tirada de dado, y imprimir, que muestra el valor actual del dado. En la función main, se crea un objeto de la clase pasando el valor 7 como inicial, lo cual no está permitido, por lo que se asigna automáticamente un 1. Luego, se imprime este valor inicial, se genera una nueva tirada aleatoria y se vuelve a imprimir el nuevo valor.

The screenshot shows the Android Studio interface with the following details:

- Project View:** Shows files like Programa123.kt, Programa124.kt, Programa125.kt, Programa126.kt, Programa127.kt, Programa128.kt, Programa129.kt, Programa130.kt, Programa131.kt, Programa132.kt, and Programa133.kt.
- Main Editor:** Displays the code for Programa127.kt:

```
5     class Dado4(valor: Int){  
6         fun imprimir() = println("Valor del dado: $valor")  
7     }  
8  
9     fun main(parametro: Array<String>) {  
10        val dado1 = Dado4( valor: 7)  
11        dado1.imprimir()  
12        dado1.tirar()  
13        dado1.imprimir()  
14    }  
15 }
```
- Run Tab:** Shows the output of the run command: "C:\Program Files\Java\jdk-23\bin\java.exe" -javaagent:C:\Program Files\Android\Android Studio\lib\idea\_rt.jar=61640:C:\Program Files\Android\Android Studio\lib\proguard.jar -Dfile.encoding=UTF-8 main". The output shows "Valor del dado: 1" and "Valor del dado: 5".
- Status Bar:** Shows file statistics: 25:1 CRLF, 134 lines, and encoding: UTF-8.

## Capítulo 29

### Problema 1

En el proyecto 128 se muestra cómo funcionan los data class, que son clases especiales para manejar datos de forma sencilla. La clase Articulo tiene tres propiedades: código, descripción y precio. Al crear objetos e imprimirlas, se muestra automáticamente su contenido gracias al método `toString()` que genera el compilador.

Se demuestra que, si dos variables apuntan al mismo objeto, un cambio en una se refleja en la otra. Luego, al usar el método `copy()`, se genera una nueva instancia independiente. Finalmente, se compara el contenido de los objetos usando `==`, que en los data class compara propiedad por propiedad.

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor containing the following Kotlin code:

```
data class Articulo(var codigo: Int, var descripcion: String, var precio: Double)
fun main(parametro: Array<String>) {
    val articulo1 = Articulo(1, "papas", 34.0)
    val articulo2 = Articulo(2, "manzanas", 24.0)
    println(articulo1)
    println(articulo2)
    val puntero = articulo1
    puntero.precio = 100f
    println(articulo1)
    println(articulo2)
}
```

Below the code editor is the run tab, which is set to "Programa128Kt". The output window shows the results of the program's execution:

```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=61647:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8
Articulo(codigo=1, descripcion=papas, precio=34.0)
Articulo(codigo=2, descripcion=manzanas, precio=24.0)
Articulo(codigo=1, descripcion=papas, precio=100.0)
Articulo(codigo=1, descripcion=papas, precio=200.0)
Articulo(codigo=1, descripcion=papas, precio=100.0)
Son distintos Articulo(codigo=1, descripcion=papas, precio=200.0) y Articulo(codigo=1, descripcion=papas, precio=100.0)
Son iguales Articulo(codigo=1, descripcion=papas, precio=200.0) y Articulo(codigo=1, descripcion=papas, precio=200.0)

Process finished with exit code 0
```

## Problema 2

En el proyecto 129 se define un data class llamado “Persona”, que contiene las propiedades nombre y edad. Aunque los data class ya incluyen automáticamente un método `toString()`, en este ejemplo se sobrescribe dicho método para personalizar el formato de salida. En lugar de mostrar el nombre del objeto y sus propiedades con etiquetas, se imprime directamente el nombre y la edad en una sola línea, separados por una coma.

En la función `main`, se crean dos objetos `Persona` con diferentes valores y se imprimen. Gracias a la sobrescritura del método `toString()`, la salida es más simple y clara, mostrando directamente: "Juan, 22" y "Ana, 59".

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor containing the following Kotlin code:

```
data class Persona4(var nombre: String, var edad: Int) {
    override fun toString(): String {
        return "$nombre, $edad"
    }
}
fun main(parametro: Array<String>) {
    var persona1 = Persona4("Juan", 22)
    var persona2 = Persona4("Ana", 59)
    println(persona1)
    println(persona2)
}
```

Below the code editor is the run tab, which is set to "Programa129Kt". The output window shows the results of the program's execution:

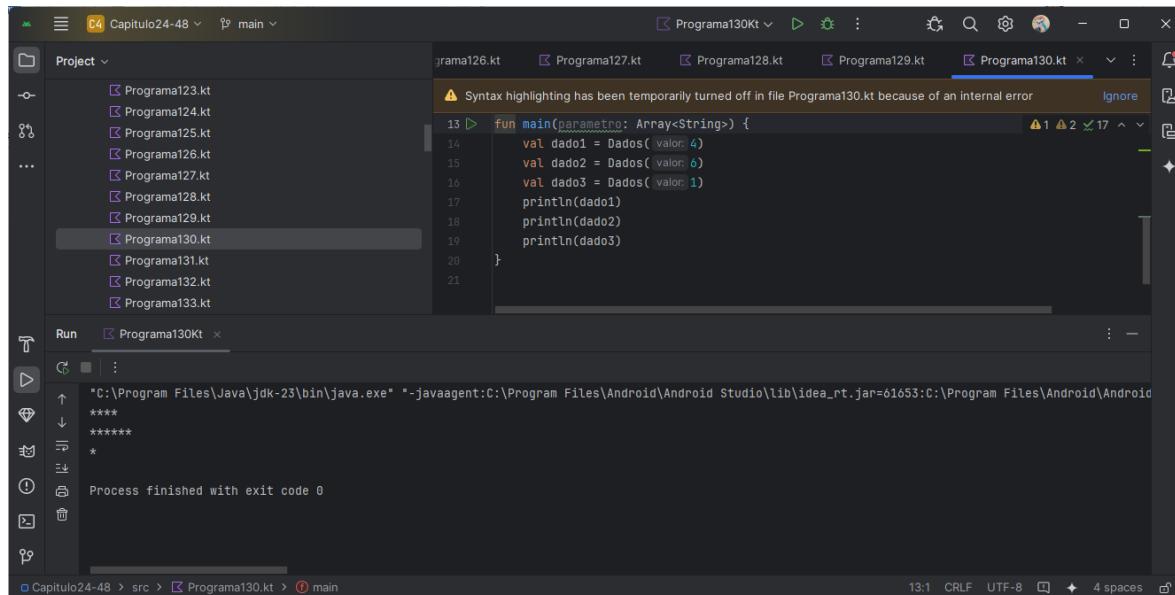
```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=61651:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8
Juan, 22
Ana, 59

Process finished with exit code 0
```

### Problema 3

En el proyecto 130 se define un data class llamado “Dados” con una sola propiedad llamada valor, que representa el valor de un dado. Se sobrescribe el método `toString()` para que, en lugar de mostrar el nombre del objeto y su valor, se impriman tantos asteriscos (\*) como indique dicha propiedad.

En la función `main`, se crean tres objetos `Dados` con valores 4, 6 y 1 respectivamente. Al imprimirlos con `println`, se ejecuta automáticamente el método `toString()`, generando una línea de asteriscos que representa gráficamente el valor de cada dado. Por ejemplo, si el valor es 4, se imprimen cuatro asteriscos (\*\*\*\*), ofreciendo una representación visual simple y clara del valor del dado.



The screenshot shows the Android Studio interface with the project navigation bar at the top. The main area displays the code for `Programa130.kt`. The code defines a `Dados` class with a `valor` property and overrides the `toString()` method to return a string of asterisks based on the value. In the run tab, the output shows three lines of asterisks corresponding to the values 4, 6, and 1 respectively. The bottom status bar indicates the file is saved in C:\Capítulo24-48\src\Programa130.kt with a main branch.

```
fun main(parametro: Array<String>) {
    val dado1 = Dados( valor: 4)
    val dado2 = Dados( valor: 6)
    val dado3 = Dados( valor: 1)
    println(dado1)
    println(dado2)
    println(dado3)
}
```

## Capítulo 30

### Problema 1

En el proyecto 131 se utiliza una enumeración llamada `TipoCarta` para representar los cuatro palos posibles de una baraja: diamante, trébol, corazón y pica. La clase `Carta` contiene dos propiedades: `tipo`, que utiliza el enum `TipoCarta`, y `valor`, que representa el número de la carta.

Dentro de la clase, se define un método llamado `imprimir` que muestra por consola el tipo de carta y su valor. En la función `main`, se crea una carta específica, en este caso de trébol con valor 4, y se llama al método `imprimir` para mostrar esa información.

The screenshot shows the Android Studio interface with the project 'Capítulo24-48' open. The 'Programa131.kt' file is selected in the Project tool window. The code in the editor is:

```
7     class Carta(val tipo: TipoCarta, val valor: Int) {  
8         }  
9     }  
10    }  
11    }  
12    fun main(parametro: Array<String>) {  
13        val carta1 = Carta(TipoCarta.TREBOL, valor: 4)  
14        carta1.imprimir()  
15    }  
16}
```

The Run tool window shows the output of the program:

```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=61777:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8  
Carta: TREBOL y su valor es 4  
Process finished with exit code 0
```

The status bar at the bottom indicates the file is 15.2 CRLF, UTF-8, and has 4 spaces.

## Problema 2

Por su parte, el proyecto 132 muestra cómo usar un enum class para representar operaciones matemáticas básicas como suma, resta, multiplicación y división. En el enum class llamado TipoOperacionn, cada constante está asociada con un símbolo correspondiente (+, -, \*, /) mediante la propiedad tipo. La clase Operacion toma dos números enteros y un objeto de tipo TipoOperacionn para realizar una operación matemática. Dentro del método operar(), se utiliza un bloque when para determinar qué operación realizar según el valor de tipoOperacion. Luego, se imprime el resultado de la operación en un formato legible, mostrando los operandos, el operador y el resultado. En la función main, se crean objetos de la clase Operacion para cada tipo de operación y se llama al método operar() para realizar las operaciones con valores específicos. Esto da como resultado la ejecución de las operaciones y la visualización de los resultados en la consola.

```
class Operacion (val valor1: Int, val valor2: Int, val tipoOperacion: TipoOperacionn) {
    fun main(parametro: Array<String>) {
        val operacion1 = Operacion(valor1 = 10, valor2 = 4, tipoOperacion = TipoOperacionn.SUMA)
        val operacion2 = Operacion(valor1 = 10, valor2 = 4, tipoOperacion = TipoOperacionn.RESTA)
        val operacion3 = Operacion(valor1 = 10, valor2 = 4, tipoOperacion = TipoOperacionn.MULTIPLICACION)
        val operacion4 = Operacion(valor1 = 10, valor2 = 4, tipoOperacion = TipoOperacionn.DIVISION)
        operacion1.operar()
        operacion2.operar()
        operacion3.operar()
        operacion4.operar()
    }
}

Process finished with exit code 0
```

### Problema 3

En cuanto al proyecto 133 este define un enum class llamado Paises, donde cada constante representa un país sudamericano, y cada país tiene una propiedad asociada llamada habitantes, que almacena la cantidad de habitantes de ese país. Cada constante en el enum class se inicializa con la cantidad correspondiente de habitantes. En la función main, se crean variables de tipo Paises para cada uno de los países definidos en el enum, y se imprime tanto el nombre del país como su cantidad de habitantes mediante la propiedad habitantes. El programa imprime esta información para cada país de la lista, mostrando el nombre del país y su número de habitantes en la consola.

```
enum class Paises {
    BRASIL(202450649),
    COLOMBIA(50364000),
    PERU(31151643),
    VENEZUELA(31028337),
    CHILE(18261884),
    ECUADOR(16298217),
    BOLIVIA(10888000),
    PARAGUAY(6460000),
    URUGUAY(3372000)
}

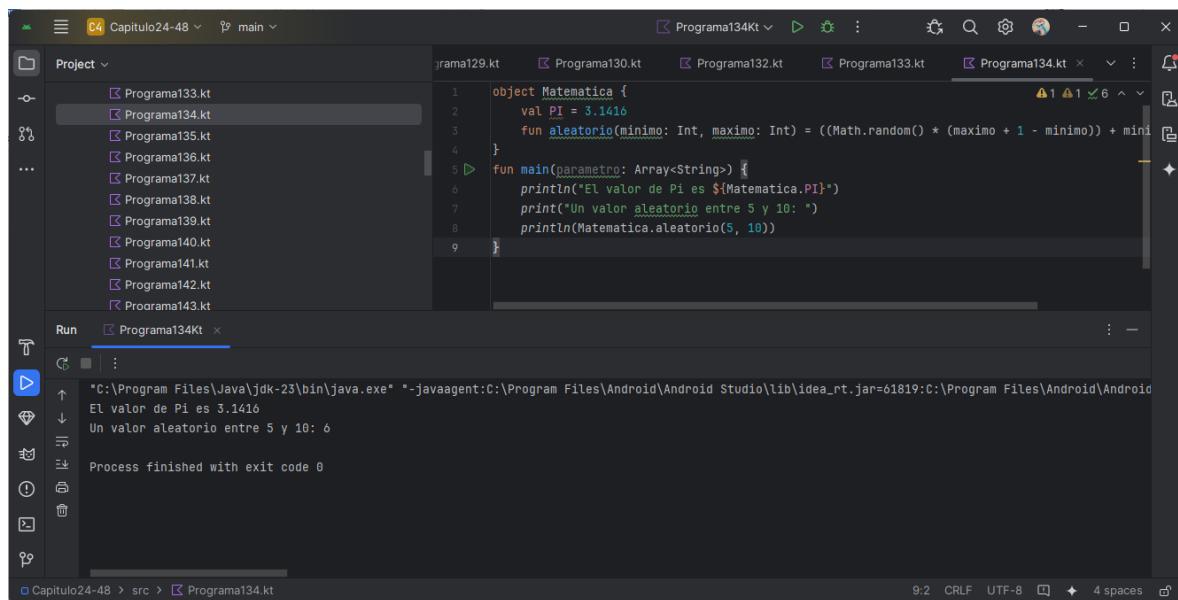
fun main(parametro: Array<String>) {
    ...
}
```

## Capítulo 31

## Problema 1

En el proyecto 134 se define un objeto singleton llamado “Matematica” utilizando la palabra clave object. Un objeto en Kotlin es similar a una clase, pero se crea solo una vez y no se pueden crear instancias adicionales de él. En este caso, el objeto Matematica tiene una propiedad PI que almacena el valor de pi (3.1416) y un método aleatorio que genera un número aleatorio entre dos valores enteros (minimo y maximo). El método aleatorio utiliza Math.random() para calcular un número aleatorio dentro del rango especificado y lo convierte en un valor entero.

En la función main, se accede al valor de PI del objeto Matematica y se imprime, seguido de un número aleatorio generado entre 5 y 10, que también se obtiene usando el método aleatorio del objeto. Para llamar a las propiedades y métodos del objeto, se antepone el nombre del objeto (Matematica) seguido del nombre de la propiedad o método. El resultado se muestra en la consola.



