

# Introducción a Kotlin

---

## Kotlin

---

### Competencias:

- Aprender qué es Kotlin y su relación con JetBrains

### Un Lenguaje llamado Kotlin

Muchos desarrolladores de software piensan que Kotlin es un lenguaje muy nuevo, esto debido a que se popularizó el año 2017 cuando fue anunciado por Google como un lenguaje más para desarrollar aplicaciones Android, junto con Java y C++.

Pero Kotlin no nace el 2017, este lenguaje fue anunciado el año 2011 por los desarrolladores de JetBrains como un proyecto que corre sobre la máquina virtual de Java y que puede interoperar con código Java.

El proyecto empezó el año 2010 debido a una necesidad del equipo de IntelliJ IDEA, este IDE está desarrollado en Java, por lo que necesitaban algo mejor que Java y que fuera igual de rápido, existía un posible candidato SCALA, pero se descartó debido a que era mucho para lo que ellos necesitaban, por lo que al no existir nada que solucionara sus problemas con Java, crearon su propio lenguaje de programación.

Hoy en día Kotlin es un lenguaje que permite trabajar sobre la JVM, Java Script y Kotlin Native.

Hoy ya Kotlin es uno de los lenguajes más queridos por los desarrolladores según StackOverflow y Google ya anunció este año a Kotlin-first como el lenguaje más utilizado para desarrollar aplicaciones en Android.

# Instalación

---

## Competencias:

- Aprender a instalar Kotlin

## Introducción

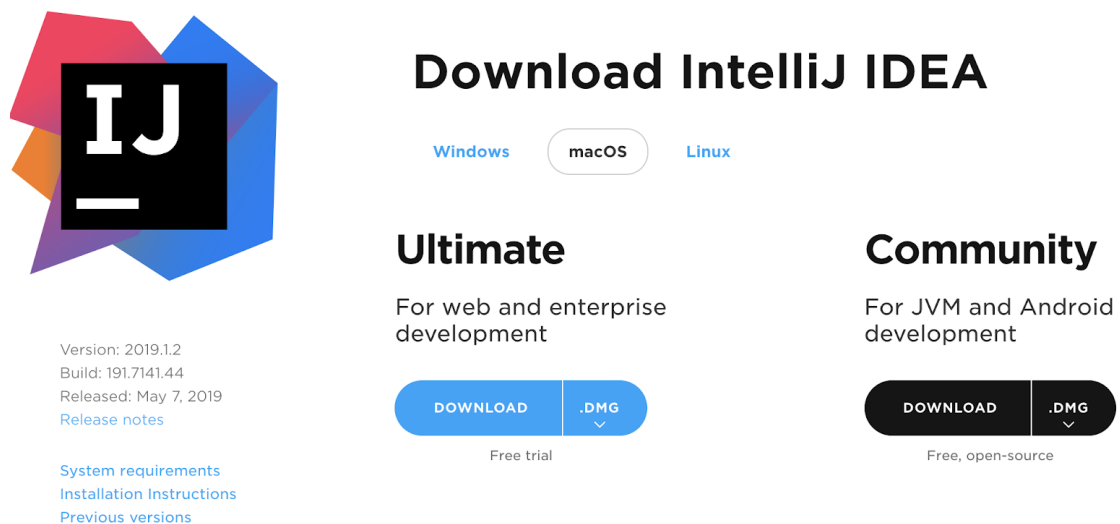
Es muy importante recordar que Kotlin trabaja sobre la máquina virtual de Java, por lo que es un requisito que en nuestro computador tengamos instalado el JDK de Java.

Requisitos para Instalar IntelliJ

## Instalando IntelliJ IDEA

Existen 2 versiones de IntelliJ que podemos utilizar, esta la versión “Community” que es gratuita y la versión “Ultimate” que es pagada para las empresas y se puede obtener una licencia para estudiantes o profesores.

### Para Mac



**Download IntelliJ IDEA**

Windows macOS Linux

**Ultimate**  
For web and enterprise development  
Free trial  
DOWNLOAD .DMG

**Community**  
For JVM and Android development  
Free, open-source  
DOWNLOAD .DMG

Version: 2019.1.2  
Build: 191.7141.44  
Released: May 7, 2019  
[Release notes](#)  
[System requirements](#)  
[Installation Instructions](#)  
[Previous versions](#)

Imagen 1. Descargar IntelliJ IDEA para Mac.

## Instalando en Mac

- **Paso 1:** Arrastramos el icono de “IntelliJ IDEA CE” hacia la carpeta de Applications.



Imagen 2. Instalar IntelliJ IDEA en Mac - Paso 1.

- **Paso 2:** Abrimos la aplicación.

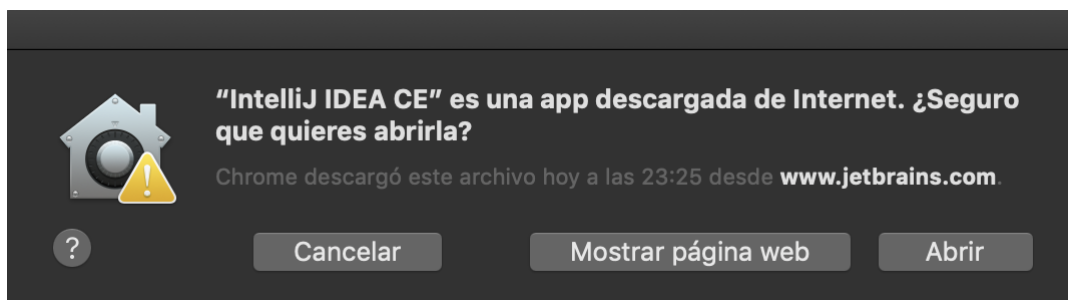


Imagen 3. Instalar IntelliJ IDEA en Mac - Paso 2.

- **Paso 3:** Presionamos el botón “OK” por defecto.

