

SENA
Centro De Comercio Y Turismo

Miguel Angel Ibarra Ocampo
Informe Estructuras De Datos Kotlin

Instructor
Cristian David Henao Hoyos

19/02/2023
Armenia, Quindío

1. Introducción a las estructuras de datos en Kotlin

a. ¿Qué son las estructuras de datos y para qué se utilizan?

R/= Las estructuras de datos sirven para organizar y almacenar datos para que puedan ser utilizados de manera efectiva.

b. Ventajas de utilizar estructuras de datos en Kotlin

R/= Kotlin al ser un lenguaje moderno, es muy versátil y posee varios puntos fuertes al momento de trabajar con estructuras de datos como:

- La variedad de estructuras de datos con las que se pueden trabajar.
- Su sintaxis clara y sencilla para trabajar con las estructuras.
- Su interoperabilidad para trabajar conjuntamente con otros lenguajes permite la integración de estructuras de datos de diferentes fuentes.

c. Diferencias entre las estructuras de datos en Kotlin y Java

R/= A pesar de que kotlin y java comparten muchas características debido a que kotlin fue diseñado a partir de java para manejar la interoperabilidad y ejecutar código java, existen diferencias en las estructuras de datos entre estas dos:

- **Null Safety:** En kotlin las variables no pueden tener un valor nulo a menos de que sea especificado, esto para prevenir errores.
- **Inmutabilidad:** En kotlin se usan las estructuras de datos inmutables, es decir, estructuras que no pueden ser modificadas, para prevenir modificaciones accidentales de los datos.
- **Rangos:** Kotlin ofrece un método de rangos en el que se pueden trabajar con los valores dentro del rango especificado.
- En kotlin se pueden usar funciones de una clase sin necesidad de heredarla.
- Kotlin infiere el tipo de dato en una variable para mejorar la sintaxis y hacerla más legible y concisa.

2. Arreglos en Kotlin

a. ¿Qué es un arreglo?

R/= Un array o arreglo es una estructura que nos va a permitir almacenar diferentes datos de un mismo tipo

b. Creación de arreglos en Kotlin

R/= En kotlin se pueden trabajar arreglos de dos formas principales, haciendo uso del constructor array y la otra es haciendo uso de la función array of

Constructor array:

```
Array.kt X
Array.kt
1 fun main () {
2
3 // Especificamos el tipo de dato que tendra el arreglo = Array<String>
4 // Posteriormente le asignamos el tamaño al arreglo y hacemos uso de la funcion lamda =>{i -> ""} para indicar que
  los indices almacenaran valores de tipo cadena
5     var array:Array<String> = Array(5,{i -> ""})
6     array[0]="Armenia"
7     array[1]="Medellin"
8     array[2]="Bogota"
9     array[3]="Valledupar"
10    array[4]="Cartagena"
11
12 }
```

Función Array Of:

```
Array.kt X ArrayOf.kt X
ArrayOf.kt
1 fun main () {
2 //No es necesario detallar el tipo de dato, ya que kotlin ya identifica que hay una diferencia de tipos
3 //Dentro del array of inicializaremos por defecto los elementos
4     var array= arrayOf("Arroz", "Carne", "Sal")
5 }
```

c. Accediendo a los elementos de un arreglo

R/= Para acceder a un elemento de un arreglo nos basamos en los índices, los cuales van del 0 al tamaño-1

Constructor array:

```
Array.kt x ArrayOf.kt
Array.kt
1 fun main () {
2     var array: Array<String> = Array(5, { i -> "" })
3     array[0] = "Armenia"
4     array[1] = "Medellin"
5     array[2] = "Bogota"
6     array[3] = "Valledupar"
7     array[4] = "Cartagena"
8
9     println(array[2])
10    println(array[4])
11 }
12
13

PROBLEMAS SALIDA TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN

PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin> java -jar Array.jar
Bogota
Cartagena
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin> 
```