The screenshot shows the Android Studio interface with the project 'Capítulo24-48' open. The 'src' folder contains several Kotlin files, with 'Programa134.kt' selected. The code editor displays the following code:

```
object Matematica {
    val PI = 3.1416
    fun aleatorio(minimo: Int, maximo: Int) = ((Math.random() * (maximo + 1 - minimo)) + minimo)
}

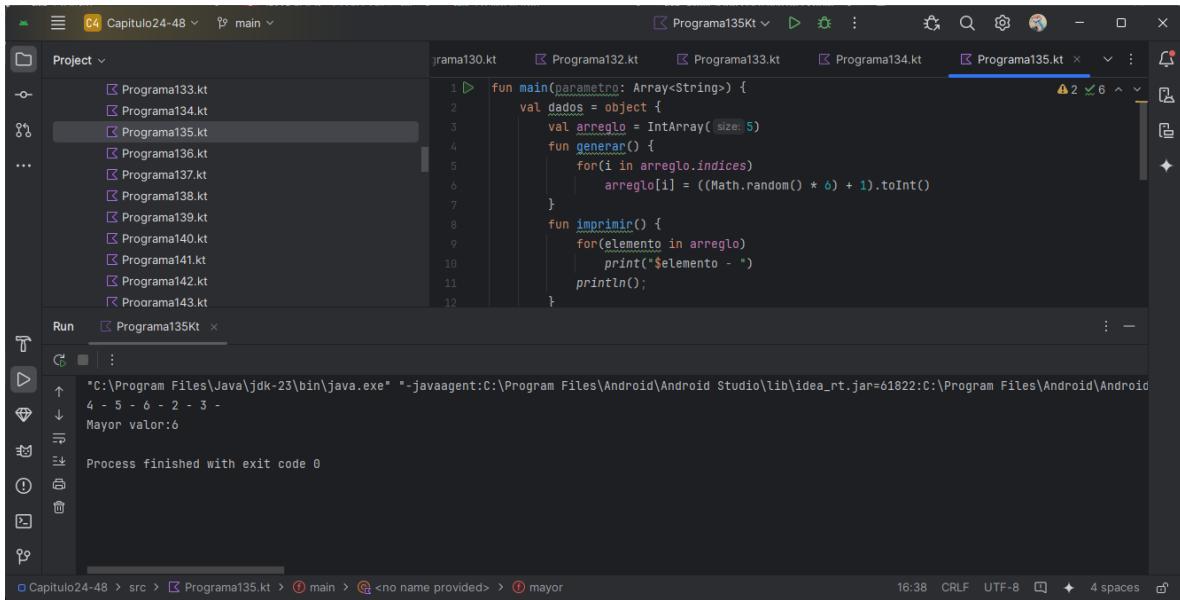
fun main(parametro: Array<String>) {
    println("El valor de Pi es ${Matematica.PI}")
    print("Un valor aleatorio entre 5 y 10: ")
    println(Matematica.aleatorio(5, 10))
}
```

The 'Run' tab shows the output of the program execution:

```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=61819:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8 Programa134Kt
El valor de Pi es 3.1416
Un valor aleatorio entre 5 y 10: 6
Process finished with exit code 0
```

## Problema 2

En el proyecto 135 se define un objeto anónimo dentro de la función main, utilizando la palabra clave object sin asignarle un nombre explícito, lo que permite crear un objeto local y temporal. Este objeto se asigna a la variable dados, y contiene un arreglo de 5 enteros junto con tres métodos: generar(), que llena el arreglo con números aleatorios entre 1 y 6; imprimir(), que imprime los valores del arreglo separándolos con guiones; y mayor(), que encuentra y devuelve el valor máximo del arreglo. Al ejecutar el programa, primero se generan los valores aleatorios, luego se imprimen los valores del arreglo y finalmente se muestra el valor más alto en el arreglo.

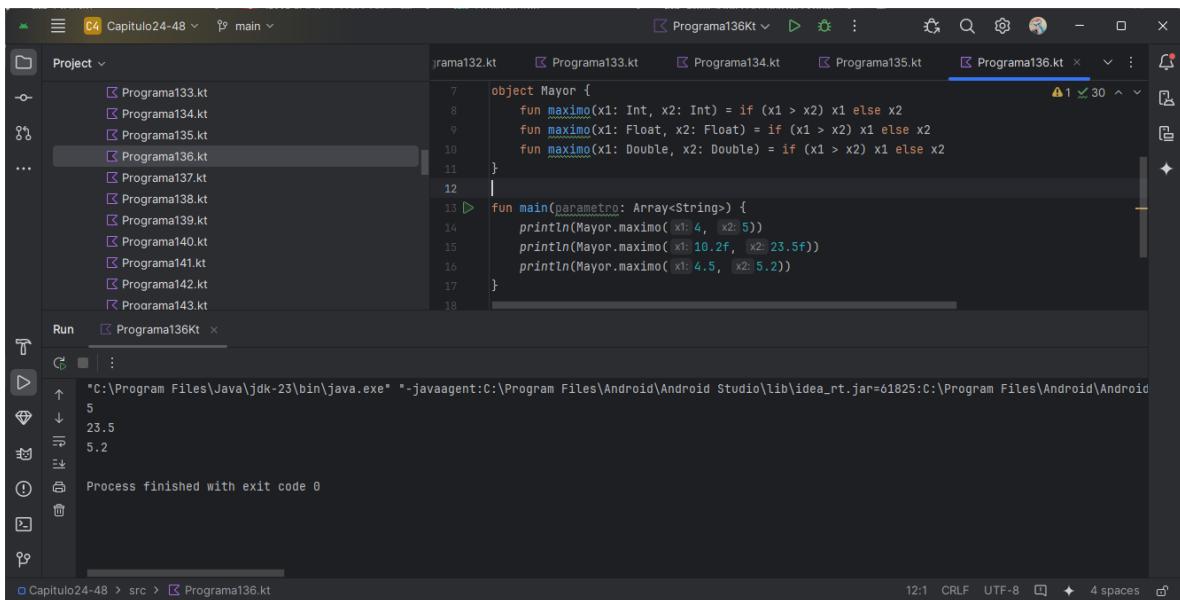


```
1 fun main(parametro: Array<String>) {
2     val datos = object {
3         val arreglo = IntArray( size: 5)
4         fun generar() {
5             for(i in arreglo.indices)
6                 arreglo[i] = ((Math.random() * 6) + 1).toInt()
7         }
8         fun imprimir() {
9             for(elemento in arreglo)
10                 print("$elemento - ")
11         }
12     }
13 }
```

"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea\_rt.jar=61822:C:\Program Files\Android\Android Studio\lib\proguard.jar=61822:lib\apex.jar=61822" Mayor valor:6  
Process finished with exit code 0

### Problema 3

En el proyecto 136 se define un objeto llamado “Mayor” que contiene tres métodos maximo, cada uno de los cuales acepta dos parámetros de tipo diferente: Int, Float y Double. Estos métodos comparan los dos valores que reciben y retornan el mayor de ellos, utilizando una estructura condicional if para determinar cuál es el mayor. En la función main, se llaman a los tres métodos con diferentes tipos de datos: enteros, flotantes y dobles, y se imprime el resultado de cada uno.



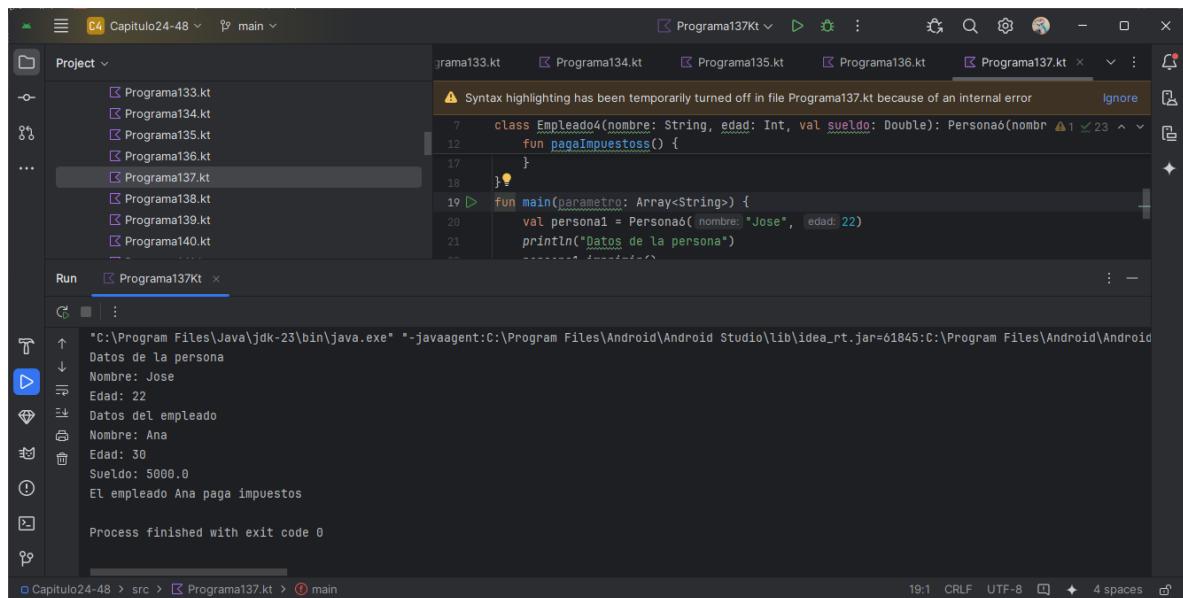
```
7 object Mayor {
8     fun maximo(x1: Int, x2: Int) = if (x1 > x2) x1 else x2
9     fun maximo(x1: Float, x2: Float) = if (x1 > x2) x1 else x2
10    fun maximo(x1: Double, x2: Double) = if (x1 > x2) x1 else x2
11 }
12
13 fun main(parametro: Array<String>) {
14     println(Mayor.maximo( x1: 4, x2: 5))
15     println(Mayor.maximo( x1: 10.2f, x2: 23.5f))
16     println(Mayor.maximo( x1: 4.5, x2: 5.2))
17 }
```

"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea\_rt.jar=61825:C:\Program Files\Android\Android Studio\lib\proguard.jar=61825:lib\apex.jar=61825" 5  
23.5  
5.2  
Process finished with exit code 0

## Capítulo 32

### Problema 1

En el proyecto 137 el programa define una clase base “Persona6” que tiene dos propiedades: nombre y edad, y un método imprimir que muestra estos valores. La clase Empleado4 hereda de Persona6 y agrega una propiedad adicional sueldo. En Empleado4, se sobrescribe el método imprimir para incluir también el sueldo del empleado, utilizando la palabra clave override y la función super para llamar al método imprimir de la clase base. Además, la clase Empleado4 define un método pagalmpuestoss, que determina si el empleado paga impuestos según su sueldo (si es mayor a 3000, paga impuestos). En la función main, se crean instancias de ambas clases (Persona6 y Empleado4), se imprime la información de cada una y se llama al método pagalmpuestoss para la instancia de Empleado4.



```

C4 Capítulo24-48
Programa137Kt
main

Project
  Programa133.kt
  Programa134.kt
  Programa135.kt
  Programa136.kt
  Programa137.kt
  Programa138.kt
  Programa139.kt
  Programa140.kt
  ...
Run Programa137Kt
  C:
    "C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=61845:C:\Program Files\Android\Android Studio\bin" -Dfile.encoding=UTF-8
    Datos de la persona
      ↓
      Nombre: Jose
      Edad: 22
      Datos del empleado
        ↓
        Nombre: Ana
        Edad: 30
        Sueldo: 5000.0
        El empleado Ana paga impuestos
    Process finished with exit code 0

19:1 CRLF UTF-8 4 spaces

```

## Problema 2

En el proyecto 138 se define una clase “Calculadora” con operaciones básicas (suma, resta, multiplicación y división) y un método imprimir() para mostrar el resultado. La clase CalculadoraCientifica hereda de Calculadora y agrega métodos para calcular el cuadrado y la raíz cuadrada de un número. En la función main, se crean objetos de ambas clases, realizan operaciones y se imprime el resultado después de cada una. La propiedad resultado en la clase Calculadora es pública, lo que permite a CalculadoraCientifica acceder a ella directamente.

```
fun main(parametros: Array<String>) {
    println("Prueba de la clase Calculadora (suma de dos números)")
    val calculadora1 = Calculadora( valor1: 10.0, valor2: 2.0 )
    calculadora1.sumar()
    calculadora1.imprimir()
    println(
        "Prueba de la clase Calculadora Científica (suma de dos números y el cuadrado y la raíz del primero)"
    )
    val calculadora2 = CalculadoraCientifica( valor1: 10.0, valor2: 2.0 )
    calculadora2.sumar()
    calculadora2.imprimir()
}
```

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor with the file `Programa138.kt` open. The code defines two classes, `Calculadora` and `CalculadoraCientifica`, which inherit from a common base class. The `main` function tests both classes by creating instances and calling their methods. The run tab shows the output of the program, which includes the printed test descriptions and the results of the calculations.

### Problema 3

En el proyecto 139 se definen dos clases relacionadas por herencia para simular un dado. La clase `Dado10` tiene una propiedad llamada `valor`, que por defecto es 1, y un método `tirar()` que genera un número aleatorio entre 1 y 6. El método `imprimir()` muestra el valor del dado. La clase `DadoRecuadro` hereda de `Dado10` y sobrescribe el método `imprimir()` para mostrar el valor dentro de un cuadro de asteriscos. En la función `main`, se crean dos objetos: uno de la clase `Dado10`, el cual genera un número aleatorio y lo imprime, y otro de la clase `DadoRecuadro`, que hace lo mismo, pero mostrando el valor dentro de un recuadro hecho de asteriscos.

```
class DadoRecuadro: Dado10() {
}

fun main(parametros: Array<String>) {
    val dado1 = Dado10()
    dado1.tirar()
    dado1.imprimir()

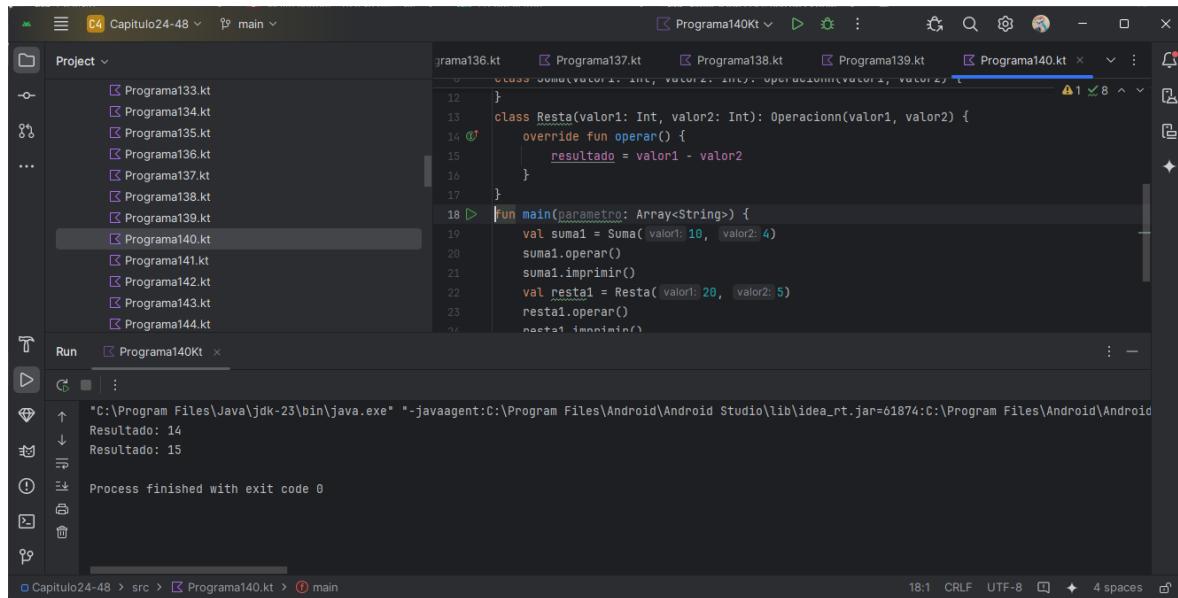
    val dado2 = DadoRecuadro()
    dado2.tirar()
    dado2.imprimir()
}
```

This screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor with the file `Programa139.kt` open. The code defines a class `DadoRecuadro` that extends `Dado10`. The `main` function creates instances of both classes and calls their `tirar` and `imprimir` methods. The output window shows the results, including the use of asterisks to frame the output of the `DadoRecuadro` object's `imprimir` method.