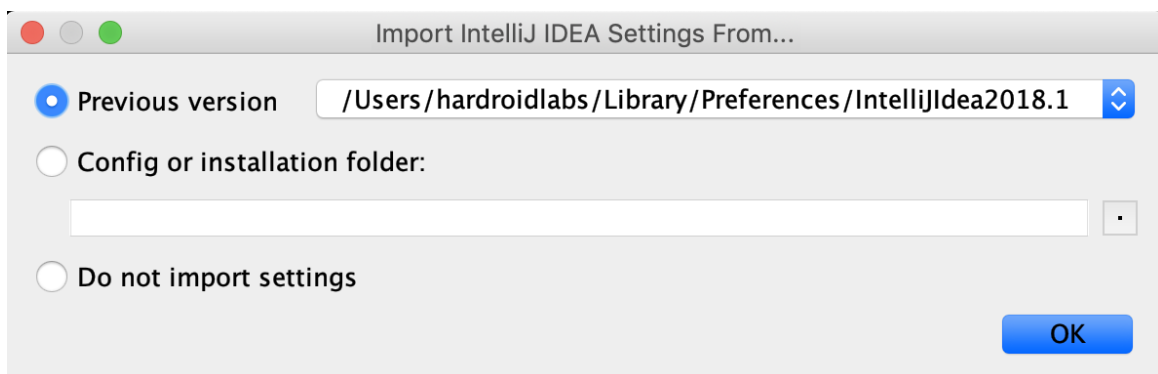


Imagen 4. Instalar IntelliJ IDEA en Mac - Paso 3.

- **Paso 4:** Presionamos el botón “Create New Project” y listo.

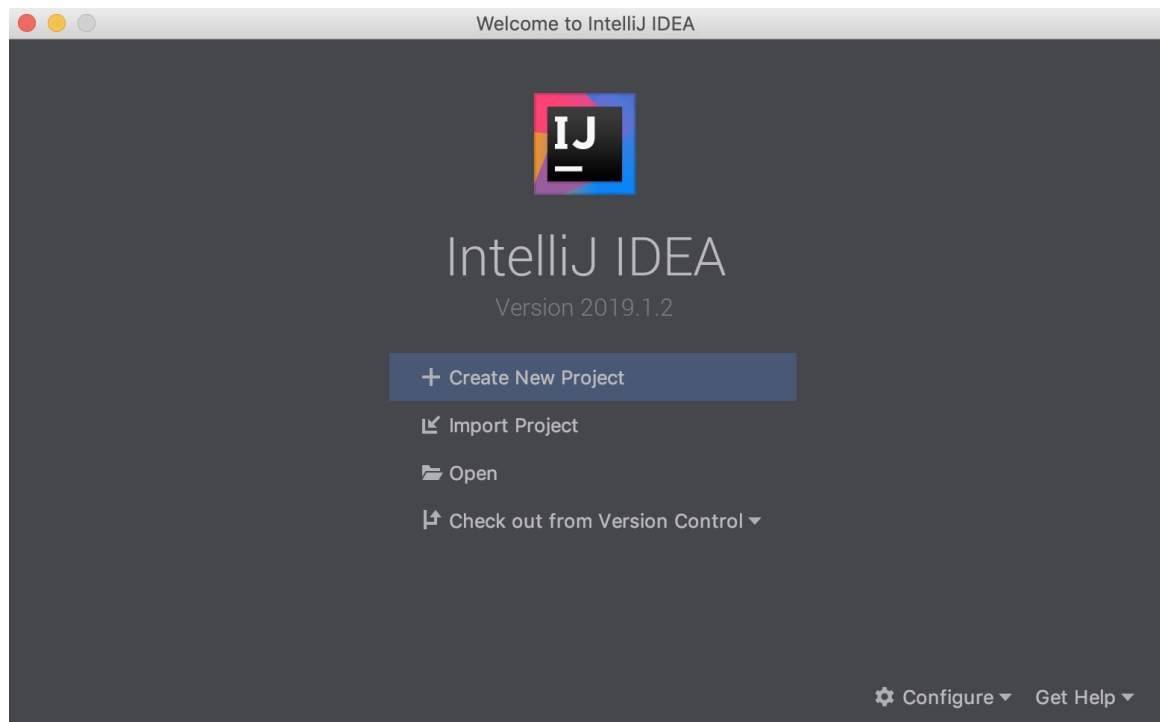


Imagen 5. Instalar IntelliJ IDEA en Mac - Paso 4.

## Para Windows

### Community

For JVM and Android development

DOWNLOAD

.EXE

Free, open-source

Imagen 6. Descargar IntelliJ IDEA para Windows.

## Instalando en Windows

- **Paso 1:** Abrimos el instalador de IntelliJ y presionamos el botón “next”.

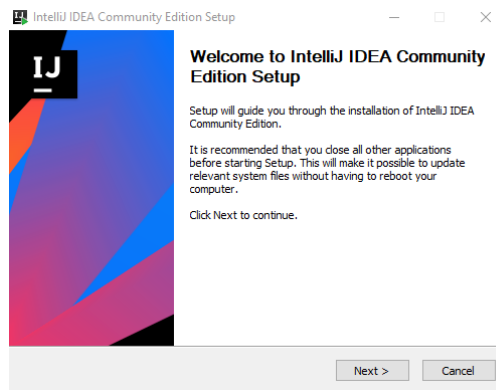


Imagen 7. Instalar IntelliJ IDEA en Windows - Paso 1.

- **Paso 2:** Seleccionamos la carpeta donde se instalará nuestro IDE, recomendando dejar la ruta por defecto y presionamos el botón “Next >”.

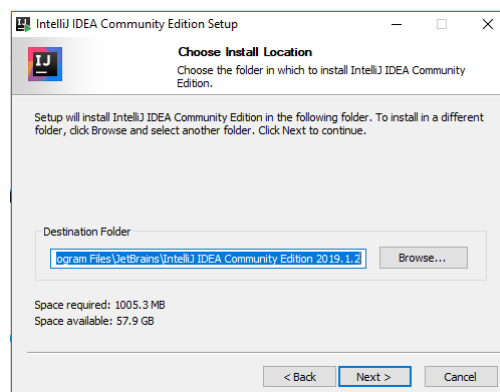


Imagen 8. Instalar IntelliJ IDEA en Windows - Paso 2.

- **Paso 3:** Seleccionamos a criterio las opciones de instalación, recomiendo seleccionar los archivos “.java”, “.groovy” y “.kt” y presionamos el botón “Next >”.

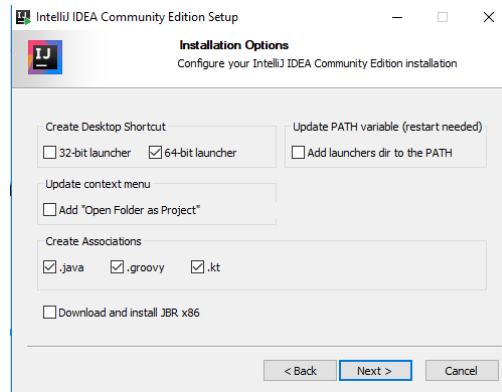


Imagen 9. Instalar IntelliJ IDEA en Windows - Paso 3.