Función Array Of:

```
Array.kt ArrayOf.kt x
ArrayOf.kt
1 fun main () {
2     //No es necesario detallar el tipo de dato, ya que kotlin ya identifica que hay una diferencia de tipos
3     //Dentro del array of inicializaremos por defecto los elementos
4     var array = arrayOf("Arroz", "Carne", "Sal")
5
6     println("El elemento en la posicion 2 es: " + array[2])
7 }
8
9

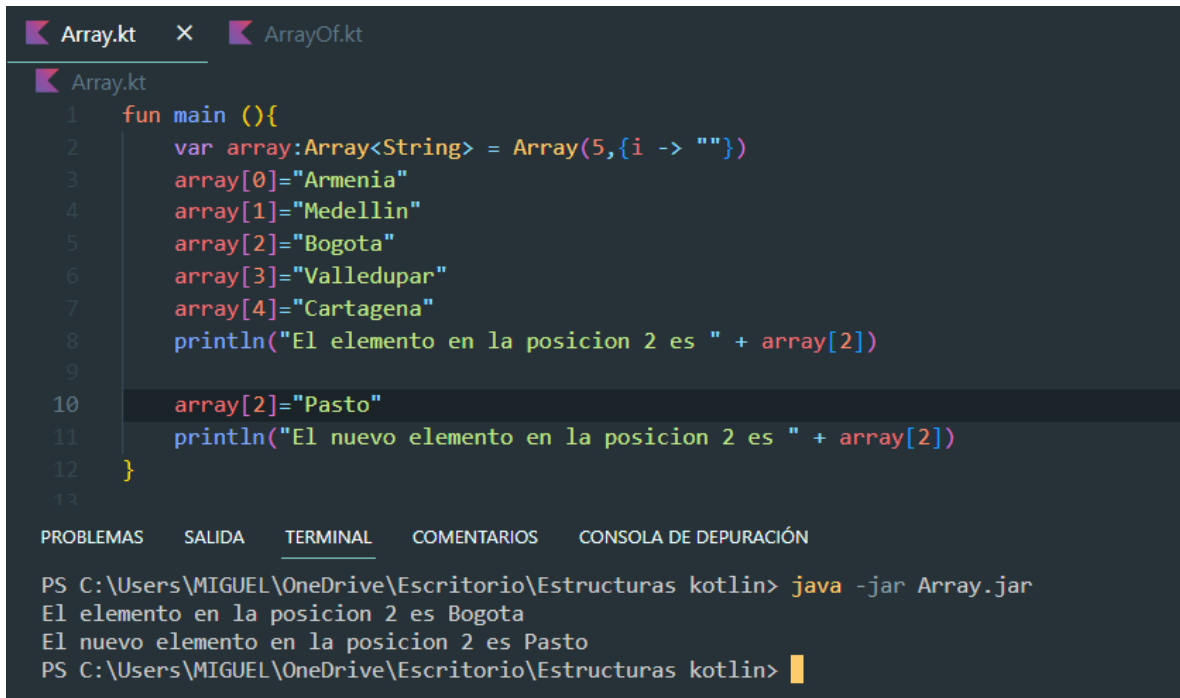
PROBLEMAS SALIDA TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN power

PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin> java -jar ArrayOf.jar
Sal
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin> kotlinc ArrayOf.kt -include-runtime -d ArrayOf.jar
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin> java -jar ArrayOf.jar
El elemento en la posicion 2 es: Sal
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin> 
```

d. Modificando los elementos de un arreglo

Constructor array:

Le damos el nuevo valor poniendo el nombre del array, la posición y el nuevo valor.