## Capítulo 33

## Problema 1

En el proyecto 140 el código define una estructura utilizando clases abstractas y herencia para realizar operaciones matemáticas básicas. La clase abstracta Operacion tiene dos propiedades: valor1 y valor2, que son los dos números sobre los cuales se realiza la operación, y una propiedad resultado que almacena el resultado de la operación. La clase también tiene un método abstracto operar(), que debe ser implementado por las subclases, y un método concreto imprimir() que muestra el resultado. La clase Suma y la clase Resta heredan de Operacion y proporcionan su propia implementación del método operar() para realizar la suma y la resta, respectivamente. En la función main, se crean instancias de Suma y Resta, se realizan las operaciones correspondientes y luego se imprime el resultado. Dado que Operacion es abstracta, no se pueden crear objetos directamente de esta clase, solo de sus subclases. Esta estructura permite que otras operaciones puedan ser fácilmente agregadas en el futuro, siguiendo el mismo patrón.



The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor with the file 'Programa140.kt' open. The code defines an abstract class 'Operacion' with properties 'valor1' and 'valor2', and methods 'operar()' and 'imprimir()'. It also defines two subclasses, 'Suma' and 'Resta', each overriding 'operar()' to perform addition and subtraction respectively. The 'main()' function creates instances of both classes and prints their results. The bottom panel shows the run log with the output of the program's execution.

```
12 }
13 class Resta(valor1: Int, valor2: Int): Operacion(valor1, valor2) {
14     override fun operar() {
15         resultado = valor1 - valor2
16     }
17 }
18 fun main(parametro: Array<String>) {
19     val suma1 = Suma(valor1: 10, valor2: 4)
20     suma1.operar()
21     suma1.imprimir()
22     val resta1 = Resta(valor1: 20, valor2: 5)
23     resta1.operar()
24     resta1.imprimir()
}
*C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=61874:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8 main
Resultado: 14
Resultado: 15
Process finished with exit code 0
18:1 CRLF UTF-8 4 spaces
```

## Problema 2

En el proyecto 141 se define una estructura de clases utilizando una clase abstracta Cuenta y dos subclases: CajaAhorro y PlazoFijo, que representan dos tipos de cuentas bancarias. La clase abstracta Cuenta tiene las propiedades comunes entre las dos cuentas: el nombre del titular y el monto de la cuenta. Además, define un método imprimir(), que es común a ambas subclases y se encarga de mostrar los datos generales de la cuenta. La subclase CajaAhorro hereda de Cuenta y no añade ninguna funcionalidad adicional, simplemente llama al método imprimir() de la clase base para mostrar los datos de la cuenta. Por otro lado, la subclase PlazoFijo añade dos propiedades específicas: plazo, que es el número de días de imposición del plazo fijo, y interes, que representa la tasa de interés aplicable. En su método imprimir(), también calcula el monto de la ganancia por interés y la muestra junto

con los demás detalles. En la función main, se crean dos objetos: uno de la clase CajaAhorro y otro de la clase PlazoFijo, y se llaman sus métodos imprimir() para mostrar los detalles de cada cuenta.

The screenshot shows the Android Studio interface with the project 'Capítulo24-48' open. The code editor displays 'Programa141.kt' with the following content:

```
29 }
30
31 fun main(parametro: Array<String>) {
32     val cajaAhorro1 = CajaAhorro( titular: "juan", monto: 10000.0)
33     cajaAhorro1.imprimir()
34     val plazoFijo1 = PlazoFijo( titular: "Ana", monto: 5000.0, plazo: 30, interes: 1.23)
35     plazoFijo1.imprimir()
36 }
```

The 'Run' tab shows the output of the program:

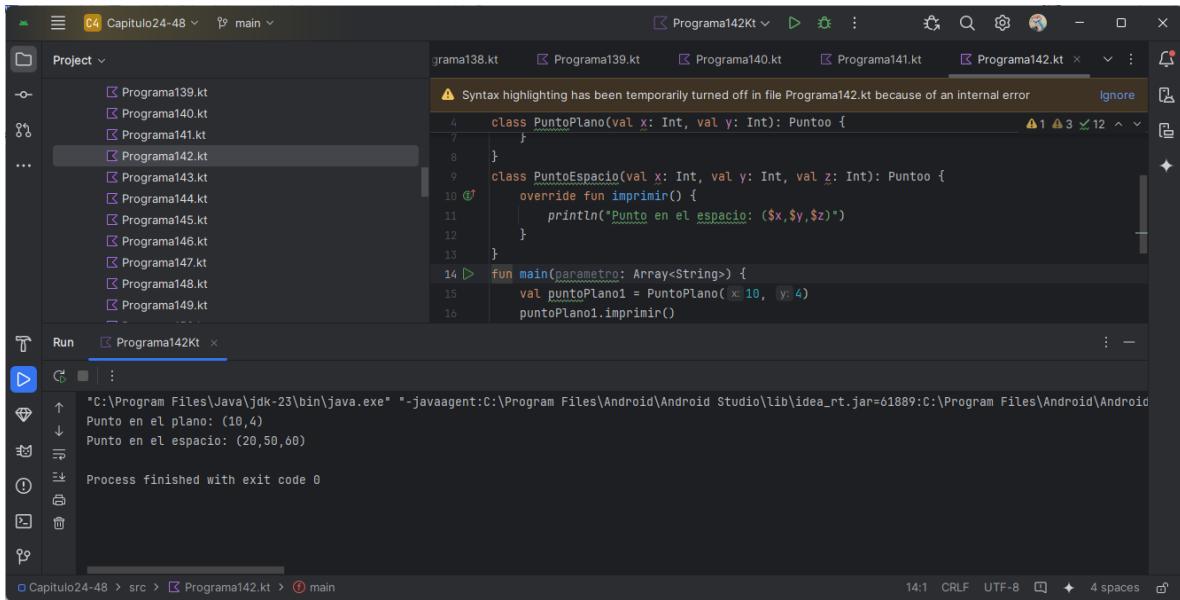
```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=61881:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8
Cuenta de caja de ahorro
Titular: juan
Monto: 10000.0
Cuenta de plazo fijo
Plazo en dias: 30
Intereses: 1.23
Importe del interes: 61.5
Titular: Ana
Monto: 5000.0

Process finished with exit code 0
```

## Capítulo 34

### Problema 1

En el proyecto 142 el código define una interfaz Punto con un método imprimir(), que es implementado por dos clases: PuntoPlano y PuntoEspacio. PuntoPlano maneja dos coordenadas (x y y) y muestra un punto en el plano 2D, mientras que PuntoEspacio maneja tres coordenadas (x, y, z) y muestra un punto en el espacio 3D. En la función main, se crean objetos de ambas clases y se llama al método imprimir() para mostrar sus respectivos puntos.



```
class PuntoPlano(val x: Int, val y: Int): Punto {
    ...
}

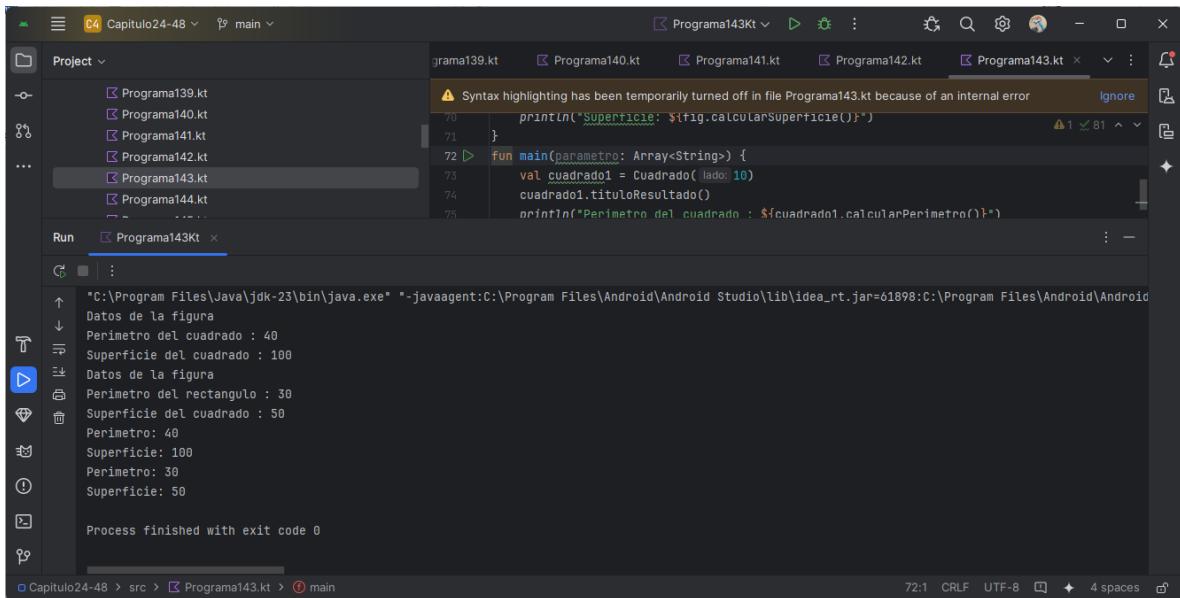
class PuntoEspacio(val x: Int, val y: Int, val z: Int): Punto {
    override fun imprimir() {
        println("Punto en el plano: ($x,$y)")
    }
}

fun main(parametro: Array<String>) {
    val puntoPlano1 = PuntoPlano(10, 4)
    puntoPlano1.imprimir()
}
```

## Problema 2

En el proyecto 143 se define una interfaz “Figura” con métodos abstractos calcularSuperficie() y calcularPerimetro(), y un método concreto tituloResultado() que imprime “Datos de la figura”. Las clases Cuadrado y Rectangulo implementan esta interfaz, con métodos para calcular el área y el perímetro de cada figura.

En la función main, se crean instancias de Cuadrado y Rectangulo, se invoca el método tituloResultado() y luego se muestran los cálculos del perímetro y la superficie de cada figura. Además, se utiliza una función imprimir() que recibe un objeto de tipo Figura y muestra sus resultados.



```
interface Figura {
    fun calcularSuperficie(): Double
    fun calcularPerimetro(): Double
    fun tituloResultado(): String
}

class Cuadrado(lado: Double) : Figura {
    override fun calcularSuperficie(): Double {
        return lado * lado
    }

    override fun calcularPerimetro(): Double {
        return lado * 4
    }

    override fun tituloResultado(): String {
        return "Datos de la figura"
    }
}

class Rectangulo(ancho: Double, alto: Double) : Figura {
    override fun calcularSuperficie(): Double {
        return ancho * alto
    }

    override fun calcularPerimetro(): Double {
        return (ancho + alto) * 2
    }

    override fun tituloResultado(): String {
        return "Datos de la figura"
    }
}

fun main(parametro: Array<String>) {
    val cuadrado1 = Cuadrado(10)
    cuadrado1.tituloResultado()
    println("Perímetro del cuadrado : ${cuadrado1.calcularPerimetro()}")
    println("Superficie del cuadrado : ${cuadrado1.calcularSuperficie()}")
}
```

## Capítulo 35

### Problema 1

En el proyecto 144 se define una clase “Persona” con dos propiedades: nombre y edad, y dos métodos: imprimir(), que muestra los datos de la persona, y esMayor(), que determina si la persona es mayor de edad al verificar si su edad es mayor o igual a 18. En la función main, se crea un arreglo llamado personas que contiene varias instancias de la clase Persona. Luego, se recorre el arreglo con un ciclo for para imprimir los datos de cada persona. Además, se utiliza otro ciclo for para contar cuántas personas son mayores de edad, incrementando un contador cant cada vez que se encuentra una persona mayor de 18 años. Finalmente, se imprime el total de personas mayores de edad.

The screenshot shows the Android Studio interface with the project navigation bar at the top. The 'Programa144.kt' file is selected in the project tree. The code editor displays the following Kotlin code:

```
class Persona(val nombre: String, val edad: Int) {
    fun imprimir() {
        println("Nombre: $nombre Edad: $edad")
    }
    fun esMayor() = if (edad >= 18) true else false
}

fun main(parametro: Array<String>) {
    val personas: Array<Person> = arrayOf(Person(nombre: "ana", edad: 22), Person(nombre: "juan", edad: 13), Person(nombre: "carlos", edad: 6), Person(nombre: "maria", edad: 72))
    println("Listado de personas")
    for (per in personas) {
        per.imprimir()
    }
    val cant = personas.filter { it.esMayor() }.size
    println("Cantidad de personas mayores de edad: $cant")
}
```

The run tab shows the output of the program:

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=61901:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8 Programa144Kt
Listado de personas
Nombre: ana Edad: 22
Nombre: juan Edad: 13
Nombre: carlos Edad: 6
Nombre: maria Edad: 72
Cantidad de personas mayores de edad: 2
Process finished with exit code 0
```

### Problema 2

En el proyecto 145 se define un data class llamado “Articuloo”, que tiene tres propiedades: codigo, descripcion y precio. Luego, se crea un arreglo articulos de tipo Articuloo, que contiene cuatro productos con sus respectivos valores. Se implementan dos funciones: imprimir, que recibe el arreglo de artículos y muestra en consola los detalles de cada uno (código, descripción y precio), y aumentarPrecio, que aumenta el precio de todos los artículos en un 10%. En la función main, primero se imprime la lista de productos con sus precios actuales usando la función imprimir, luego se aplica el aumento de 10% con la función aumentarPrecio y finalmente se vuelve a imprimir la lista de artículos con los precios actualizados.

```
Programma145Kt
```

```
Programma141.kt Programma142.kt Programma143.kt Programma144.kt Programma145.kt
```

```
17
18 fun main(parametro: Array<String>) {
19     val articulos: Array<Articulo> = arrayOf(Articulo(codigo: 1, descripcion: "papas", precio: 7.5),
20                                                 Articulo(codigo: 2, descripcion: "manzanas", precio: 23.0),
21                                                 Articulo(codigo: 3, descripcion: "naranjas", precio: 12.0),
22                                                 Articulo(codigo: 4, descripcion: "cebolla", precio: 21.0))
23
24     public final data class Articulo(
25         val codigo: Int,
26         val descripcion: String,
27         val precio: Float
28     )
29
30     Programma145.kt
31     KotlinYA
```

```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=61914:C:\Program Files\Android\Android Studio\bin" -Dfile.encoding=UTF-8
Listado de precios actual
Código: 1 - Descripción papas Precio: 7.5
Código: 2 - Descripción manzanas Precio: 23.0
Código: 3 - Descripción naranjas Precio: 12.0
Código: 4 - Descripción cebolla Precio: 21.0

Listado de precios con aumento del 10%
Código: 1 - Descripción papas Precio: 8.25
Código: 2 - Descripción manzanas Precio: 25.52
Código: 3 - Descripción naranjas Precio: 13.2
Código: 4 - Descripción cebolla Precio: 23.1

Process finished with exit code 0
```

### Problema 3

En el proyecto 146 se declara una clase llamada “Dadoo”, que tiene una propiedad llamada valor de tipo Int, la cual representa el valor del dado. La clase también tiene dos métodos: tirar, que asigna un valor aleatorio entre 1 y 6 al dado, utilizando Math.random(), y imprimir, que muestra el valor actual del dado en la consola.

En la función main, se define un arreglo llamado dados que contiene 5 objetos de tipo Dadoo. Luego, se recorre el arreglo para llamar al método tirar de cada dado, lo que genera un valor aleatorio para cada uno. Posteriormente, se recorre nuevamente el arreglo y se llama al método imprimir para mostrar los valores de los dados en la consola.