- **Paso 4:** Presionamos el botón “Install”.

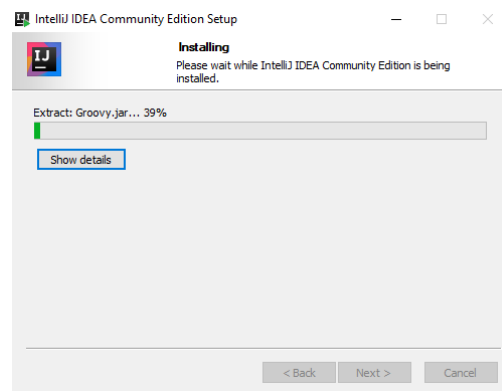


Imagen 10. Instalar IntelliJ IDEA en Windows - Paso 4.

- **Paso 5:** Presionamos finalizar, podemos seleccionar la opción “Run IntelliJ” y nuestro IDE se abrirá.

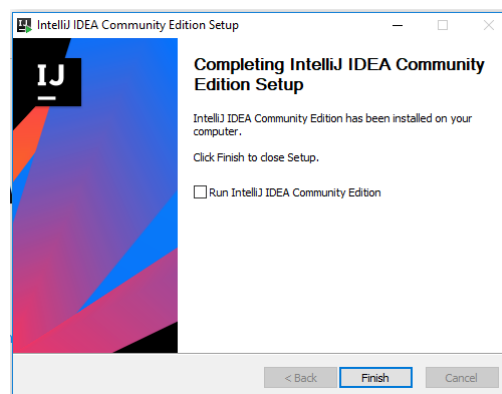


Imagen 11. Instalar IntelliJ IDEA en Windows - Paso 5.

- **Paso 6:** Seteamos las configuraciones presionando el botón “OK”.

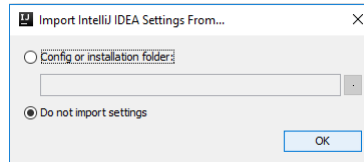


Imagen 12. Instalar IntelliJ IDEA en Windows - Paso 6.

- **Paso 7:** Aceptamos las condiciones de privacidad de IntelliJ y presionamos el botón “Continue”.

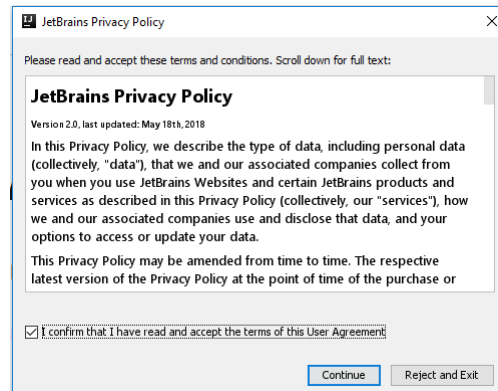


Imagen 13. Instalar IntelliJ IDEA en Windows - Paso 7.

- **Paso 8:** Seleccionamos el tema que más te guste, en mi caso, “Darcula”.

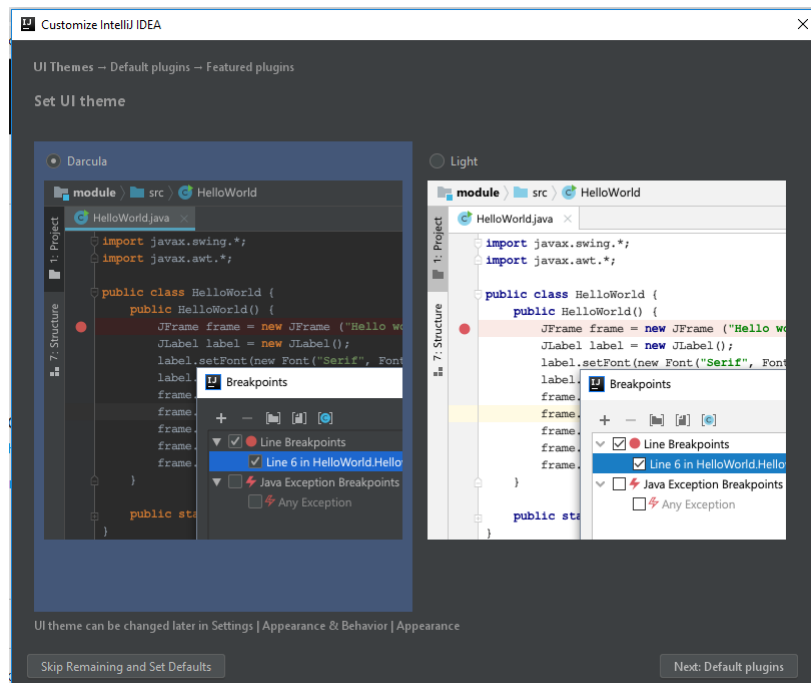


Imagen 14. Instalar IntelliJ IDEA en Windows - Paso 8.

## Ejercicio: “Hola Mundo en Kotlin”

Ahora que tenemos instalado nuestro IntelliJ IDEA, ya podemos crear nuestro primer proyecto en Kotlin, vamos a hacer un “Hola Mundo!”

- **Paso 1:** Abrimos nuestro IDE y presionamos el botón “Create New Project” .