The screenshot shows an IDE with two tabs: `Array.kt` and `ArrayOf.kt`. The `Array.kt` tab is active, displaying the following Kotlin code:


```
1 fun main () {
2     var array: Array<String> = Array(5, {i -> ""})
3     array[0] = "Armenia"
4     array[1] = "Medellin"
5     array[2] = "Bogota"
6     array[3] = "Valledupar"
7     array[4] = "Cartagena"
8     println("El elemento en la posicion 2 es " + array[2])
9
10    array[2] = "Pasto"
11    println("El nuevo elemento en la posicion 2 es " + array[2])
12 }
13
```

Below the code editor, there are tabs for `PROBLEMAS`, `SALIDA`, `TERMINAL`, `COMENTARIOS`, and `CONSOLA DE DEPURACIÓN`. The `TERMINAL` tab is selected, showing the output of the program:

```
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin> java -jar Array.jar
El elemento en la posicion 2 es Bogota
El nuevo elemento en la posicion 2 es Pasto
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin>
```

Función Array Of:

Cambiamos el elemento dentro del arreglo por medio del `array.set`, especificamos la posición y el nuevo valor.



The screenshot shows an IDE with two tabs: `Array.kt` and `ArrayOf.kt`. The `ArrayOf.kt` tab is active, displaying the following Kotlin code:

```
1 fun main () {
2
3     var array = arrayOf("Arroz", "Carne", "Sal")
4     //Con el set le asignamos el nuevo valor especificando la posicion que deseamos tomar y el nuevo valor.
5     array.set(2, "Limon")
6
7     println("El elemento en la posicion 2 es: " + array[2])
8 }
9
10
```

Below the code editor, there are tabs for `PROBLEMAS`, `SALIDA`, `TERMINAL`, `COMENTARIOS`, and `CONSOLA DE DEPURACIÓN`. The `TERMINAL` tab is selected, showing the output of the program:

```
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin> java -jar ArrayOf.jar
El elemento en la posicion 2 es: Limon
```

e. Recorriendo un arreglo

La sentencia se compone de una declaración de variables, una expresión contenedora, compuesta por el operador in



```
Array.kt x ArrayOf.kt
Array.kt
1 fun main () {
2     var array:Array<String> = Array(5,{i -> ""})
3     array[0]="Armenia"
4     array[1]="Medellin"
5     array[2]="Bogota"
6     array[3]="Valledupar"
7     array[4]="Cartagena"
8
9     for (i in array) {
10        println(i)
11    }
12 }
13

PROBLEMAS SALIDA TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN

PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin> java -jar Array.jar
Armenia
Medellin
Bogota
Valledupar
Cartagena
```

f. Funciones útiles para trabajar con arreglos en Kotlin

Algunas funciones útiles para trabajar con arreglos en kotlin son:



```
Array.kt x ArrayOf.kt
Array.kt
1 fun main () {
2     var array:Array<String> = Array(5,{i -> ""})
3     array[0]="Armenia"
4     array[1]="Medellin"
5     array[2]="Bogota"
6     array[3]="Valledupar"
7     array[4]="Cartagena"
8
9     // .size devuelve el tamaño del array.
10    println(array.size)
11    // .get esta función devuelve el valor de un elemento en una posición dada en el array.
12    println(array.get(0))
13    // .set esta función permite modificar el valor de un elemento en una posición dada en el array.
14    array.set(4, "Santander")
15    println(array.get(4))
16    // indexOf: esta función devuelve la posición de la primera aparición de un valor en el array, o -1 si el valor
17    // no está presente.
18    println(array.indexOf("Bogota"))
19 }
20

PROBLEMAS SALIDA TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN powershell + v

PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin> java -jar Array.jar
5
Armenia
Santander
2
```

3. Listas en Kotlin

a. ¿Qué es una lista?