```
Programma146Kt
```

```
Programma141.kt Programma142.kt Programma143.kt Programma144.kt Programma145.kt Programma146.kt
```

```
5
6 class Dadoo(var valor: Int = 1){
7 }
8
9 fun main(parametro: Array<String>) {
10     var dados: Array<Dadoo> = arrayOf(Dadoo(), Dadoo(), Dadoo(), Dadoo(), Dadoo())
11
12     for(dado in dados)
13         dado.tirar()
14
15     for(dado in dados)
16         dado.imprimir()
17 }
```

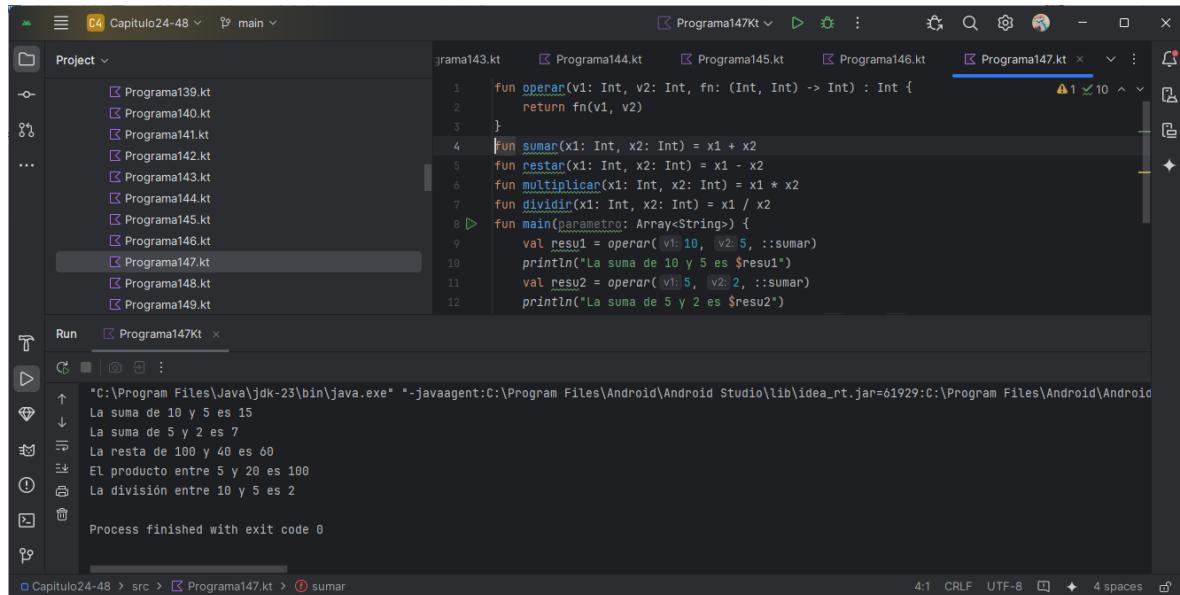
```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=61914:C:\Program Files\Android\Android Studio\bin" -Dfile.encoding=UTF-8
Valor del dado: 2
Valor del dado: 2
Valor del dado: 6
Valor del dado: 3
Valor del dado: 3

Process finished with exit code 0
```

## Capítulo 36

### Problema 1

En el proyecto 147 se define una función de orden superior operar que toma dos enteros y una función como parámetros, ejecutando la función con los enteros y devolviendo el resultado. Hay cuatro funciones matemáticas (sumar, restar, multiplicar, dividir) que realizan operaciones básicas. En la función main, se llama a operar con diferentes funciones matemáticas y se imprimen los resultados.



The screenshot shows the Android Studio interface with the project 'Capítulo24-48' open. The 'Programa147.kt' file is selected in the Project tool window. The code defines a function 'operar' that takes two integers and a function 'fn' as parameters, returning the result of applying 'fn' to the two integers. It also defines four helper functions: 'sumar' (addition), 'restar' (subtraction), 'multiplicar' (multiplication), and 'dividir' (division). The 'main' function demonstrates the use of 'operar' with these helpers to calculate sums and products.

```
fun operar(v1: Int, v2: Int, fn: (Int, Int) -> Int) : Int {
    return fn(v1, v2)
}

fun sumar(x1: Int, x2: Int) = x1 + x2
fun restar(x1: Int, x2: Int) = x1 - x2
fun multiplicar(x1: Int, x2: Int) = x1 * x2
fun dividir(x1: Int, x2: Int) = x1 / x2

fun main(parametro: Array<String>) {
    val resu1 = operar( v1: 10, v2: 5, ::sumar)
    println("La suma de 10 y 5 es $resu1")
    val resu2 = operar( v1: 5, v2: 2, ::sumar)
    println("La suma de 5 y 2 es $resu2")
}
```

### Problema 2

En el proyecto 148 se define una clase “Persona” con dos propiedades: nombre y edad. La clase tiene un método esMayor que toma una función como parámetro y determina si la persona es mayor según la función proporcionada. Existen dos funciones externas, mayorEstados Unidos y mayorArgentina, que verifican si la edad de la persona cumple con los requisitos de edad mínima en cada país (21 años para Estados Unidos y 18 años para Argentina). En el main, se crea un objeto Persona y se verifica si la persona es mayor en ambos países, pasando las funciones correspondientes como parámetros a esMayor.

The screenshot shows the Android Studio interface with the following details:

- Project Tree:** Shows files like Programa139.kt, Programa140.kt, Programa141.kt, Programa142.kt, Programa143.kt, Programa144.kt, Programa145.kt, Programa146.kt, Programa147.kt, Programa148.kt, Programa149.kt, and Programa150.kt.
- Code Editor:** Displays the following Kotlin code in Programa148.kt:

```
6     fun mayorEstadosUnidos(edad: Int): Boolean {
7         return false
8     }
9
10    fun mayorArgentina(edad: Int): Boolean {
11        if (edad >= 18)
12            return true
13        else
14            return false
15    }
16
17    fun main(parametro: Array<String>) {
18        val personal1 = Personas( nombre: "juan", edad: 18)
19        if (personal1.esMayor(::mayorArgentina))
20            println("El personal ${personal1.nombre} es mayor si vive en Argentina")
21    }
22}
```
- Run Tab:** Set to "Programa148Kt".
- Output Tab:** Shows the command run and the output:

```
*C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=61934:C:\Program Files\Android\Android Studio\bin" -Dfile.encoding=UTF-8 Programa148Kt
juan es mayor si vive en Argentina
juan no es mayor si vive en Estados Unidos
Process finished with exit code 0
```
- Status Bar:** Shows "17:2 CRLF UTF-8" and "4 spaces".

## Capítulo 37

### Problema 1

En el proyecto 149 el código muestra cómo utilizar funciones de orden superior en Kotlin, específicamente una función llamada operar que recibe dos valores enteros y una expresión lambda como parámetros. Esta lambda representa una operación matemática que también debe aceptar dos enteros y devolver un entero. En la función main, se realizan tres llamadas a operar, pasando diferentes expresiones lambda: una para sumar, otra para restar y una más compleja que eleva el primer valor al exponente indicado por el segundo. Este último caso demuestra que las expresiones lambda pueden contener varias líneas y estructuras de control como bucles. Además, se presenta una forma más legible y común en Kotlin para escribir lambdas cuando son el último parámetro de una función, colocándolas fuera de los paréntesis.

The screenshot shows the Android Studio interface with the following details:

- Project Tree:** Shows files like Programa147.kt, Programa148.kt, Programa149.kt, Programa151.kt, Programa152.kt, Programa153.kt, Programa154.kt, Programa155.kt, Programa156.kt, Programa157.kt, and Programa158.kt.
- Main Editor:** Displays the content of Programa149.kt:

```
1 fun operarr(v1: Int, v2: Int, fn: (Int, Int) -> Int) : Int{  
2     return fn(v1, v2)  
3 }  
4 fun main(parametro: Array<String>) {  
5     val suma = operarr( 1, 2, {x, y -> x + y})  
6     println(suma)  
7     val resta = operarr( 12, 2, {x, y -> x - y})  
8     println(resta)  
9     var elevarcuarta = operarr( 2, 4, {x, y ->  
10        var valor = 1  
11        for(i in 1 .. y)  
12            valor = valor * x  
13    }  
14    valor  
15 }  
16 }
```
- Run Tab:** Set to "Programa149Kt".
- Output Tab:** Shows the command run: "C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea\_rt.jar=61939:C:\Program Files\Android\Android Studio\bin" "C:\Users\user\AndroidStudioProjects\Capitulo24-48\src\main\kotlin\Programa149Kt.kt". It also shows the process finished with exit code 0.
- Status Bar:** Shows file encoding as 4:1 CRLF, character set as UTF-8, and code style as 4 spaces.

## Problema 2

En el proyecto 150, por su parte, nos muestra cómo utilizar funciones de orden superior y expresiones lambda en Kotlin para aplicar distintos filtros sobre un arreglo de enteros. La función imprimirSi recibe un arreglo y una función lambda que determina si cada elemento debe ser impreso. En main, se genera un arreglo de 10 números aleatorios entre 0 y 99, y se llama a imprimirSi con distintas lambdas para imprimir valores que cumplen ciertas condiciones: múltiplos de 2, múltiplos de 3 o 5, valores mayores o iguales a 50, valores dentro de ciertos rangos específicos, o simplemente todos los valores del arreglo. Inicialmente, las lambdas usan una sintaxis explícita con un parámetro x, pero luego se optimiza utilizando la palabra reservada it, ya que en Kotlin, cuando una lambda tiene un solo parámetro, se puede omitir su declaración y utilizar directamente it.

```
Programa150.kt
fun imprimirSi(arreglo: IntArray, fn:Int) -> Boolean {
    for(i in arreglo.indices)
        if(fn(i))
            return true
    return false
}

fun main(parametro: Array<String>) {
    val arreglo1 = IntArray( size: 10)
    for(i in arreglo1.indices)
        arreglo1[i] = i*2
    if(imprimirSi(arreglo1){it % 3 == 0})
        println("Imprimir los valores múltiplos de 3 o de 5")
    else
        println("Imprimir los valores mayores o iguales a 50")
    if(imprimirSi(arreglo1){it > 10})
        println("Imprimir los valores comprendidos entre 1 y 10, 20 y 30, 90 y 95")
    else
        println("Imprimir todos los valores")
}
Process finished with exit code 0
```

### Problema 3

En el proyecto 151 se utiliza una función de orden superior llamada filtrar, que permite extraer caracteres de una cadena según una condición definida con una expresión lambda. En el main, se aplican tres filtros distintos: uno para obtener solo las vocales, otro para las letras minúsculas y otro para los caracteres no alfabéticos.

```
Programa151.kt
fun filtrar(cadena: String, fn: (Char) -> Boolean): String {
    var resultado = ""
    for(it in cadena)
        if(fn(it))
            resultado += it
    return resultado
}

fun main(parametro: Array<String>) {
    val cadena="Esto es la prueba 1 o la prueba 2?"
    println("String original")
    println(cadena)
    val resultado1 = filtrar(cadena) {
        if (it == 'a' || it == 'e' || it == 'i' || it == 'o' || it == 'u' ||
            it == 'A' || it == 'E' || it == 'I' || it == 'O' || it == 'U' )
            true
        else
            false
    }
    println("Solo las vocales")
    println(resultado1)
    val resultado2 = filtrar(cadena) {
        if (it.isLetter())
            true
        else
            false
    }
    println("Solo los caracteres en minúsculas")
    println(resultado2)
    val resultado3 = filtrar(cadena) {
        if (it.isDigit())
            true
        else
            false
    }
    println("Solo los caracteres no alfabéticos")
    println(resultado3)
}
Process finished with exit code 0
```

## Capítulo 38

### Problema 1

En el proyecto 152 se genera un arreglo de 20 enteros aleatorios entre 0 y 10, lo imprime y luego analiza sus elementos usando funciones de orden superior como

count, all y any. Cuenta cuántos valores son menores o iguales a 5, verifica si todos los elementos son menores o iguales a 9 y si hay al menos un 10.

```
1 fun main(parametro: Array<String>) {
2     val arreglo = IntArray(size: 20) {(Math.random() * 11).toInt()}
3     println("Listado completo del arreglo")
4     for(elemento in arreglo)
5         println("$elemento")
6     println()
7     val cant1 = arreglo.count { it <= 5}
8     println("Cantidad de elementos menores o iguales a 5: $cant1")
9     if (arreglo.all {it <= 9})
10        println("Todos los elementos son menores o iguales a 9")
```

## Problema 2

En el proyecto 153 se crea un arreglo de 10 números flotantes, solicita al usuario que los ingrese por teclado y luego cuenta cuántos son menores a 50. Finalmente, muestra si todos los valores ingresados cumplen con esa condición usando funciones como count y all para analizar el contenido del arreglo.

```
5 fun main(parametro: Array<String>) {
6     println("Ingrese elemento:")
7     arreglo[1] = readln().toFloat()
8 }
```

```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62019:C:\Program Files\Android\Android
Ingres elemento:4
Ingres elemento:9
Ingres elemento:10
Ingres elemento:12
Ingres elemento:6
Ingres elemento:8
Ingres elemento:39
Ingres elemento:658
Ingres elemento:92
Ingres elemento:50
Listado completo del arreglo
4.0 9.0 10.0 12.0 6.0 8.0 39.0 658.0 92.0 50.0
Cantidad de valores ingresados menores a 50: 7
No todos los valores son menores a 50

Process finished with exit code 0
```

## Capítulo 39

### Problema 1

En el proyecto 154 se genera un arreglo de 10 números enteros aleatorios entre 0 y 99, los imprime y luego utiliza una función de orden superior llamada recorrerTodo, que recibe una función lambda como parámetro, para recorrer el arreglo. Con esta función se cuenta cuántos elementos son múltiplos de 3 y se suma el total de los elementos mayores a 50.

The screenshot shows the Android Studio interface with the project 'Capítulo24-48' open. The 'src' folder contains several Kotlin files (Programa147.kt, Programa148.kt, etc.) and one Java file (Programa154Kt.java). The 'Programa154.kt' file is selected and shown in the code editor. The code defines a function 'recorrerTodo' that takes an IntArray and a lambda function 'fn:(Int) -> Unit'. It then iterates over the array and applies the lambda function to each element. Below it, the 'main' function creates an IntArray 'arreglo1' of size 10, fills it with random integers between 0 and 100, prints the array, and then uses 'recorrerTodo' to count elements divisible by 3 and sum elements greater than 50. The run tab shows the output: 'Impresion de todo el arreglo' followed by a list of 10 random numbers, 'La cantidad de elementos múltiplos de 3 son 2', 'La suma de todos los elementos mayores a 50 es 446', and 'Process finished with exit code 0'.

```
fun recorrerTodo(arreglo: IntArray, fn:(Int) -> Unit) {
    for(elemento in arreglo)
        (fn(elemento))
}

fun main(parametro: Array<String>) {
    val arreglo1 = IntArray( size: 10)
    for (i in arreglo1.indices)
        arreglo1[i] = ((Math.random() * 100)).toInt()
    println("Impresion de todo el arreglo")
    for (elemento in arreglo1)
        print("$elemento ")
    println()
    var cantidad = 0
    var suma = 0
    for (elemento in arreglo1) {
        if (elemento % 3 == 0)
            cantidad++
        if (elemento > 50)
            suma += elemento
    }
    println("La cantidad de elementos múltiplos de 3 son $cantidad")
    println("La suma de todos los elementos mayores a 50 es $suma")
}
```

## Problema 2

En el proyecto 155 se genera un arreglo de 10 enteros aleatorios entre 0 y 99 y luego realiza varias operaciones sobre él utilizando la función de orden superior forEach, que recorre cada elemento del arreglo. Primero, imprime todos los elementos del arreglo. Luego, cuenta cuántos de esos elementos son múltiplos de 3, incrementando un contador cada vez que encuentra un múltiplo de 3. Después, suma todos los elementos mayores a 50. Finalmente, imprime ambos resultados: la cantidad de elementos múltiplos de 3 y la suma de los elementos mayores a 50.

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor with the file `Programa155.kt` open. The code defines a `main` function that creates an array of 10 integers, prints each element, counts how many are multiples of 3, and prints the total sum of elements greater than 50. The run output window shows the console output: "Impresion de todo el arreglo", followed by a list of 10 integers, "La cantidad de elementos múltiples de 3 son 5", "La suma de todos los elementos mayores a 50 es 257", and "Process finished with exit code 0".