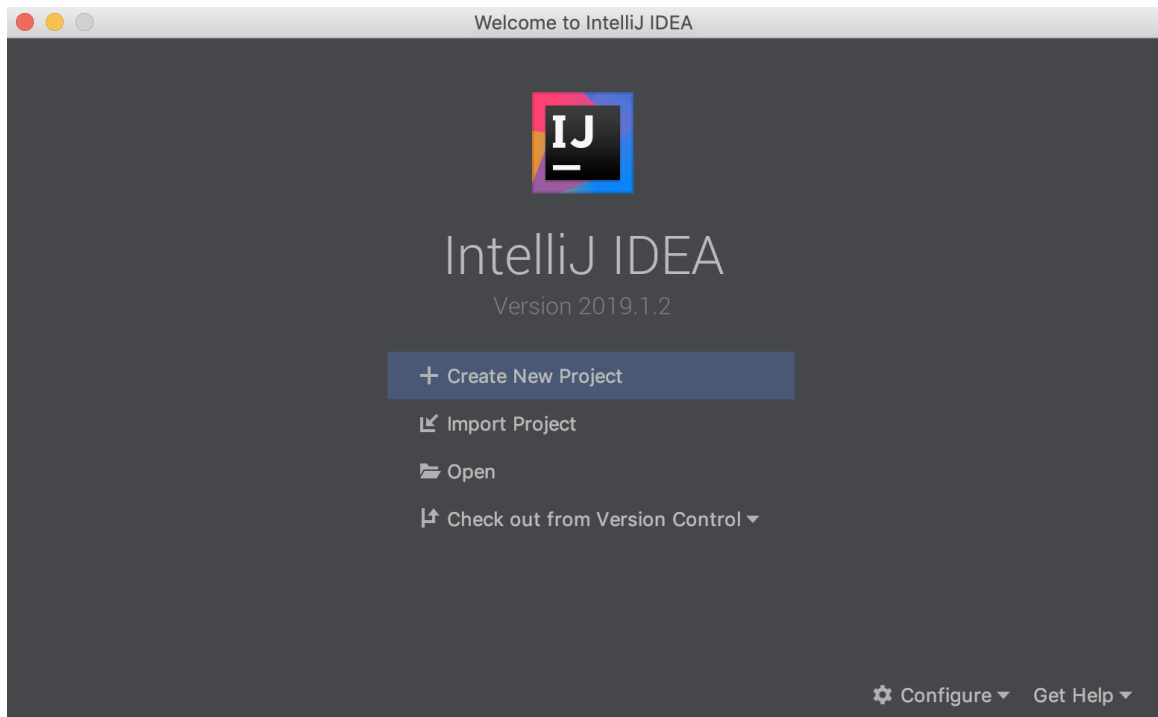


Imagen 15. Crear nuevo proyecto

- **Paso 2:** Hacemos click en: Java -> Kotlin ->JVM y presionamos el botón “Next”.

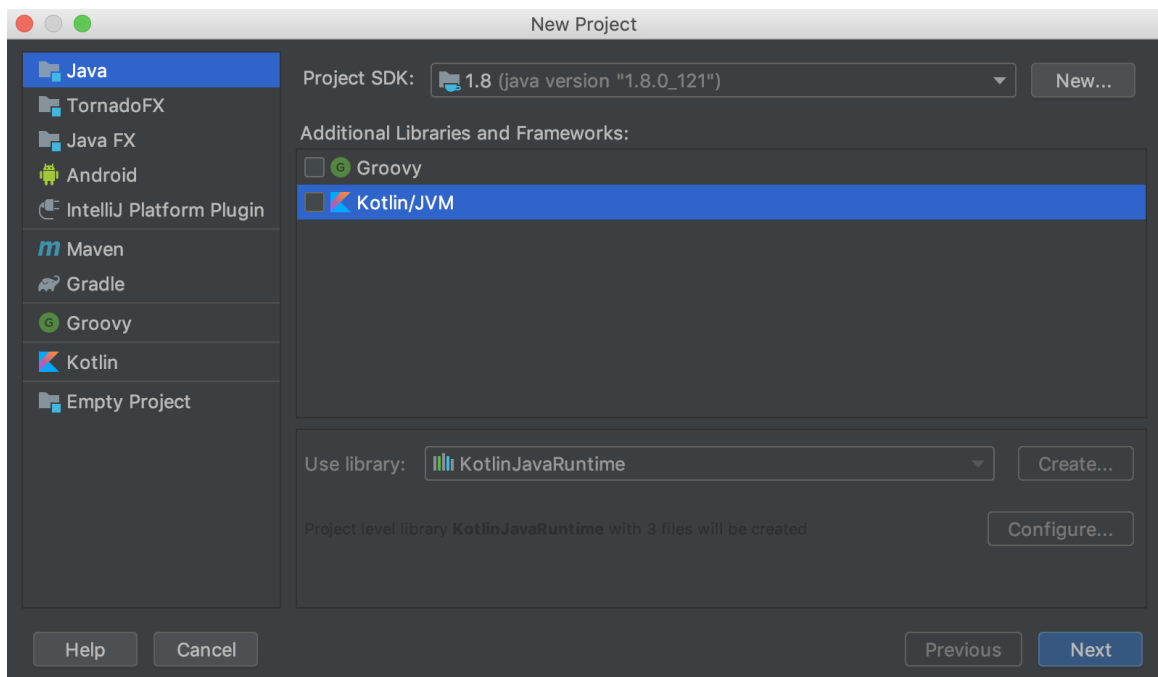


Imagen 16. Seleccionar librerías y frameworks



- **Paso 3:** Presionamos el botón “Next” .