R/= una lista es una estructura de datos en la cual cada elemento puede ser de cualquier tipo de dato existen dos tipos de listas mutables (que pueden modificarse luego de su creación, agregar o quitar elementos.) y no mutables.

b. Creación de listas en Kotlin

R/= Para la creación de las listas se realiza de la siguiente manera:

```
listas.kt X
Listas > listas.kt
1 fun main(){
2 //Lista no mutable
3 val list=listOf(1,2,3,4,5,6)
4
5 //Lista mutable
6 val countries=mutableListOf("Perú", "Canada", "Colombia", "Noruega", "Japon")
7
8 }
```

c. Accediendo a los elementos de una lista

R/ Para acceder a los elementos de una lista utilizamos el método .get seguido de la posición del elemento al que queremos acceder, al igual que en los arreglos:


```
listas.kt X
Listas > listas.kt
1 fun main(){
2 //Lista no mutable
3 val list=listOf(1,2,3,4,5,6)
4 println(list.get(2))
5
6 //Lista mutable
7 val countries=mutableListOf("Perú", "Canada", "Colombia", "Noruega", "Japon")
8 println(countries.get(1))
9 }
10

PROBLEMAS SALIDA TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN Code
[Running] cd "c:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Listas\" && kotlinc listas.kt
-include-runtime -d listas.jar && java -jar listas.jar
3
Canada
```

d. Modificando los elementos de una lista

R/= En las listas inmutables no es posible modificar los elementos, pero en las listas mutables se realiza de la siguiente manera:

- Por medio del índice y el nuevo valor= `countries [1] = "MEXICO"`
- Por medio de la función `set`, el índice y el nuevo valor = `countries.set(3, "Marruecos")`



```
listas.kt x
Listas > listas.kt
1 fun main(){
2 //Lista no mutable
3 // val list=ListOf(1,2,3,4,5,6)
4 // println(list.get(2))
5
6 //Lista mutable
7 val countries=mutableListOf("Perú", "Canada", "Colombia", "Noruega", "Japon")
8 countries[1]= "MEXICO"
9 countries.set(3, "Marruecos")
10 println(countries.get(1))
11 println(countries.get(3))
12 }
13

PROBLEMAS SALIDA TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN Code
[Done] exited with code=0 in 11.275 seconds

[Running] cd "c:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Listas\" && kotlinc listas.kt
-include-runtime -d listas.jar && java -jar listas.jar
MEXICO
Marruecos
```


e. Recorriendo una lista

R/ Para recorrer una lista se puede realizar de dos formas, por medio de un ciclo for o por un forEach, ambos funcionan para cualquier tipo de lista y de dato.

```
listas.kt
Listas > listas.kt
1 fun main(){
2     //Lista no mutable
3     val list=listOf(1,2,3,4,5,6)
4     //Recorrer por medio de un forEach
5     list.forEach{element ->
6         println(element)
7     }
8
9     //Lista mutable
10    val countries=mutableListOf("Perú", "Canada", "Colombia", "Noruega", "Japon")
11    countries[1]= "MEXICO"
12    countries.set(3, "Marruecos")
13    //recorrer una lista por medio de for
14    for (i in countries) {
15        println(i)
16    }
17
18 }
19
```

PROBLEMAS TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN SALIDA

```
[Running] cd "c:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin>Listas\" && kotl
1
2
3
4
5
6
Per
MEXICO
Colombia
Marruecos
Japon
```

f. Funciones útiles para trabajar con listas en Kotlin

R/ Algunas de las funciones esenciales para trabajar con listas son:

```
listas.kt ×
Listas > listas.kt
1 fun main(){
2     //Lista no mutable
3     // val list=listOf(1,2,3,4,5,6)
4     val asiaCountries = listOf("Japon", "China", "Indonesia", "china")
5
6     //Lista mutable
7     val countries=mutableListOf("Peru", "Canada", "Colombia", "Noruega", "Argentina")
8     //Cambiar un elemento con el .set
9     countries.set(3, "Marruecos")
10    println(countries)
11    // .add agrega un elemento al final de una lista mutable.
12    countries.add("España")
13    println(countries)
14    // .removeAt elimina un elemento de una lista mutable por medio del indice.
15    countries.removeAt(0)
16    println(countries)
17    // Funcion para agregar todos los elementos de una lista mutable o no a otra lista.
18    countries.addAll(asiaCountries)
19    println(countries)
20 }
```