```
1 fun main(parametro: Array<String>) {
2     val arreglo1 = IntArray( size: 10)
3     for (i in arreglo1.indices)
4         arreglo1[i] = ((Math.random() * 100)).toInt()
5     println("Impresion de todo el arreglo")
6     for (elemento in arreglo1)
7         print("$elemento ")
8     println()
9     var cantidad = 0
10    arreglo1.forEach {
11        if (it % 3 == 0)
12            cantidad++
13    }
14    println("La cantidad de elementos múltiples de 3 son $cantidad")
15    var suma = 0
16    for (elemento in arreglo1) {
17        if (elemento > 50)
18            suma += elemento
19    }
20    println("La suma de todos los elementos mayores a 50 es $suma")
21 }
```

### Problema 3

En el proyecto 156 se define una clase “Persona” con dos propiedades: nombre y edad, junto con dos métodos: uno para imprimir los datos de la persona (`imprimir`) y otro para verificar si la persona es mayor de edad (`esMayor`). En la función `main`, se crea un arreglo de objetos Persona, cada uno con un nombre y una edad. Luego, el programa imprime el listado de personas utilizando el método `imprimir` de cada objeto. Posteriormente, se utiliza la función `forEach` de la clase `Array` para recorrer cada persona y contar cuántas son mayores de edad (es decir, cuyas edades son 18 o más). Finalmente, el número de personas mayores de edad se imprime en consola.

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor with the file `Programa156.kt` open. The code defines a class `Personaaaa` with properties `nombre` and `edad`, and methods `imprimir` and `esMayor`. The `main` function creates an array of four `Personaaaa` objects and prints each one using `imprimir`. It then uses `forEach` to count how many have an `edad` of 18 or more and prints that count. The run output window shows the console output: "Listado de personas", followed by details for four persons (ana, juan, carlos, maria), "Cantidad de personas mayores de edad: 2", and "Process finished with exit code 0".

```
1 class Personaaaa(val nombre: String, val edad: Int) {
2     fun imprimir() {
3         println("Nombre: $nombre Edad: $edad")
4     }
5     fun esMayor() = if (edad >= 18) true else false
6 }
7 fun main(parametro: Array<String>) {
8     val personas: Array<Personaaaa> = arrayOf(Personaaaa( nombre: "ana", edad: 22),
9                                                 Personaaaa( nombre: "juan", edad: 13),
10                                                Personaaaa( nombre: "carlos", edad: 6),
11                                                Personaaaa( nombre: "maria", edad: 72))
12 }
```

## Problema 4

En el proyecto 157 se define una clase “Dado0”, que tiene una propiedad valor que representa el valor actual del dado (por defecto 1). La clase también incluye dos métodos: tirar, que asigna un valor aleatorio entre 1 y 6 al dado, y imprimir, que muestra el valor del dado en consola. En la función main, se crea un arreglo de 5 objetos Dado0. Cada dado se "tira" (se les asigna un valor aleatorio) y luego se imprime el valor de cada uno. Posteriormente, el programa utiliza una estructura forEach para recorrer los dados y contar cuántos de ellos tienen los valores 1, 2, 3, 4, 5 y 6, incrementando los contadores correspondientes en base al valor del dado. Finalmente, se imprimen las cantidades de dados que resultaron en cada uno de esos valores.

The screenshot shows the Android Studio interface. The top bar displays the project name 'Capítulo24-48' and the file 'main'. The left sidebar shows a 'Project' tree with several Kotlin files. The main editor window contains the following code:

```
fun main(parametro: Array<String>) {
    var dados: Array<Dado0> = arrayOf(Dado0(), Dado0(), Dado0(), Dado0(), Dado0())
    for(dado in dados)
        dado.tirar()
    for(dado in dados)
```

The 'Run' tab is selected, showing the output of the program:

```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62083:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8 Programa157Kt
Valor del dado: 5
Valor del dado: 4
Valor del dado: 6
Valor del dado: 2
Valor del dado: 5
Cantidad de dados que tienen el valor 1: 0
Cantidad de dados que tienen el valor 2: 1
Cantidad de dados que tienen el valor 3: 0
Cantidad de dados que tienen el valor 4: 1
Cantidad de dados que tienen el valor 5: 2
Cantidad de dados que tienen el valor 6: 1

Process finished with exit code 0
```

The bottom status bar shows the path 'Capítulo24-48 > src > Programa157.kt > main' and the time '22:18'.

## Capítulo 40

### Problema 1

En el proyecto 158 se define una función de extensión para la clase String, que permite agregar nuevas funcionalidades a las cadenas de texto sin necesidad de modificar la propia clase String. La primera función de extensión, mitadPrimera, devuelve la primera mitad de la cadena, utilizando el método substring para obtener una subcadena desde el inicio hasta la mitad de la longitud de la cadena original. La segunda función de extensión, segundaMitad, devuelve la segunda mitad de la cadena, utilizando substring para extraer desde la mitad de la longitud hasta el final de la cadena. En el método main, se crea una cadena llamada cadena1 con el valor "Viento", y luego se imprimen tanto la primera mitad como la segunda mitad de la cadena usando las funciones de extensión definidas previamente.

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor with the file `Programa158.kt` open. The code defines three extension functions for the `String` class: `mitadPrimera()`, `segundaMitad()`, and `main()`. The `main()` function creates a string `cadena1` with the value "Viento" and prints its first and second halves. The run tab shows the command used to run the program and the output: `[0 1 2 3 4 5 6 7 8 9]`.

```
1 fun String.mitadPrimera(): String {
2     return this.substring(range: 0 .. < this.length/2-1)
3 }
4 fun String.segundaMitad(): String{
5     return this.substring(range: this.length/2 .. < this.length-1)
6 }
7 fun main(args: Array<String>) {
8     val cadena1 = "Viento"
9     println(cadena1.mitadPrimera())
10    println(cadena1.segundaMitad())
11 }
```

## Problema 2

En el proyecto 159 se define una función de extensión llamada `imprimir` para la clase `IntArray`, que imprime todos los elementos del arreglo. En el main, se crea un arreglo `IntArray` con valores de 0 a 9 utilizando una expresión lambda y luego se llama al método `imprimir` para mostrar el contenido del arreglo en la consola.

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor with the file `Programa159.kt` open. The code defines an extension function `imprimir()` for the `IntArray` class that prints its elements. The `main()` function creates an array `arreglo1` with size 10 and calls its `imprimir()` method. The run tab shows the command used to run the program and the output: `[0 1 2 3 4 5 6 7 8 9]`.

```
1 fun IntArray.imprimir() {
2     print("[")
3     for(elemento in this) {
4         print("$elemento ")
5     }
6     println("]")
7 }
8 fun main(args: Array<String>) {
9     val arreglo1= IntArray( size: 10, {it})
10    arreglo1.imprimir()
11 }
```

## Problema 3

En el proyecto 160 se agrega una función de extensión llamada `imprimir` a la clase `String`, la cual imprime la cadena de texto almacenada en el objeto. En el main, se llama al método `imprimir` sobre la cadena "Hola Mundo" y sobre la variable `cadena1`, que contiene "Fin", mostrando ambas cadenas en la consola.

The screenshot shows the Android Studio interface with the project 'Capítulo24-48' open. The code editor displays 'Programa160.kt' with the following content:

```
4     println(this)
5 }
6
7 fun main(args: Array<String>) {
8     "Hola Mundo".imprimir()
9     val cadena1 = "Fin"
10    cadena1.imprimir()
11 }
```

The run tab shows the output of the program:

```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62116:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8
Hola Mundo
Fin
Process finished with exit code 0
```

## Problema 4

En el proyecto 161 se define una función de extensión llamada hasta para la clase Int. La función permite imprimir los números desde el valor del entero (almacenado en el objeto que llama la función) hasta el valor pasado como parámetro. En el main, se demuestra su uso con dos ejemplos: el valor 10 imprime los números del 10 al 100, y el valor 5 imprime los números del 5 al 10, separados por guiones.

The screenshot shows the Android Studio interface with the project 'Capítulo24-48' open. The code editor displays 'Programa161.kt' with the following content:

```
//PROBLEMA PROPUESTO 161
/*Codificar la función de extensión llamada "hasta" que debe extender la clase Int y tiene que
el valor entero que almacena el objeto hasta el valor que llega como parámetro:
fun Int.hasta(fin: Int) {*/
    fun Int.hasta(fin: Int) {
        for(valor in this .. fin)
            print("$valor-")
        println()
    }
}

fun main(args: Array<String>) {
    val v = 10
    v.hasta(100)
    5.hasta(10)
}
```

The run tab shows the output of the program:

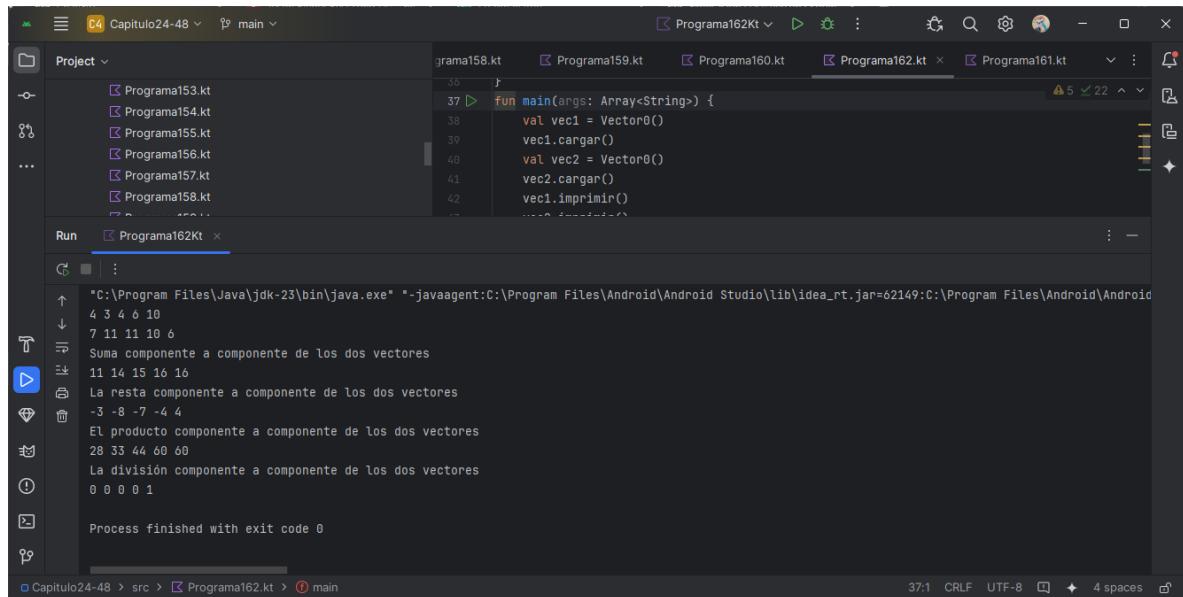
```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62127:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8
10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45-46-47-48-49-50-51-52-53-54-55-56-57-58-59-
5-6-7-8-9-10-
Process finished with exit code 0
```

## Capítulo 41

### Problema 1

En el proyecto 162 se define una clase “Vector0” con un arreglo de 5 enteros, un método cargar() que asigna valores aleatorios y un método imprimir() que muestra

el contenido del arreglo. Además, sobrecarga los operadores +, -, \*, y / para realizar operaciones componente por componente entre dos vectores. En el main, se crean dos vectores, se cargan con valores aleatorios y se realizan operaciones de suma, resta, multiplicación y división, imprimiendo los resultados de cada operación.



```
fun main(args: Array<String>) {
    val vec1 = Vector0()
    vec1.cargar()
    val vec2 = Vector0()
    vec2.cargar()
    vec1.imprimir()
}

4 3 4 6 10
7 11 11 10 6
Suma componente a componente de los dos vectores
11 14 15 16 16
La resta componente a componente de los dos vectores
-3 -8 -7 -4 4
El producto componente a componente de los dos vectores
28 33 44 60 60
La división componente a componente de los dos vectores
0 0 0 0 1

Process finished with exit code 0
```

## Problema 2

En el proyecto 163 se define la clase “Vector1”, que tiene un arreglo de 5 enteros. Incluye un método cargar() para asignar valores aleatorios al arreglo, y un método imprimir() para mostrar los valores. Además, sobrecarga el operador \* para multiplicar cada elemento del arreglo por un valor entero. En la función main, se crea un objeto de tipo Vector1, se carga con valores aleatorios y se imprime. Luego, se realiza una multiplicación del vector por el número 10 usando el operador sobrecargado, y se muestra el resultado. Además, el programa demuestra cómo, al sobrecargar el operador, también se puede realizar la multiplicación en el orden inverso, utilizando una función de extensión que permite que un Int multiplique un Vector.

```
1  class Vector1 {  
2      operator fun times(valor: Int): Vector1 {  
3          }  
4  }  
5  fun main(args: Array<String>) {  
6      val vec1 = Vector1()  
7      vec1.cargar()  
8      vec1.imprimir()  
9      println("El producto de un vector con el número 10 es")  
10     val vecProductoEnt = vec1 * 10  
11     vecProductoEnt.imprimir()  
12 }
```

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor with the file `Programa163.kt` selected. The code defines a `Vector1` class with a `times` operator function that multiplies the vector by an integer. It also contains a `main` function that creates a `Vector1` object, calls `cargar` to fill it with values, prints its original state, and then prints its state after being multiplied by 10. The run output shows the original vector [2 8 1 7] and the resulting vector [20 80 80 10 70].

### Problema 3

En el proyecto 164 el programa define la clase `Vector3`, que tiene un arreglo de 5 enteros. Los métodos `cargar()` y `imprimir()` permiten cargar valores aleatorios en el arreglo e imprimirlos. Se sobrecargan los operadores de incremento (`++`) y decremento (`--`) mediante los métodos `inc()` y `dec()`, los cuales incrementan o decrementan en uno cada elemento del arreglo, respectivamente. En la función `main`, se crea un objeto de tipo `Vector3`, se carga con valores aleatorios y se imprime el vector original. Luego, se aplica el operador `++` para incrementar cada elemento del vector en uno, y después el operador `--` para disminuirlos en uno.

```
1  class Vector3 {  
2      }  
3  fun main(args: Array<String>) {  
4      var vec1 = Vector3()  
5      vec1.cargar()  
6      println("Vector original")  
7      vec1.imprimir()  
8      vec1++  
9      println("Luego de llamar al operador ++")  
10     vec1.imprimir()  
11     vec1--  
12 }
```

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor with the file `Programa164.kt` selected. The code defines a `Vector3` class and a `main` function. The `main` function creates a `Vector3` object, calls `cargar` to fill it with random values, prints its original state, increments each element using the `++` operator, prints the state again, and then decrements each element using the `--` operator. The run output shows the original vector [8 7 11 1 4], the vector after `++` ([9 8 12 2 5]), and the vector after `--` ([8 7 11 1 4]).

### Problema 4

En el proyecto 165 se define la clase “Persona0” que tiene dos propiedades: nombre y edad. Se incluye un método imprimir() que muestra el nombre y la edad de la persona. Además, se sobrecarga el operador compareTo para permitir comparar dos objetos de la clase Persona0 según su edad. El operador devuelve un -1 si la persona es más joven, 1 si es mayor, o 0 si tienen la misma edad. En la función main, se crean dos objetos Persona0, persona1 y persona2, con edades iguales. Luego, se utiliza el operador > y < para determinar cuál de las dos personas es mayor, imprimiendo el resultado correspondiente. Si tienen la misma edad, se imprime un mensaje que lo indica.

```

class Persona0 (val nombre: String, val edad: Int) {
    operator fun compareTo(per2: Persona0): Int {
        if (edad < per2.edad) return -1
        if (edad > per2.edad) return 1
        else return 0
    }
}

fun main(parametro: Array<String>) {
    val persona1 = Persona0("Juan", 30)
    persona1.imprimir()
    val persona2 = Persona0("Ana", 30)
    persona2.imprimir()
}

```

## Problema 5

En el proyecto 166 se define una clase “TaTeTi” que simula un tablero de 3x3 para el juego Tic-Tac-Toe, usando un arreglo unidimensional. Sobreponga los operadores set y get para acceder a las casillas del tablero mediante coordenadas de fila y columna, haciendo el código más legible. El operador set asigna valores a las casillas y el get devuelve su contenido. El método imprimir muestra el estado del tablero tras cada asignación. Este enfoque facilita la manipulación del tablero como si fuera una matriz 2D. Además, menciona la posibilidad de sobreponer el operador de paréntesis con el método invoke.