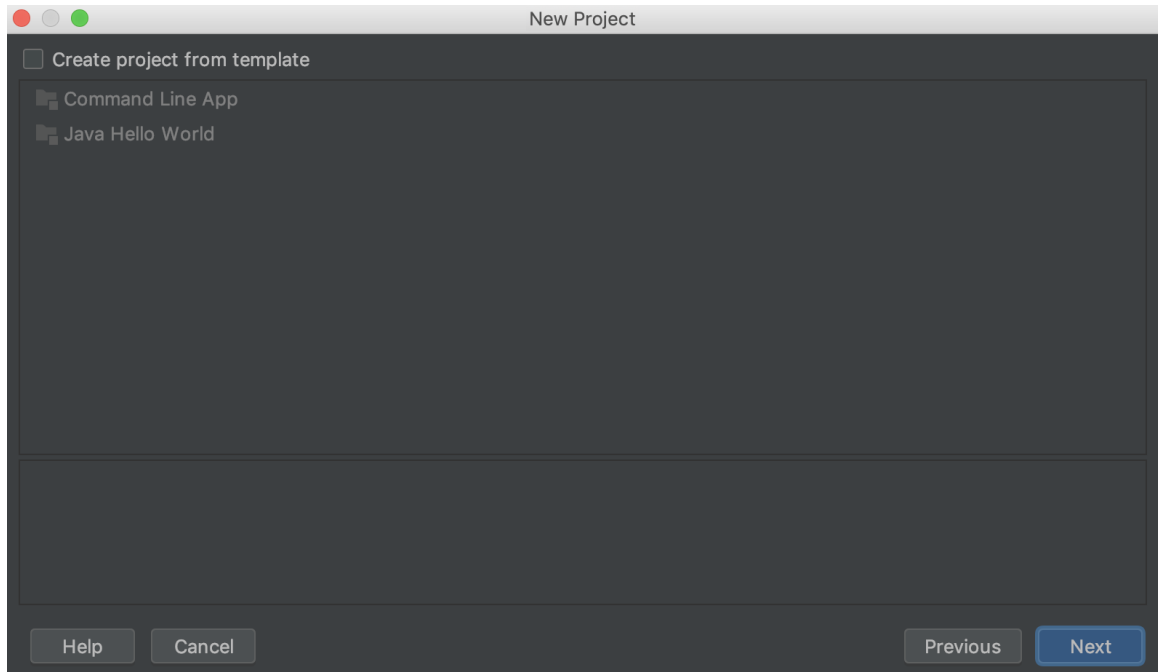


Imagen 17. Crear proyecto desde plantilla

- **Paso 4:** Agregamos el nombre del proyecto, nombre del módulo y presionamos el botón “Finish”.

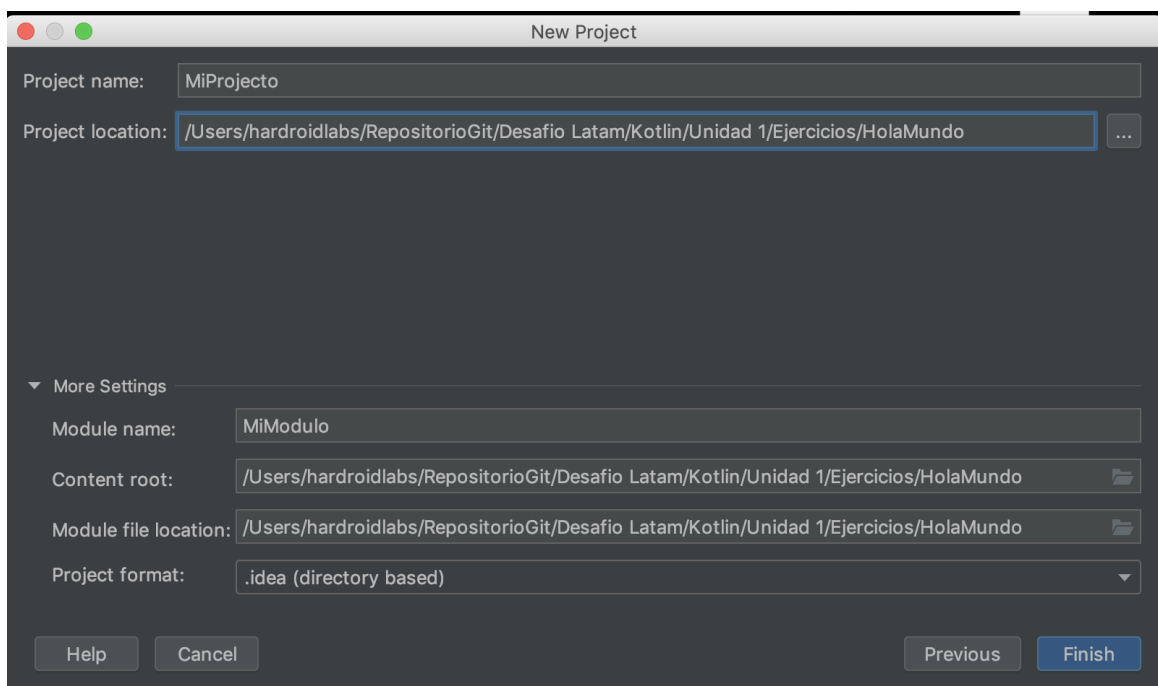


Imagen 18. Agregar nombre al proyecto y módulo

**Project name:** Este será el nombre de nuestro proyecto, se creará una carpeta con este nombre.

**Project location:** Esta es la ruta donde se crea nuestro proyecto.

**Module name:** Este corresponde al nombre del módulo, un proyecto Kotlin puede contener muchos módulos, por lo que en este caso se crea un proyecto con un módulo llamado “MiModulo”

- **Paso 5:** Botón derecho en la carpeta “src” y seleccionamos “New/Kotlin File/Class”.

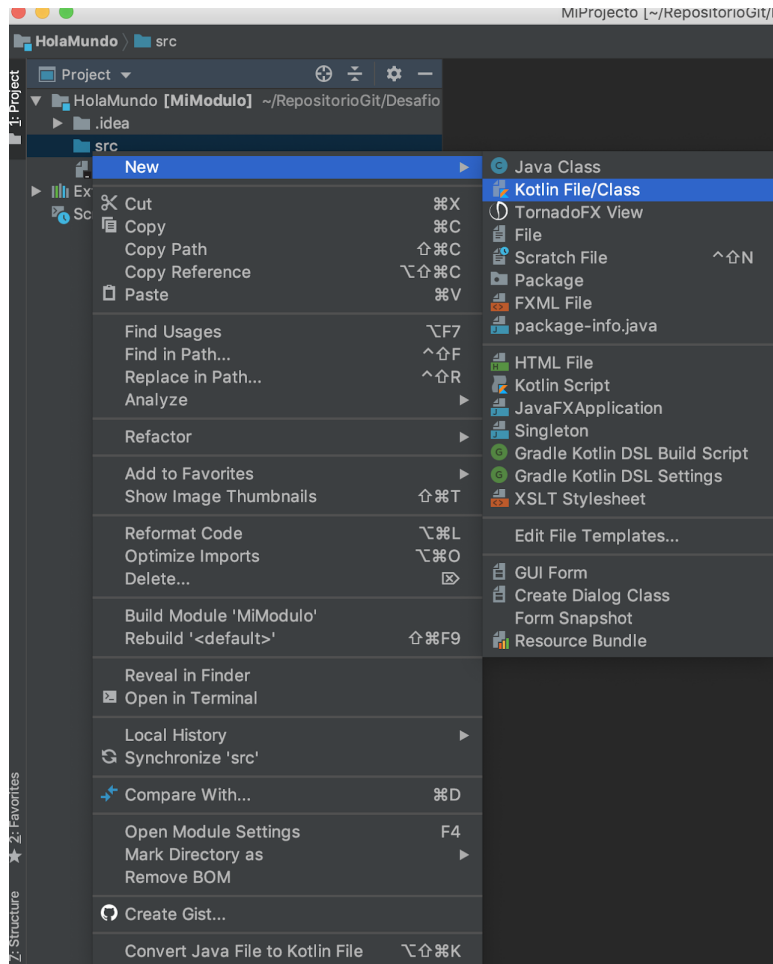


Imagen 19. Agregar nuevo archivo/clase Kotlin

- **Paso 6:** Escribimos el nombre del archivo Kotlin y presionamos el botón “OK”.