PROBLEMAS TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN SALIDA

```
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin>Listas> java -jar listas.jar
[Peru, Canada, Colombia, Marruecos, Argentina]
[Peru, Canada, Colombia, Marruecos, Argentina, España]
[Canada, Colombia, Marruecos, Argentina, España]
[Canada, Colombia, Marruecos, Argentina, España, Japon, China, Indonesia, china]
```

4. Conjuntos en Kotlin

a. ¿Qué es un conjunto?

R/= un conjunto (set en inglés) es una colección de elementos únicos y desordenados, es decir, no permite elementos duplicados y no garantiza un orden específico de los elementos. Los conjuntos son una estructura de datos muy útil en programación, ya que permiten almacenar elementos sin duplicados y pueden ser utilizados para realizar operaciones de conjunto como la unión, la intersección y la diferencia.

Al igual que las listas existen conjuntos mutables e inmutables

b. Creación de conjuntos en Kotlin

```
set.kt X
Conjuntos > set.kt
1 fun main () {
2
3 // creacion conjunto o set mutable
4 val state = mutableSetOf("gas", "liquido", "solido")
5
6
7 // creacion conjunto o set immutable
8 val planets = setOf("Earth", "Mars", "mercury", "Uranus", "Saturn")
9
10 }
```

c. Accediendo a los elementos de un conjunto

R/= Los conjuntos no poseen índices, por esta razón no se puede acceder a los elementos por medio de este, por el contrario se puede visualizar si un elemento pertenece a un conjunto, otra alternativa consta en convertir un conjunto a una lista por medio del `toList()`

```
set.kt X listas.kt
Conjuntos > set.kt
1 fun main () {
2
3 // creacion conjunto o set mutable
4 val state = mutableSetOf("gas", "liquido", "solido")
5 //conversion de un conjunto a una lista por medio del .toList
6 val listState = state.toList()
7 //las listas si poseen indices, por lo tanto podemos usar el .get
8 println(listState.get(2))
9
10
11 // creacion conjunto o set immutable
12 val planets = setOf("Earth", "Mars", "mercury", "Uranus", "Saturn")
13 //funcion .contains para validar que exista un elemento dentro del conjutno
14 if (planets.contains("Mars")) {
15     println("El conjunto contiene la palabra 'Mars'")
16 } else {
17     println("El conjunto no contiene la palabra 'Mars'")
18 }
19 }
```