```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62181:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8
1 0 0
0 0 0
0 0 0
1 0 2
0 0 0
1 0 0
1 0 2
0 0 2
1 0 0
1 0 2
1 0 2
1 0 0
Process finished with exit code 0
```

## Problema 6

En el proyecto 167 se define una clase “Dados” que simula un conjunto de datos mediante un arreglo de 10 enteros. En el método tirar, se asignan valores aleatorios entre 1 y 6 a cada elemento del arreglo. La sobrecarga del operador de paréntesis (invoke) permite acceder a los valores de cada "dado" pasando un índice, lo que facilita la obtención de los resultados de cada tirada. En el main, se crea un objeto dados, se llama al método tirar para asignar los valores aleatorios, y luego se imprime el resultado de cada dado usando el operador sobrecargado dados(nro). También se menciona la posibilidad de sobrecargar otros operadores como +=, -=, \*=, etc., dependiendo de las necesidades del problema.

```
class Dados {
    var arreglo = IntArray(10)
    init {
        for (i in 0..9) {
            arreglo[i] = ((Math.random() * 6) + 1).toInt()
        }
    }
    operator fun invoke(nro: Int) = arreglo[nro]
}

fun main(args: Array<String>) {
    val dados = Dados()
    dados.tirar()
    println(dados[0])
}
```

```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62195:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8
6
2
5
2
1
4
6
4
2
3
Process finished with exit code 0
```

## Problema 7

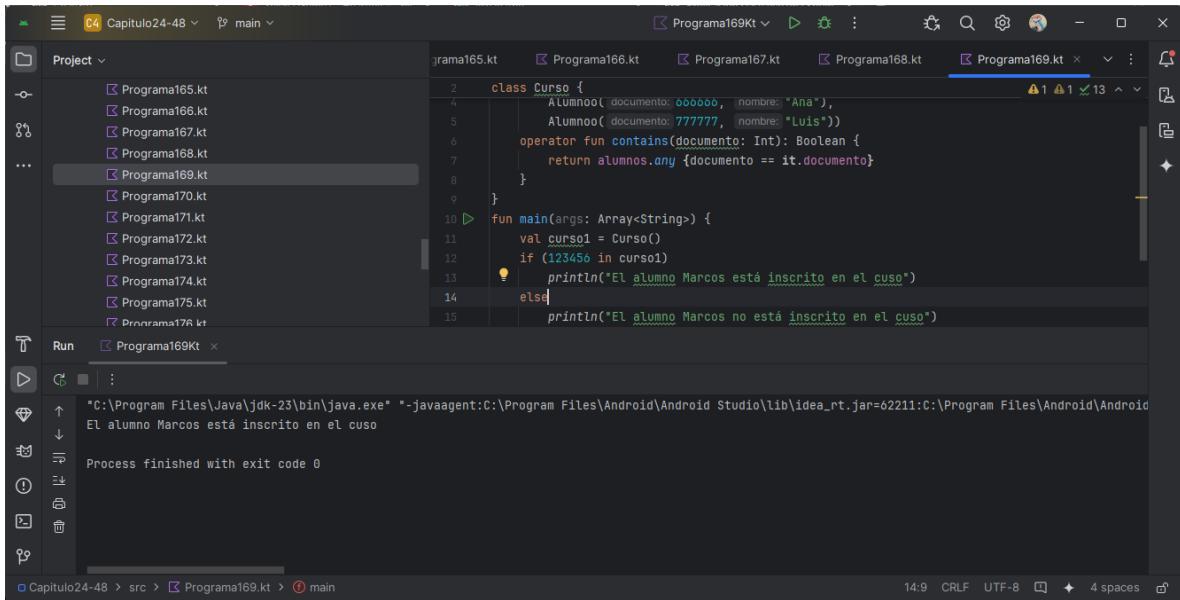
En el proyecto 168 el programa define una clase “Vector” que contiene un arreglo de 5 elementos, donde cada uno es inicializado con su índice correspondiente. La sobrecarga del operador `+=` se implementa en el método `plusAssign`, que toma otro objeto Vector y suma sus elementos correspondientes a los de arreglo. En el main, se crean dos objetos de la clase Vector, se imprimen, y luego el contenido de `vec2` se acumula en `vec1` utilizando el operador `+=`. Finalmente, se imprime el resultado de `vec1` después de la acumulación.

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor with the file `Programa168.kt` open. The code defines a class `Vector4` with a `main` function that prints two vectors and then adds them together. The run tab shows the output of the program running on an emulator, displaying the vectors and their sum. The bottom status bar shows the build number as 20:2 and other system information.

```
class Vector4 {
}
fun main(args: Array<String>) {
    val vec1 = Vector4()
    vec1.imprimir()
    val vec2 = Vector4()
    vec2.imprimir()
    vec1 += vec2
    vec1.imprimir()
}
```

## Problema 8

En el proyecto 169 se define una clase “Alumno”, que tiene dos propiedades: documento (un número entero que representa el documento del alumno) y nombre (el nombre del alumno). La clase `Curso` contiene un arreglo de objetos `Alumno`, y sobrecarga el operador `in` mediante el método `contains`. Este método recibe un número de documento y verifica si existe un alumno con ese documento en el arreglo, devolviendo `true` si lo encuentra y `false` en caso contrario. En la función `main`, se crea una instancia de `Curso`, y se utiliza el operador `in` para comprobar si un alumno con un documento específico (en este caso, 123456) está inscrito en el curso. El resultado se imprime en pantalla.

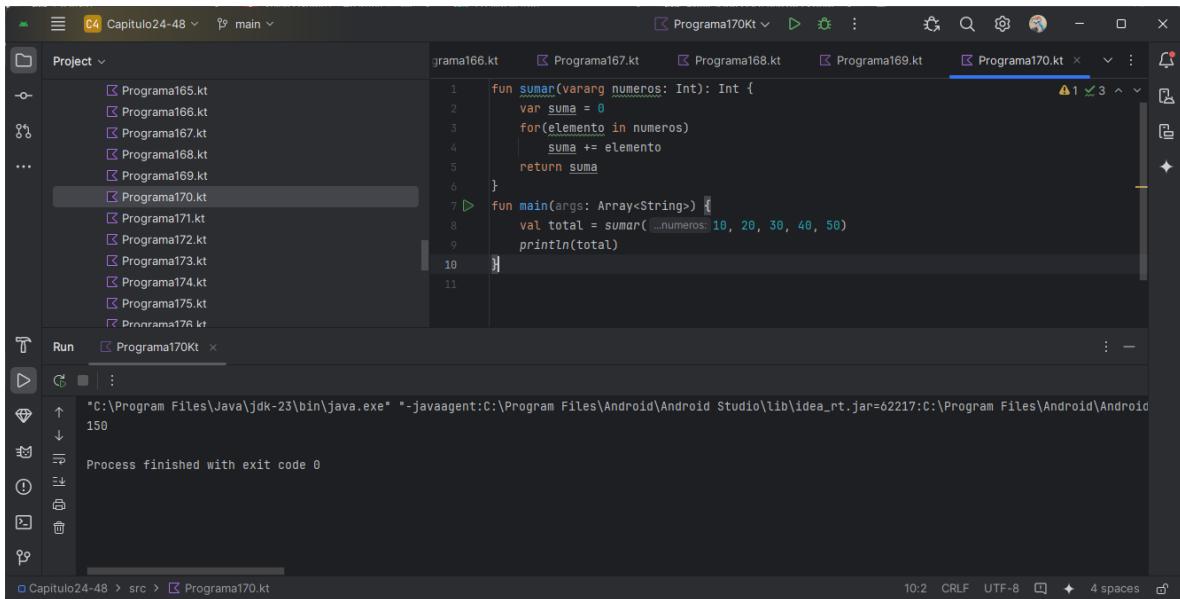


```
2     class Curso {
3         Alumno( documento: 666666, nombre: "Ana"),
4         Alumno( documento: 777777, nombre: "Luis")
5         operator fun contains(documento: Int): Boolean {
6             return alumnos.any {documento == it.documento}
7         }
8     }
9
10 fun main(args: Array<String>) {
11     val curso1 = Curso()
12     if (123456 in curso1)
13         println("El alumno Marcos está inscrito en el cuso")
14     else
15         println("El alumno Marcos no está inscrito en el cuso")
```

## Capítulo 42

### Problema 1

En el proyecto 170 se define una función llamada “sumar” que puede recibir un número variable de argumentos enteros gracias al uso del modificador vararg. Dentro de la función, se recorre cada uno de los números recibidos y se acumulan en una variable suma, la cual se retorna al final. En la función main, se llama a sumar con cinco números: 10, 20, 30, 40 y 50, y luego se imprime el total.



```
1     fun sumar(vararg numeros: Int): Int {
2         var suma = 0
3         for(elemento in numeros)
4             suma += elemento
5         return suma
6     }
7
8     fun main(args: Array<String>) {
9         val total = sumar(10, 20, 30, 40, 50)
10        println(total)
11    }
12
```

### Problema 2

En el proyecto 171 se define una función “operar” que realiza una suma o un promedio sobre una lista de enteros, según el valor recibido desde un enum class llamado

“Operacion”. Utiliza el modificador vararg para permitir un número variable de argumentos. En main, se llama a la función para calcular e imprimir tanto la suma como el promedio de los valores 10, 20 y 30. También se menciona en el libro que, si vararg no es el último parámetro, se deben usar argumentos nombrados, y que se puede usar el operador \* (spread) para pasar un arreglo como lista de argumentos.

```

enum class Operacion{
    SUMA,
    PROMEDIO
}
fun operar(tipoOperacion: Operacion, vararg arreglo: Int): Int {
    when (tipoOperacion) {
        Operacion.SUMA -> return arreglo.sum()
        Operacion.PROMEDIO -> return arreglo.average().toInt()
    }
}
fun main(args: Array<String>) {
    val resultado1 = operar(Operacion.SUMA, ...arreglo: 10, 20, 30)
    println("La suma es $resultado1")
}

```

### Problema 3

En cuanto al proyecto 172 se define una función llamada “cantidadMayores” que recibe una cantidad variable de edades y retorna cuántas son mayores o iguales a 18. Utiliza el modificador vararg para permitir múltiples argumentos y la función count para contar los que cumplen la condición. En main, se llama a la función con una lista de edades y se imprime el resultado.

```

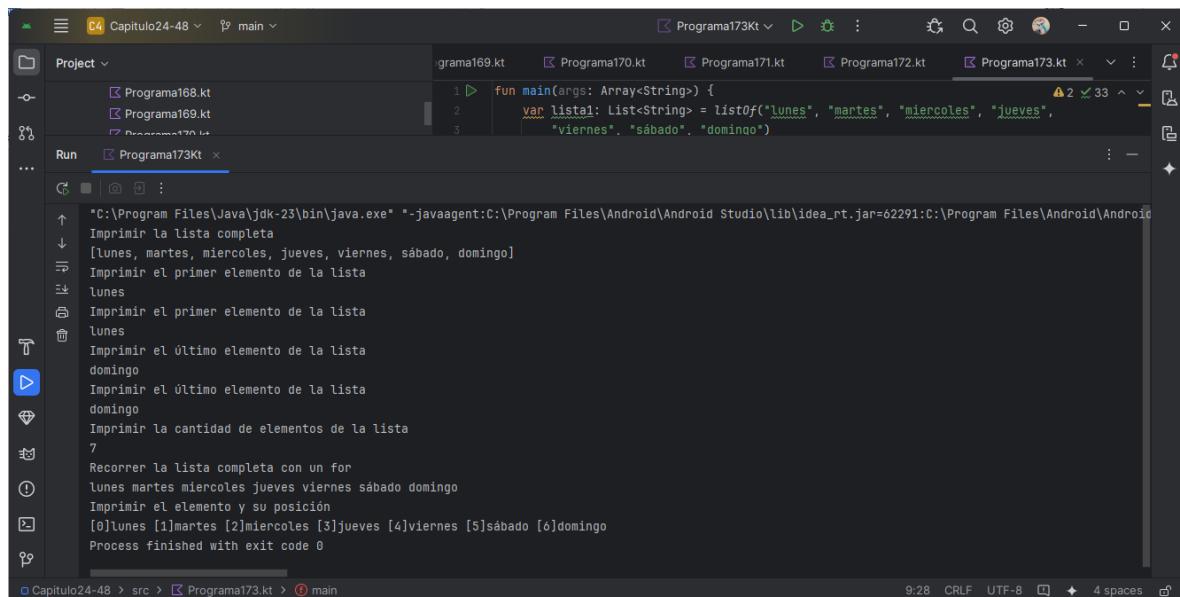
//PROBLEMA PROGRESO 172
/*Confeccionar una función que reciba una serie de edades y nos retorne la cantidad que son como mínimo se envía un entero a la función*/
fun cantidadMayores(vararg edades: Int) = edades.count {it >= 18}
fun main(args: Array<String>) = println(cantidadMayores(...edades: 3, 65, 32, 23, 2, 98, 23, 45))

```

## Capítulo 45

### Problema 1

En el proyecto 173 se crea una lista inmutable de tipo `List<String>` con los días de la semana utilizando la función `listOf`. Luego muestra cómo acceder y recorrer sus elementos: imprime la lista completa, el primer y último elemento usando tanto subíndices como funciones (`first()` y `last()`), y la cantidad total de elementos con `size`. También demuestra cómo recorrer todos los elementos con un `for` y cómo imprimir cada elemento junto a su posición usando los índices. Como la lista es inmutable, no se pueden modificar ni agregar elementos después de su creación.



The screenshot shows the Android Studio interface with the project 'Capítulo24-48' open. The 'Programa173Kt' file is selected in the run configuration dropdown. The code in the editor is:

```
fun main(args: Array<String>) {
    var lista: List<String> = listOf("lunes", "martes", "miércoles", "jueves",
        "viernes", "sábado", "domingo")
```

The terminal window shows the output of running the program:

```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62291:C:\Program Files\Android\Android Studio\bin" -Dfile.encoding=UTF-8
Imprimir la lista completa
[lunes, martes, miércoles, jueves, viernes, sábado, domingo]
Imprimir el primer elemento de la lista
lunes
Imprimir el primer elemento de la lista
lunes
Imprimir el último elemento de la lista
domingo
Imprimir el último elemento de la lista
domingo
Imprimir la cantidad de elementos de la lista
7
Recorrer la lista completa con un for
lunes martes miércoles jueves viernes sábado domingo
Imprimir el elemento y su posición
[0]lunes [1]martes [2]miércoles [3]jueves [4]viernes [5]sábado [6]domingo
Process finished with exit code 0
```

### Problema 2

En el proyecto 174 se crea una lista inmutable de cinco enteros utilizando la función `List`, que recibe como primer parámetro el tamaño de la lista y como segundo una función lambda que se ejecuta para generar cada elemento. En este caso, la lambda llama a la función cargar, que solicita al usuario un número entero y lo devuelve. Así, la lista se llena con cinco valores ingresados por el usuario y luego se imprime. Cada valor returnedo por la lambda es almacenado en la lista.

```
1 fun cargar(): Int {
2     print("Ingrese un entero:")
3     val valor = readLine()!!.toInt()
4     return valor
5 }
6 fun main(args: Array<String>) {
7     var lista1: List<Int> = List( size: 5, {cargar()})
8     println(lista1)
9 }
```

Ingrese un entero:1  
Ingrese un entero:2  
Ingrese un entero:7  
Ingrese un entero:4  
Ingrese un entero:59  
[1, 2, 7, 4, 59]  
Process finished with exit code 0

### Problema 3

En el proyecto 175 el programa trabaja con una lista mutable de edades creada con mutableListOf. Permite modificar elementos, acceder a posiciones específicas, calcular el promedio, agregar elementos al inicio o al final, y eliminar tanto por posición como por condición. Se cuenta cuántas edades son mayores o iguales a 18 y luego se eliminan todas las que valen 12. Finalmente, se borra toda la lista y se verifica si está vacía, mostrando un mensaje si no hay datos almacenados.