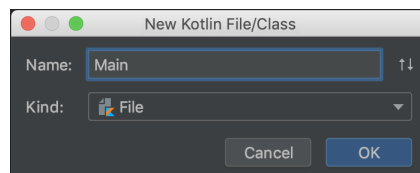


Imagen 20. Agregar nombre al nuevo archivo/clase

- **Paso 7:** Abrimos el archivo “Main.kt” y escribimos la siguiente función “main”. Esta función es equivalente al método main de Java “public static void main”.

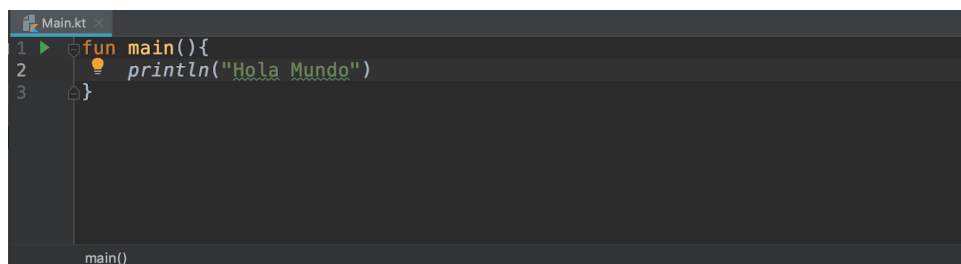


Imagen 21. Archivo main.kt

- **Paso 8:** Ejecutamos el método “main” haciendo clic en el botón Play o ejecutamos la acción que se indica en nuestro IDE.

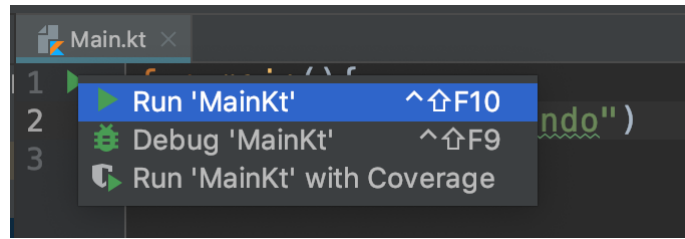


Imagen 22. Ejecutar acción

- **Paso 9:** Vemos el saludo en la consola de nuestro IDE.



Imagen 23. Ejecutar código

# Variables

---

## Competencias:

- Conocer las variables en Kotlin
- Declarar variables en Kotlin
- Declarar variables por inferencia en Kotlin
- Aprender formato de caracteres en Kotlin

## Introducción

Todo el código necesita de variables, en este capítulo aprenderemos cómo funcionan las variables en Kotlin y como se deben declarar para ser utilizadas, aprenderemos sobre el uso de variables por inferencia en Kotlin y cómo usar una variable mutable o inmutable.

## Variables

Cuándo pienses en variables imagínate que estas son vasos, existen distintos tipos de vasos de distinta forma y distinto color, pero todas tienen algo en común, un vaso contiene algo.

## Declarar variables en Kotlin

Para poder declarar una variable en Kotlin, necesitamos crear antes una función, para los ejemplos utilizaremos la función “**main**”, que profundizaremos en el capítulo de funciones. Además de crear una función, necesitamos saber 3 cosas:

1. Tipo de variable (“var” o “val”)
2. Nombre de la variable
3. Tipo de valor o clase (Numérico, Cadena de Caracteres, Lógico, etc.)

```
fun main() {  
    val numero : Int  
    var variable : TipoVariable  
}
```

## Variables por inferencia en Kotlin

En Kotlin podemos omitir la declaración del tipo de valor, y esto lo hacemos inicializando la variable con un valor determinado.

```
fun main() {  
    val nombre = "Goro Daimon"  
    val edad = 22  
}
```

Acá podemos ver como omitimos el tipo de valor en la declaración de la variable “*nombre*” y “*edad*”.

En Kotlin al asignarle un valor a la variable, el compilador determina automáticamente el tipo de valor que estas variables tendrán, para este ejemplo la variable “*nombre*” es String y la variable “*edad*” es Int.

## Diferencias entre un val y var

En Kotlin existen dos tipos de variables, las variables mutables y las variables inmutables.

**Las variables inmutables** se declaran como “*val*” y contienen un valor que “**No**” puede ser modificado. **Las variables mutables** se declaran como “*var*” y estas variables “**Si**” pueden ser modificadas.

```
fun main() {  
    val numeroFijo = 10  
    var numeroVariable = 10  
    numeroFijo = 12  
}
```

En este código podemos ver la declaración de dos variables, estas dos variables son de tipo entero (Int), pero solo la variable “numeroVariable” puede ser modificado su valor.

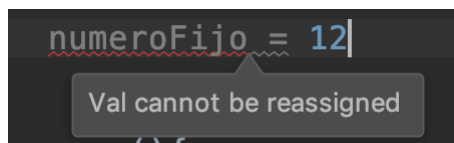


Imagen 24. Declaración de variables

En esta imagen podemos ver el error que nos da nuestro IDE al tratar de modificar una variable declarada como “val”.

## String Template en Kotlin

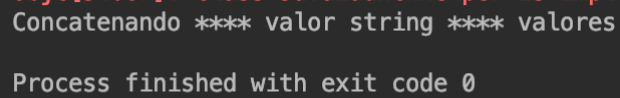
Una de las mejoras importantes que trae Kotlin como lenguaje de programación, es la facilidad de trabajar con variables String, a esto se les conoce como **"String Template"**.

Cuando estemos utilizando una variable String podemos utilizar el signo "\$" para concatenar valores.

```
fun main() {  
    val x = "valor string"  
    val template = "Concatenando **** $x **** valores"  
    println(template)  
}
```

Imagen 25. Código para concatenar valores

Esto nos facilita mucho el concatenar distintos valores String, dejando un código fácil de entender.

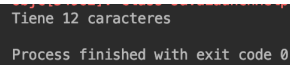


```
Concatenando **** valor string **** valores  
Process finished with exit code 0
```

Imagen 26. Resultado de concatenar valores

También podemos llamar a una función del lenguaje o de una variable utilizando el signo "\$" seguido del uso de llaves "{ }". Dentro de estas llaves podemos hacer la llamada a la función que necesitemos.

```
fun main() {  
    val x = "valor string"  
    val template = "Tiene ${x.count()} caracteres"  
    println(template)  
}
```



```
Tiene 12 caracteres  
Process finished with exit code 0
```

Imagen 27. Llamada a función

## Ejercicio1 : “Probando las Variables”

1. Crear una función main
2. Dentro de la función main declarar una variables mutable y una variable inmutable de forma inteligente.
3. Modifica el valor de la variable mutable.
4. Imprime el valor por pantalla de las dos variables.

### Método main

```
fun main() {  
    var a = 10  
    val e = "Goku"  
    a = 20  
    println("Variable mutable: $a")  
    println("Variable inmutable: $e")  
}
```

```
Variable mutable: 20  
Variable inmutable: Goku  
  
Process finished with exit code 0
```

Imagen 28. Resultado de la ejecución del método main - Ejercicio 1.



## Ejercicio 2: “Probando las Variables Numéricas”

1. Crear una función main
2. Dentro de la función main declarar tres variables numéricas con los valores 10, 33, 66.
3. Declara una variable resultado que guarde la suma de las 3 variables.
4. Imprime la variable resultado
5. Modificar la variable con el valor 10, por el valor 55.
6. Imprimir nuevamente variable resultado
7. Declarar una variable promedio, que calcule el promedio de las 3 variables
8. Imprimir promedio

### Método main

```
fun main() {  
    var a = 10  
    val b = 33  
    val c = 66  
    var resultado = a + b + c  
    println("$a + $b + $c = $resultado")  
    a = 55  
    resultado = a + b + c  
    println("$a + $b + $c = $resultado")  
    val promedio = resultado/3  
    print("Promedio : $promedio")  
}
```

```
10 + 33 + 66 = 109  
55 + 33 + 66 = 154  
Promedio : 51  
Process finished with exit code 0
```

Imagen 29. Resultado de la ejecución del método main - Ejercicio 2.

# Variables Básicas en Kotlin

---

## Competencias:

- Variables enteras.
- Variables de coma flotante.
- Variables booleanas.
- Variables String

## Introducción

Conoceremos las variables Básicas que existen en Kotlin y para qué sirve cada una de estos tipos de datos.

### Variables enteras

En Kotlin existen 4 tipos de variables básicas para trabajar con números, y estos son: Byte, Short, Int y Long y el uso de estas variables dependerá del número que nosotros queramos almacenar en nuestra variable.

Tipo	Tamaño	Rango de valor
Byte	8 bits	-128 a 127
Short	16 bits	-32768 a 32767
Int	32 bits	-2147483648 a 2147483647
Long	64 bits	-9223372036854775808 a 9223372036854775807

Nota: Para que los alumnos no tengan que saber estos valores de memoria, recuerde que en cada clase Byte, Short, Int y Long existe una constante llamada `MAX_VALUE` y `MIN_VALUE`.

```
fun main() {  
    val byteMaximo = Byte.MAX_VALUE  
    val byteMinimo = Byte.MIN_VALUE  
    val shortMinimo = Short.MIN_VALUE  
    val shortMaximo = Short.MAX_VALUE  
    val intMinimo = Int.MIN_VALUE  
    val intMaximo = Int.MAX_VALUE  
    val longMinimo = Long.MIN_VALUE  
    val longMaximo = Long.MAX_VALUE  
}
```

## Variables de coma flotante

En Kotlin existen 2 tipos de variables básicas para trabajar con valores de coma flotante y estos son:

Float y Double, uso de estas variables dependerá del número que nosotros queramos almacenar en nuestra variable.



Imagen 30. Tipos de variables básicas

Para el caso de las variables Float, al momento de asignar valor a la variable, esta debe venir acompañada con una letra “f” o “F”:

```
fun main() {  
    val x = 19.5f  
    var y : Double = 20.5  
}
```

## Variables booleanas

Las variables booleanas almacenan sólo dos valores, “verdadero” o “falso”

```
fun main() {  
    val x = true  
    val y : Boolean  
    y = false  
}
```

## Cadenas de caracteres

En Kotlin existen dos tipos básicos para trabajar con cadenas de caracteres. En el caso de que se necesite almacenar un carácter único utilizamos el tipo Char. En el caso de que se necesite almacenar una cadena de caracteres, utilizamos String.

```
fun main() {  
    val x = 'a'  
    val y = "Cadena de caracteres"  
    var vocal : Char  
    val nombre : String  
    vocal = 'u'  
    nombre = "Chizuru Kagura"  
}
```

En este ejemplo podemos ver cómo al igual que en Java, los objetos String se utilizan con comilla doble “Esto es un String” y los objetos Char se utilizan con comilla simple ‘A’

### Ejercicio 3: “Probando las Variables Básicas”

1. Crear una función main
2. Dentro de la función main declarar una variable de tipo Int, String, Char y Float
3. Imprime el valor por pantalla de todas variables.

#### Solución: Método main

```
fun main() {  
    val entera = 10  
    val cadena = "Goku"  
    val vocal = 'U'  
    val flotante = 10F  
    println("Variable Int: $entera")  
    println("Variable String: $cadena")  
    println("Variable Char: $vocal")  
    println("Variable Float: $flotante")  
}
```

```
Variable Int: 10  
Variable String: Goku  
Variable Char: U  
Variable Float: 10.0  
  
Process finished with exit code 0
```

Imagen 31. Resultado de la ejecución con el método main - Ejercicio 3

# Tipos de funciones

---

## Competencias:

- Tipos de funciones
- Métodos Java en relación a las Funciones Kotlin
- Funciones en Kotlin
- Conocer el uso de Void, void y Unit en Kotlin
- Funciones con parámetros

## Introducción

Hasta ahora hemos trabajado con la función “main”, en este capítulo aprenderemos a separar nuestro código en funciones Kotlin, veremos los tipos de funciones que tiene Kotlin y cómo podemos utilizar nuestras funciones en distintas partes de nuestro proyecto.

### Tipos de Funciones

Para poder comprender mejor los tipos de funciones en Kotlin, comparemos las funciones de Kotlin con las funciones de Java.