```
1 fun main(args: Array<String>) {
2     val edades: MutableList<Int> = mutableListOf(23, 67, 12, 35, 12)
3
4     // Lista de edades
5     [23, 67, 12, 35, 12]
6
7     // Lista completa después de modificar la primera edad
8     [60, 67, 12, 35, 12]
9
10    // Primera edad almacenada en la lista
11    60
12
13    // Promedio de edades
14    37.2
15
16    // Agregamos una edad al final de la lista:
17    Lista de edades
18    [60, 67, 12, 35, 12, 50]
19
20    // Agregamos una edad al principio de la lista:
21    Lista de edades
22    [17, 60, 67, 12, 35, 12, 50]
23
24    // Eliminamos la primera edad de la lista:
25    Lista de edades
26    [60, 67, 12, 35, 12, 50]
27
28    // Cantidad de personas mayores de edad:4
29    Lista de edades después de borrar las que tienen 12 años
30
```

```
Capítulo24-48 main
Programa171.kt Programa172.kt Programa173.kt Programa174.kt Programa175.kt
1 fun main(args: Array<String>) {
2     val edades: MutableList<Int> = mutableListOf(23, 67, 12, 35, 12)
3
4     println("37.2")
5     println("Agregamos una edad al final de la lista:")
6     Lista de edades
7     [60, 67, 12, 35, 12, 50]
8     println("Agregamos una edad al principio de la lista:")
9     Lista de edades
10    [17, 60, 67, 12, 35, 12, 50]
11    Eliminamos la primer edad de la lista:
12    Lista de edades
13    [60, 67, 12, 35, 12, 50]
14    Cantidad de personas mayores de edad:4
15    Lista de edades después de borrar las que tienen 12 años
16    [60, 67, 35, 50]
17    Lista de edades luego de borrar la lista en forma completa
18    []
19    No hay edades almacenadas en la lista
20
21 Process finished with exit code 0
```

## Problema 4

En el proyecto 176 se genera una lista mutable de 20 enteros aleatorios entre 1 y 6 utilizando MutableList y una lambda con Math.random(). Luego, cuenta cuántos elementos tienen el valor 1 con count, elimina todos los elementos con valor 6 usando removeAll, y muestra la lista original y la lista resultante después de la eliminación.

```
Capítulo24-48 main
Programa171.kt Programa172.kt Programa173.kt Programa174.kt Programa175.kt Programa176.kt
1 fun main(args: Array<String>) {
2     val lista: MutableList<Int> = MutableList(size: 20) { ((Math.random() * 6) + 1).toInt() }
3     println("Lista completa")
4     println(lista)
5     val cant = lista.count { it == 1 }
6     println("Cantidad de elementos con el valor 1: $cant")
7     lista.removeAll { it == 6 }
8     println("Lista luego de borrar los elementos con el valor 6")
9     println(lista)
10 }
```

## Problema 5

En el proyecto 177 se define una clase “Persona” con propiedades nombre y edad, y métodos para imprimir los datos y verificar si la persona es mayor de edad. En la

función main, se crea una lista mutable de tres personas, se imprime la información de cada una usando forEach, y se cuenta cuántas son mayores de edad con count.

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor with the file 'Programa177.kt' selected. The code defines a class 'Persona00' with a constructor taking name and age, and a method 'esMayoresEdad()' that prints a message if the age is less than 18. The 'main' function creates a mutable list of three 'Persona00' objects, prints their details using 'forEach', and counts how many are 18 or older using 'count'. The run tab shows the output: 'Listado de todas personas', followed by three person entries, and 'La cantidad de personas mayores de edad es 2'. The bottom status bar shows the time as 14:40 and other settings.

```
1  class Persona00 (var nombre: String, var edad: Int) {  
2      fun esMayoresEdad() {  
3          println("No es mayor de edad $nombre")  
4      }  
5  }  
6  
7  fun main(args: Array<String>) {  
8      val personas: MutableList<Persona00>  
9      personas = mutableListOf(Persona00("Juan", 22), Persona00("Ana", 19),  
10         Persona00("Marcos", 12))  
11     personas.forEach { it.imprimir() }  
12     val cant = personas.count { it.edad >= 18 }  
13     println("La cantidad de personas mayores de edad es $cant")  
14 }  
15  
16  
17  
18
```

## Problema 6

En el proyecto 178 el programa crea una lista inmutable de 100 elementos enteros con valores aleatorios entre 0 y 20. Luego, mediante el uso de un forEach y un bloque when, cuenta cuántos elementos están en los rangos de 1 a 4, 5 a 8 y 10 a 13. Finalmente, verifica si el número 20 está presente en la lista y lo imprime en consecuencia.

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor with the file 'Programa178.kt' selected. The code starts with a comment explaining the problem: creating an immutable list of 100 integers between 1 and 20, counting how many fall into ranges 1-4, 5-8, and 10-13, and checking if the number 20 is present. It uses a 'when' block inside a 'forEach' loop to count elements in each range. The run tab shows the output: a list of 100 random numbers, followed by 'Cantidad de valores comprendidos entre 1..4: 21', 'Cantidad de valores comprendidos entre 5..8: 10', 'Cantidad de valores comprendidos entre 10..13: 21', and 'La lista contiene el 20'. The bottom status bar shows the time as 9:18 and other settings.

```
1 //PROBLEMA PROPUESTO 178  
2 /*Crear una lista inmutable de 100 elementos enteros con valores aleatorios comprendidos entre 1 y 20.  
3 contar cuantos hay comprendidos entre 1 y 4, 5 y 8 y cuantos entre 10 y 13.  
4 Imprimir si el valor 20 està presente en la lista.*/  
5 fun main(args: Array<String>) {  
6     val lista1 = List( size: 100, { ((Math.random() * 21)).toInt() })  
7     println(lista1)  
8     var cant1 = 0  
9     var cant2 = 0  
10    var cant3 = 0  
11    lista1.forEach { when(it) {  
12        in 1 .. 4 -> cant1++  
13        in 5 .. 8 -> cant2++  
14        in 10 .. 13 -> cant3++  
15    } }  
16    if (lista1.contains(20)) {  
17        println("La lista contiene el 20")  
18    }  
19 }  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100
```

## Problema 7

En el proyecto 179 se define una clase Empleado0 con dos propiedades: nombre y sueldo. Se crea una lista mutable con tres instancias de empleados y se imprimen los datos de cada uno. Además, se define una función calcularGastos que calcula el gasto total de la empresa sumando los sueldos de todos los empleados en la lista. Después de calcular los gastos iniciales, se agrega un nuevo empleado con un sueldo de 3000 y se recalcula el gasto total en sueldos.

The screenshot shows the Android Studio interface with the project 'Capítulo24-48' open. The code editor displays 'Programa179.kt' with the following content:

```
15 val suma = empleados.sumByDouble { it.sueldo }
16 println("Total de gastos de la empresa:$suma")
17 }
18
19 fun main(args: Array<String>) {
20     val empleados: MutableList<Empleado0> = mutableListOf(Empleado0("Juan", 2000.0),
21                 Empleado0("Ana", 3500.0),
22                 Empleado0("Carlos", 1500.0))
23     empleados.forEach { it.imprimir() }
24     calcularGastos(empleados)
```

The run output window shows the execution results:

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62386:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8 Programa179Kt
Juan tiene un sueldo de 2000.0
Ana tiene un sueldo de 3500.0
Carlos tiene un sueldo de 1500.0
Total de gastos de la empresa:7000.0
Gastos luego de ingresar un nuevo empleado que cobra 3000
Total de gastos de la empresa:10000.0

Process finished with exit code 0
```

At the bottom, the status bar indicates the file path 'Capítulo24-48 > src > Programa179.kt' and the encoding '18:1 CRLF UTF-8'.

## Problema 8

En el proyecto 180 el programa permite cargar las alturas de 5 personas a través del teclado y almacenarlas en una lista inmutable de tipo Float. Luego, calcula el promedio de estas alturas utilizando la función average. Posteriormente, cuenta cuántas personas tienen una altura superior al promedio y cuántas tienen una altura inferior, usando las funciones count con las condiciones apropiadas. Finalmente, imprime el promedio de las alturas, la cantidad de personas más altas y la cantidad de personas más bajas que el promedio.

```
fun cargar(): Float {  
    ...  
}  
fun main(args: Array<String>) {  
    val lista1: List<Float> = List( size: 5) {cargar()}  
    val promedio = lista1.average()  
    println("La altura promedio de las personas es $promedio")  
    val altos = lista1.count { it > promedio}  
    val bajos = lista1.count { it < promedio}  
}  
*C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62396:C:\Program Files\Android\Android Studio\bin" Capítulo24-48  
Ingrese la altura de la persona (Ej. 1.92):1.70  
Ingrese la altura de la persona (Ej. 1.92):1.54  
Ingrese la altura de la persona (Ej. 1.92):1.60  
Ingrese la altura de la persona (Ej. 1.92):1.90  
Ingrese la altura de la persona (Ej. 1.92):1.80  
La altura promedio de las personas es 1.70799992370655  
La cantidad de personas más altas al promedio es: 2  
La cantidad de personas más bajas al promedio es: 3  
Process finished with exit code 0
```

## Capítulo 46

### Problema 1

En el proyecto 181 se crea un Map en Kotlin, donde las claves son nombres de países y los valores son sus respectivas poblaciones. Primero, imprime el mapa completo. Luego, recorre las entradas del mapa e imprime la clave y el valor de cada una. Después, muestra la cantidad de elementos que tiene el mapa. Se realiza la búsqueda de la población de Argentina mediante su clave, y si se encuentra, se imprime. Lo mismo ocurre para Brasil, pero si no se encuentra, imprime un mensaje indicando que no está presente. Finalmente, el programa calcula y muestra la suma de las poblaciones de todos los países almacenados en el mapa. También se muestra cómo crear un Map utilizando el operador to para definir las entradas de manera más concisa.

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is a tree view of files: Programa171.kt, Programa172.kt, Programa173.kt, Programa174.kt, Programa175.kt, Programa176.kt, Programa177.kt, Programa178.kt, Programa179.kt, Programa180.kt, and Programa181.kt. The main editor window displays the following Kotlin code:

```
1 fun main(args: Array<String>) {
2     val paises: Map<String, Int> = mapOf( Pair("argentina", 40000000),
3                                         Pair("espana", 46000000),
4                                         Pair("uruguay", 34000000))
5     println("Listado completo del Map")
6     println(paises)
7     for ((clave, valor) in paises)
8         println("Para la clave $clave tenemos almacenado $valor")
```

The run tab shows the output of the program:

```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62465:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8
Listado completo del Map
{argentina=40000000, españa=46000000, uruguay=34000000}
Para la clave argentina tenemos almacenado 40000000
Para la clave españa tenemos almacenado 46000000
Para la clave uruguay tenemos almacenado 34000000
La cantidad de elementos del mapa es 3
La cantidad de habitantes de argentina es 40000000
brasil no se encuentra cargado en el Map
Cantidad total de habitantes de todos los paises es 89400000
Process finished with exit code 0
```

## Problema 2

En el proyecto 182 se crea un Map de productos, donde las claves son los nombres de los productos (de tipo String) y los valores son los precios de esos productos (de tipo Float). El primer paso es definir el mapa con cinco productos y sus respectivos precios. La función imprimir recorre el mapa y para cada entrada imprime el nombre del producto y su precio. La función cantidadPrecioMayor20 cuenta cuántos productos tienen un precio superior a 20 utilizando el método count y una función lambda que evalúa si el valor del producto supera los 20. Finalmente, el programa imprime la cantidad de productos con precios superiores a 20.

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is a tree view of files: Programa171.kt, Programa172.kt, Programa173.kt, Programa174.kt, Programa175.kt, Programa176.kt, Programa177.kt, Programa178.kt, Programa179.kt, Programa180.kt, and Programa182.kt. The main editor window displays the following Kotlin code:

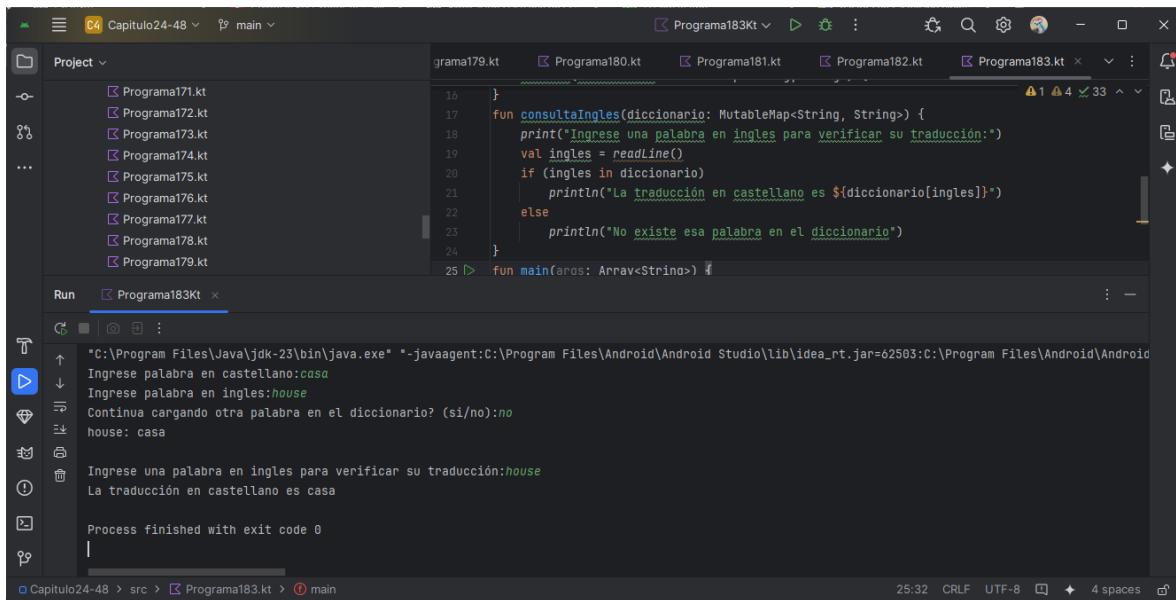
```
1 fun imprimir(productos: Map<String, Float>) {
2 }
3 fun cantidadPrecioMayor20(productos: Map<String, Float>) {
4     val cant = productos.count{ it.value > 20 }
5     println("Cantidad de productos con un precio superior a 20: $cant")
6 }
7
8 fun main(args: Array<String>) {
9     val productos: Map<String, Float> = mapOf("papas" to 12.5f,
10                                         "manzanas" to 26f,
11                                         "peras" to 31f,
12                                         "mandarinas" to 15f,
13                                         "pomelos" to 28.0f)
```

The run tab shows the output of the program:

```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62468:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8
papas tiene un precio 12.5
manzanas tiene un precio 26.0
peras tiene un precio 31.0
mandarinas tiene un precio 15.0
pomelos tiene un precio 28.0
Cantidad de productos con un precio superior a 20: 3
Process finished with exit code 0
```

## Problema 3

Por su parte, el proyecto 183 simula un diccionario interactivo. Permite al usuario ingresar pares de palabras en castellano e inglés, almacenándolas en un MutableMap. Luego, muestra todas las palabras registradas y permite consultar la traducción de una palabra en inglés. El proceso de carga y consulta continúa hasta que el usuario decide detenerse.



```
Capítulo24-48 main
Programa179.kt Programa180.kt Programa181.kt Programa182.kt Programa183.kt

16 }
17 fun consultaIngles(diccionario: MutableMap<String, String>) {
18     print("Ingrese una palabra en inglés para verificar su traducción:")
19     val ingles = readLine()
20     if (ingles in diccionario)
21         println("La traducción en castellano es ${diccionario[ingles]}")
22     else
23         println("No existe esa palabra en el diccionario")
24 }
25 fun main(args: Array<String>) {

Run Programa183Kt
↑ "C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62503:C:\Program Files\Android\Android
↓ Ingrese palabra en castellano:casa
Continua cargando otra palabra en el diccionario? (si/no):no
house: casa
Ingrese una palabra en inglés para verificar su traducción:house
La traducción en castellano es casa
Process finished with exit code 0
```