En Java, a las funciones se llaman métodos, y estos se clasifican en dos grupos, métodos sin tipo de retorno y métodos con tipo de retorno.

### Método sin tipo de retorno

```
public void imprimirNombreCompleto(){  
    System.out.println("Sanji Vinsmoke");  
}
```

Son todos los métodos declarados como “void”, y no necesitan un return dentro de sus llaves { }.

### Métodos con tipo de retorno

```
public String getNombreCompleto(){  
    return "Monkey D. Luffy";  
}
```

Son todos los métodos que declaran una clase en antes de especificar su nombre y devuelve un objeto de la clase declarada. Estos métodos tienen la obligación de llevar un return entre sus llaves { }.

En el ejemplo anterior clase declarada es “**String**” y el valor del objeto retornado es “**Monkey D. Luffy**”.

# Funciones en Kotlin

En Kotlin, solo existen los métodos con tipo de retorno y a estos métodos se les llama funciones.

Y estos se declaran de la siguiente forma:

```
fun getNombreCompleto(): String{  
    return "Monkey D. Luffy"  
}
```

- “fun”: Indica que es una función Kotlin.
- “getNombreCompleto”: es el nombre de la función.
- “ ( ) ”: Indica los parámetros que tiene la función, en este caso es vacío o sin parámetro.“
- “ : String ” : indica el tipo de retorno que tendrá la función, en este caso un “**String**”.
- “ return “**Monkey D. Luffy**” “ : Es el objeto retornado por la función.

## Repasando métodos void de Java

Ya se mencionó que Kotlin solo trabaja con funciones que retornan valores, entonces: ¿Qué pasa con los métodos void que tiene Java y existen en Kotlin ?

Para responder a esta pregunta, entendamos bien lo que es un método void. “void” es una palabra reservada del lenguaje Java, y lo utilizamos para indicar que un método no retorna valor. “Void” con mayúscula, es el nombre de una clase Java que se encuentra en el package “java.lang”, es un wrapper que transforma un valor primitivo “void” a un objeto Java.

El uso de la clase “Void” no es muy popular, esta clase existe desde que Java soporto el uso de objetos genéricos “Generics”. Todo valor primitivo Java tiene un equivalente en una clase Java, por eso existe la clase Integer, Boolean, Double, Float, etc.

## Funciones void en Kotlin

En Kotlin no existen la palabra reservada “void”, sólo existe la clase “Void”, pero si le indicamos a una función que retornará un objeto tipo “Void” nos llevaremos una sorpresa.

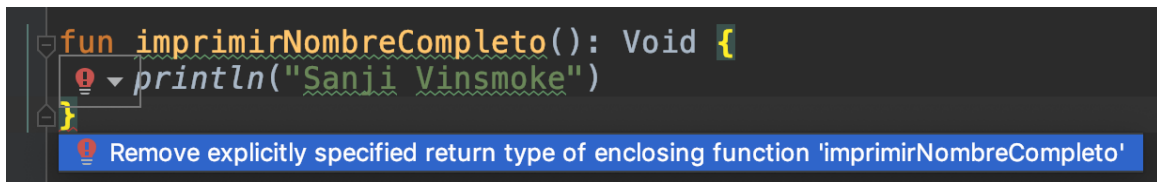


Imagen 32. Función void

Me dice que remueva explícitamente el “Void” especificado como tipo de retorno.

Entonces... ¿Cómo declaro una función Void en Kotlin?

R) con la clase “Unit”

La clase Unit es el equivalente en Kotlin a la variable primitiva “void” de Java

```
fun imprimirNombreCompleto(): Unit {  
    println("Sanji Vinsmoke")  
}
```

Si declaramos una función que retorna un tipo “Unit”, automáticamente Kotlin detecta que esa función no retorna nada (como los void en Java).

De hecho si declaramos explícitamente que nuestra función retorna un tipo “Unit”, el mismo lenguaje te sugiere que no lo agregues. Ya que todas las funciones en Kotlin que no tengan declarado el tipo de retorno, automáticamente se declara un tipo de retorno “Unit” y lo podemos ver en la siguiente imagen.



Imagen 33. Retorno "Unit"

Así se declara una función “void” en Kotlin y ya sabemos que en realidad devuelve un “Unit”

```
fun imprimirNombreCompleto() {  
    println("Sanji Vinsmoke")  
}
```

Imagen 34. Declarar función void en Kotlin que devuelve un "Unit"



## Funciones con parámetros

La forma de declarar los parámetros en una función Kotlin es indicando el nombre de la variable, seguido por " : " donde indicamos el tipo de dato de esta variable.

En caso de querer agregar más de un parámetro, este se indica por medio de una " , " Ej:

```
fun imprimirNombreCompleto(nombres: String, apellidos: String, edad: Int){
    println("Nombres: $nombres")
    println("Apellidos: $apellidos")
    print("Edad: $edad")
}
```

## Función Main en Kotlin

Una de las funciones más importantes a nivel de programación es el método main.

El método main es tan importante, que si desarrollamos una aplicación y no creamos el método main, no podemos ejecutar nuestra aplicación.

En Kotlin a los métodos les llamamos funciones, y la función main no es la excepción.

A diferencia de Java, la función main de Kotlin es mucho más sencilla de declarar

Ejemplo:

```
fun main(){
    println("Hola Mundo, soy la función Main y no soy estática.")
}
```

Otro tema importante a destacar, es que no necesitamos que nuestra función main se encuentre dentro de una clase, es decir, solo creando un archivo Kotlin ya es suficiente para poder declarar nuestra función main.



Imagen 35. Función main dentro del archivo .kt

## Ejercicio 4: “Probando las funciones”

1. Crear una función llamada `getNombre` que devuelva un `String` con tu nombre.
2. Crea una función llamada `imprimirNombreCompleto` que reciba los siguientes parámetros “nombres”, “apellidos”, edad.
3. Crear una función llamada `main()` que llame a estas dos funciones.

### Solución

```
► fun main(){  
    println("Nombre: ${getNombre()}")  
    imprimirNombreCompleto( nombre: "Roger", apellido: "Gol D.", edad: 53)  
}
```

Imagen 36. Código método `main` - Ejercicio 4

```
Nombre: Sanji Vinsmoke  
Nombre: Roger  
Apellido: Gol D.  
Edad: 53  
Process finished with exit code 0
```

Imagen 37. Resultado de la ejecución del método `main` - Ejercicio 4