#### Problema 4

En cuanto al proyecto 184 se gestiona un inventario de productos utilizando un MutableMap donde cada producto se almacena con un código único. Se definen varias funciones: una para cargar productos en el mapa, otra para listar todos los productos con su información (nombre, precio, stock), una más para consultar un producto por su código y mostrar su detalle, y por último, una función que cuenta cuántos productos no tienen stock disponible. En el main, se crean los productos, se listan, se consulta un producto y se muestra la cantidad de productos sin stock.

```
Programa184Kt
```

```
Programa180.kt Programa181.kt Programa182.kt Programa183.kt Programa184.kt
```

```
Project
```

```
Programa171.kt Programa172.kt Programa173.kt Programa174.kt Programa175.kt Programa176.kt Programa177.kt Programa178.kt
```

```
Run Programa184Kt
```

```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62508:C:\Program Files\Android\Android Studio\lib\proguard.jar=62508:lib\proguard.jar" -Dfile.encoding=UTF-8
```

```
Listado completo de productos
```

```
Código: 1 Descripción Papas Precio: 13.15 Stock Actual: 200
```

```
Código: 15 Descripción Manzanas Precio: 20.0 Stock Actual: 0
```

```
Código: 20 Descripción Peras Precio: 3.5 Stock Actual: 0
```

```
Ingrese el código de un producto:
```

```
Nombre: Papas Precio: 13.15 Stock: 200
```

```
Cantidad de artículos sin stock: 2
```

```
Process finished with exit code 0
```

## Problema 5

En el proyecto 185 se gestiona un registro de alumnos y las materias que cursan, utilizando un MutableMap donde la clave es el DNI del alumno y el valor es una lista mutable de objetos de la clase Materia, que contiene el nombre de la materia y la nota. La función cargar solicita el número de alumnos y, para cada uno, carga su DNI y sus materias junto con las notas, hasta que el usuario decide dejar de ingresar más materias. Luego, esa lista de materias se asigna al DNI correspondiente en el mapa. La función imprimir recorre el mapa y muestra el DNI del alumno junto con todas las materias y sus notas. Finalmente, la función consultaPorDni permite consultar las materias y notas de un alumno ingresando su DNI.

```
Programa185Kt
```

```
Programa181.kt Programa182.kt Programa183.kt Programa184.kt Programa185.kt
```

```
Project
```

```
Programa171.kt Programa172.kt Programa173.kt
```

```
Run Programa185Kt
```

```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62519:C:\Program Files\Android\Android Studio\lib\proguard.jar=62519:lib\proguard.jar" -Dfile.encoding=UTF-8
```

```
Cuantos alumnos cargará ?:3
```

```
Ingrese dni:210849
```

```
Ingrese materia del alumno:ingles
```

```
Ingrese nota:89
```

```
Ingrese otra nota (si/no):no
```

```
Ingrese dni:210850
```

```
Ingrese materia del alumno:matemáticas
```

```
Ingrese nota:90
```

```
Ingrese otra nota (si/no):no
```

```
Ingrese dni:210851
```

```
Ingrese materia del alumno:español
```

```
Ingrese nota:100
```

```
Ingrese otra nota (si/no):no
```

```
Dni del alumno: 210849
```

```
Materia: inglés
```

```
Nota: 89
```

```
Ingrese otra nota (si/no):no
Dni del alumno: 210849
Materia: ingles
Nota: 89

Dni del alumno: 210850
Materia: matematicas
Nota: 90

Dni del alumno: 210851
Materia: espanol
Nota: 100

Ingrese el dni del alumno a consultar:210850
Cursa las siguientes materias
Nombre de materia: matematicas
Nota: 90

Process finished with exit code 0
```

## Problema 6

En el proyecto 186 se crea una agenda utilizando un `MutableMap`, donde la clave es una instancia de la clase `Fecha` (que contiene día, mes y año) y el valor es un String con las actividades programadas para esa fecha. En la función cargar, se permite al usuario ingresar varias fechas junto con las actividades asociadas, y estas se almacenan en el mapa. La función imprimir recorre el mapa y muestra todas las fechas registradas junto con sus actividades correspondientes. Finalmente, la función consultaFecha permite consultar las actividades de una fecha específica ingresada por el usuario, y si esa fecha está registrada en el mapa, muestra las actividades; en caso contrario, informa que no existen actividades para esa fecha.

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62535:C:\Program Files\Android\Android
Ingrese fecha
Ingrese el dia:22
Ingrese el mes:02
Ingrese el año:2025
Ingrese todas las actividades para ese dia:Clases de baile
Ingrese otra fecha (si/no):no
Fecha 22/2/2025
Actividades: Clases de baile

Ingrese una fecha a consultar
Ingrese el dia:22
Ingrese el mes:02
Ingrese el año:2025
Actividades: Clases de baile

Process finished with exit code 0
```

## Capítulo 47

### Problema 1

En el proyecto 187 se trabaja con un conjunto mutable (MutableSet) que almacena enteros. Se comienza creando un conjunto con los valores 2, 7, 20, 150 y 3. Luego, se muestra el listado completo de los elementos y se imprime la cantidad de elementos del conjunto. A continuación, se agrega el valor 500 al conjunto y se vuelve a imprimir el conjunto con este nuevo valor, mostrando que el conjunto no acepta elementos duplicados, ya que al intentar agregar nuevamente el 500, no se repite. El programa verifica si el número 500 está presente en el conjunto utilizando el operador in y muestra un mensaje correspondiente. Después, el valor 500 es eliminado del conjunto, y se confirma que ya no está presente. Finalmente, se cuenta cuántos elementos del conjunto tienen un valor mayor o igual a 10 y se imprime el resultado.

The screenshot shows the Android Studio interface with the project 'Capítulo24-48' open. The code editor displays 'Programa187.kt' with the following content:

```
1 fun main(args: Array<String>) {
2     val conjunto1: MutableSet<Int> = mutableSetOf(2, 7, 20, 150, 3)
3     println("Listado completo del conjunto")
4     for(elemento in conjunto1)
5         print("$elemento ")
6     println()
7     println("Cantidad de elementos del conjunto: ${conjunto1.size}")
```

The 'Run' tab is selected, showing the command: "C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea\_rt.jar=62553:C:\Program Files\Android\Android Studio\lib". The run output pane shows the following sequence of events:

- Listado completo del conjunto
- 2 7 20 150 3
- Cantidad de elementos del conjunto: 5
- Listado completo del conjunto luego de agregar el 500
- 2 7 20 150 3 500
- Listado completo del conjunto luego de volver a agregar el 500
- 2 7 20 150 3 500
- El 500 está almacenado en el conjunto
- Eliminamos el elemento 500 del conjunto
- El 500 no está almacenado en el conjunto
- Cantidad de elementos con valores superiores o igual a 10: 2

At the bottom, it says "Process finished with exit code 0".

### Problema 2

El proyecto 188 tiene como objetivo verificar si una fecha ingresada por el usuario es un feriado, comparándola con un conjunto de fechas previamente definidas. Primero, se declara una clase de datos llamada Fecha que contiene tres propiedades: dia, mes y año. Luego, se crea un conjunto inmutable llamado feriados, que contiene dos fechas específicas, el 1 de enero de 2017 y el 25 de diciembre de 2017. A continuación, el programa solicita al usuario que ingrese una fecha, pidiendo el día, el mes y el año. Despues, se utiliza el operador in para comprobar si la fecha ingresada por el usuario está presente en el conjunto feriados. Si la fecha se encuentra en el conjunto, el programa imprime "La fecha ingresada es feriado"; de lo contrario, muestra "La fecha ingresada no es feriado".

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor with a file named Programa188.kt. The code defines a data class Fechaa and a main function that prints a message and asks for user input to check if a specific date is a holiday.

```
data class Fechaa(val dia: Int, val mes: Int, val año: Int)
fun main(args: Array<String>) {
    var feriados: Set<Fechaa> = setOf(Fechaa(dia: 1, mes: 1, año: 2017),
                                         Fechaa(dia: 25, mes: 12, año: 2017))
    println("Ingrese una fecha")
    print("Ingrese el dia:")
    val dia = readLine()!!.toInt()
    print("Ingrese el mes:")
    val mes = readLine()!!.toInt()
    print("Ingrese el año:")
    val año = readLine()!!.toInt()
    if (Fechaa(dia, mes, año) in feriados)
```

The run tab shows the output of the program:

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62556:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8 Programa188Kt
Ingresé una fecha
Ingresé el dia:14
Ingresé el mes:03
Ingresé el año:2024
La fecha ingresada no es feriado
Process finished with exit code 0
```

### Problema 3

En el proyecto 189 se simula un juego de lotería donde se genera un cartón con 6 números aleatorios entre 1 y 50. Luego, se sortean números aleatorios de un bolillero hasta que todos los números del cartón sean sorteados. El programa cuenta cuántos sorteos fueron necesarios para completar el cartón y muestra el resultado. Utiliza conjuntos mutables (MutableSet) para almacenar los números sin duplicados.

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the code editor with a file named Programa189.kt. The code defines a function verificarTriunfo and a main function that generates a lottery ticket and draws numbers until it matches.

```
fun verificarTriunfo(carton: MutableSet<Int>, bolillero: MutableSet<Int>) {
    ...
}
fun main(args: Array<String>) {
    val carton: MutableSet<Int> = mutableSetOf()
    generarCarton(carton)
    println("Cartón de lotería generado")
    println(carton)
    val bolillero: MutableSet<Int> = mutableSetOf()
    ...
}
```

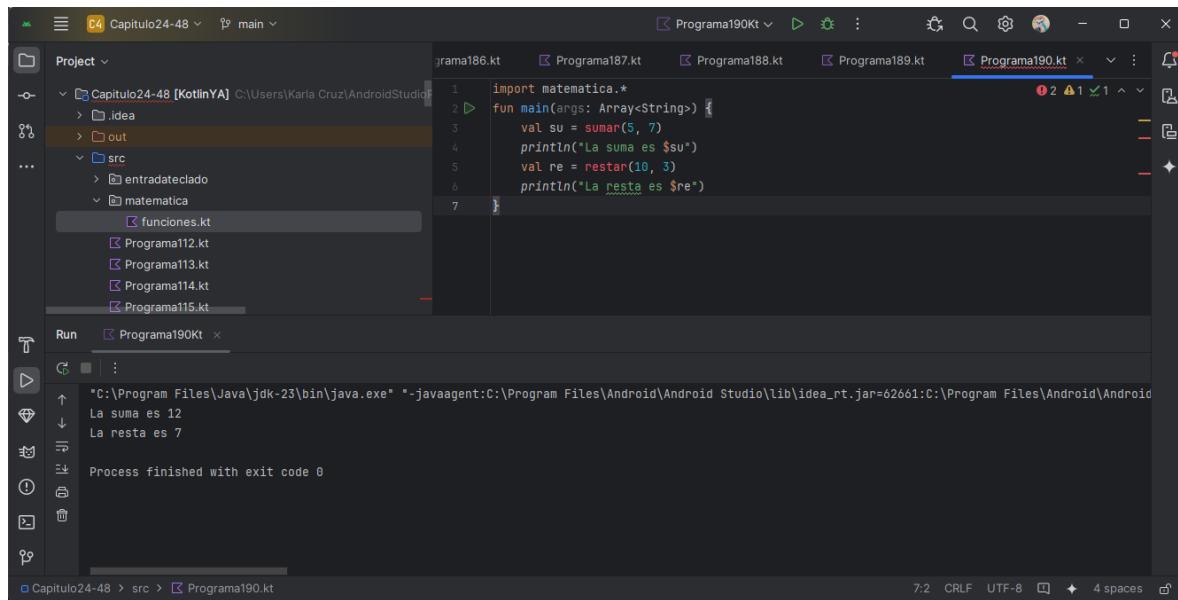
The run tab shows the output of the program:

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62562:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8 Programa189Kt
Cartón de lotería generado
[7, 18, 15, 45, 34, 21]
Números del bolillero
[18, 24, 31, 2, 44, 35, 16, 34, 50, 22, 30, 10, 42, 14, 33, 41, 17, 4, 20, 25, 13, 21, 36, 39, 28, 49, 26, 47, 5, 46, 29, 27, 3, 48, 32, 19, 40, 43, 8
Se sacaron 48 bolillas hasta que el cartón ganó.
Process finished with exit code 0
```

## Capítulo 48

### Problema 1

En el proyecto 190 se muestra cómo utilizar funciones definidas en un paquete en Kotlin. En el archivo funciones.kt, se crea un paquete llamado matematica, donde se definen dos funciones: sumar, que recibe dos enteros y devuelve su suma, y restar, que realiza una resta entre dos enteros. En el archivo Programa190.kt, se importa el paquete matematica mediante la instrucción import matematica.\*, lo que permite acceder a todas las funciones definidas en él. Luego, en la función main, se llaman las funciones sumar y restar para realizar operaciones con valores específicos y se imprimen los resultados.



The screenshot shows the Android Studio interface with the project 'Capítulo24-48' open. The 'src' directory contains several files: Programa186.kt, Programa187.kt, Programa188.kt, Programa189.kt, and Programa190.kt. The 'matematica' package is defined under 'src'. Inside 'matematica', there is a file named 'funciones.kt' which contains the definitions for 'sumar' and 'restar'. The 'Programa190.kt' file imports 'matematica.\*' and uses it in its main function to perform calculations and print results. The 'Run' tab shows the output of the program being run, displaying 'La suma es 12' and 'La resta es 7'.

```
import matematica.*  
fun main(args: Array<String>) {  
    val su = sumar(5, 7)  
    println("La suma es $su")  
    val re = restar(10, 3)  
    println("La resta es $re")  
}
```

## Capítulo 48

### Problema 1

En el proyecto 191 se crea un paquete llamado entradateclado que contiene funciones para leer diferentes tipos de datos desde la entrada del usuario. Dentro de este paquete, en el archivo entrada.kt, se definen tres funciones: retornarInt, retornarDouble y retornarFloat. Cada una de estas funciones recibe un mensaje como parámetro, imprime ese mensaje en la consola y luego lee la entrada del usuario, convirtiéndola al tipo correspondiente (entero, doble o flotante). En el archivo principal, se importa el paquete entradateclado mediante la instrucción import entradateclado.\* y se utilizan las funciones definidas en el paquete para leer dos números enteros desde la entrada del usuario. Estos números se suman y se imprime el resultado en la consola.

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "Capítulo24-48". It contains a ".idea" folder, an "out" folder, a "src" folder with packages "entradaclado" and "matematica", and files "Programa112.kt", "Programa113.kt", "Programa114.kt", and "Programa115.kt".
- Code Editor:** The main editor window displays the file "Programa191.kt" with the following code:

```
//PROBLEMA PROPUESTO PROGRAMA 191
import entradateclado.*

fun main(args: Array<String>) {
    val numero1 = retornarInt("Ingrese primer valor:")
    val numero2 = retornarInt("Ingrese segundo valor")
    println("La suma es ${numero1 + numero2}")
}
```
- Run Tab:** The "Run" tab is selected, showing the output of the program:

```
*C:\Program Files\Java\jdk-23\bin\java.exe* "-javaagent:C:\Program Files\Android\Android Studio\lib\idea_rt.jar=62753:C:\Program Files\Android\Android Studio\lib\proguard.jar" -Dfile.encoding=UTF-8 Programa191Kt
Ingresar primer valor:10
Ingresar segundo valor20
La suma es 30
Process finished with exit code 0
```
- Status Bar:** The status bar at the bottom shows "Capítulo24-48 > src > Programa191.kt" and various file statistics like "8:2 CRLF", "UTF-8", and "4 spaces".

**Nota:** Los códigos con comentarios se encuentran subidos en [github](#).