PROBLEMAS TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN SALIDA

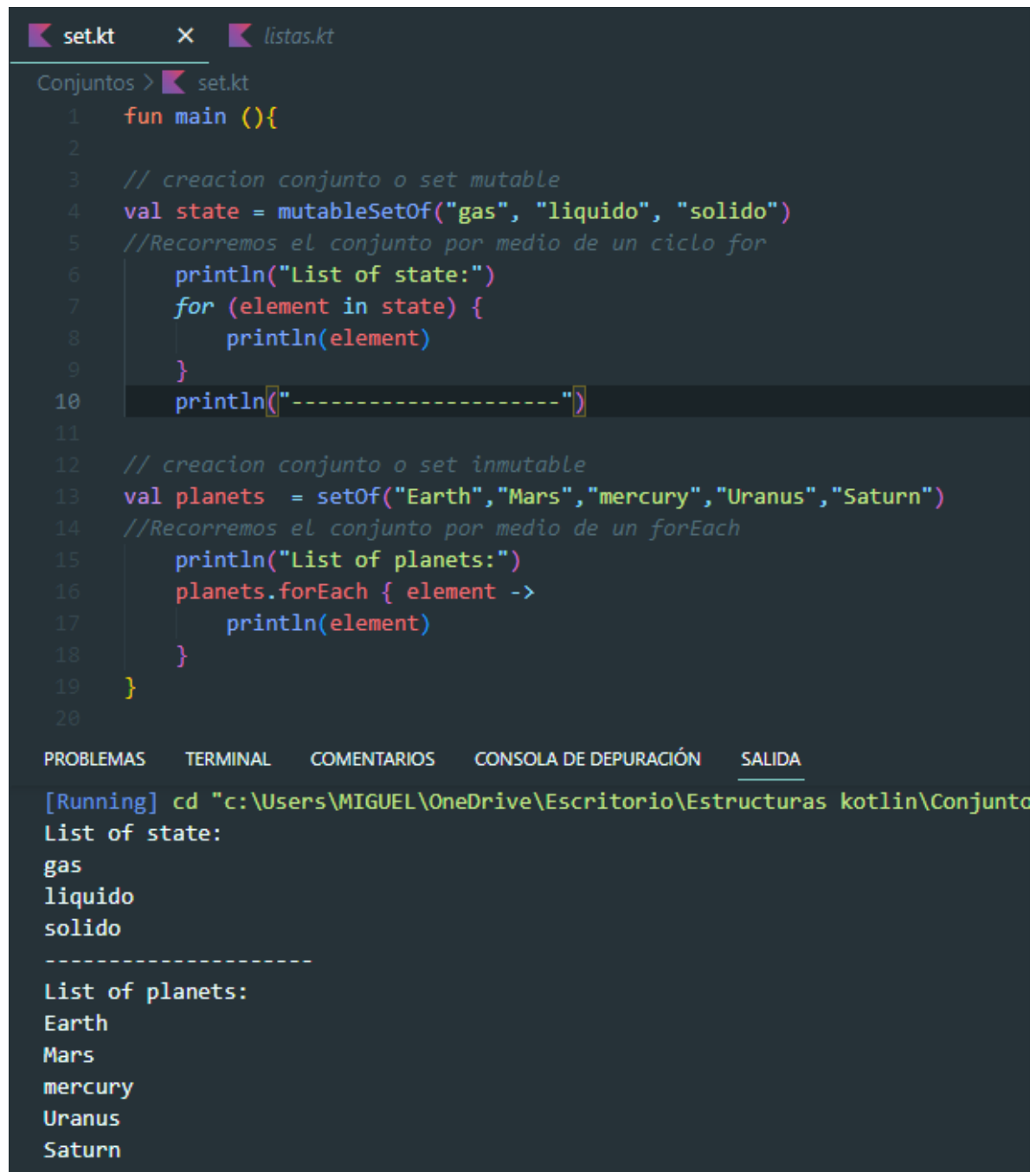
```
[Running] cd "c:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Conjuntos\" && ko
-jar set.jar
solido
El conjunto contiene la palabra 'Mars'
```

d. Modificando los elementos de un conjunto

R/ Los conjuntos al no poseer índices, no es posible acceder individualmente a un elemento, por lo tanto no se puede modificar un elemento en específico, una alternativa seria eliminar el elemento a actualizar y agregar al conjunto el nuevo valor.

e. Recorriendo un conjunto

R/ Los conjuntos se pueden recorrer de dos maneras, un ciclo for o un forEach:



```
set.kt  x  listas.kt
Conjuntos > set.kt
1  fun main (){
2
3  // creacion conjunto o set mutable
4  val state = mutableSetOf("gas", "liquido", "solido")
5  //Recorremos el conjunto por medio de un ciclo for
6      println("List of state:")
7      for (element in state) {
8          println(element)
9      }
10     println("-----")
11
12     // creacion conjunto o set immutable
13     val planets = setOf("Earth","Mars","mercury","Uranus","Saturn")
14     //Recorremos el conjunto por medio de un forEach
15     println("List of planets:")
16     planets.forEach { element ->
17         println(element)
18     }
19 }
20
```

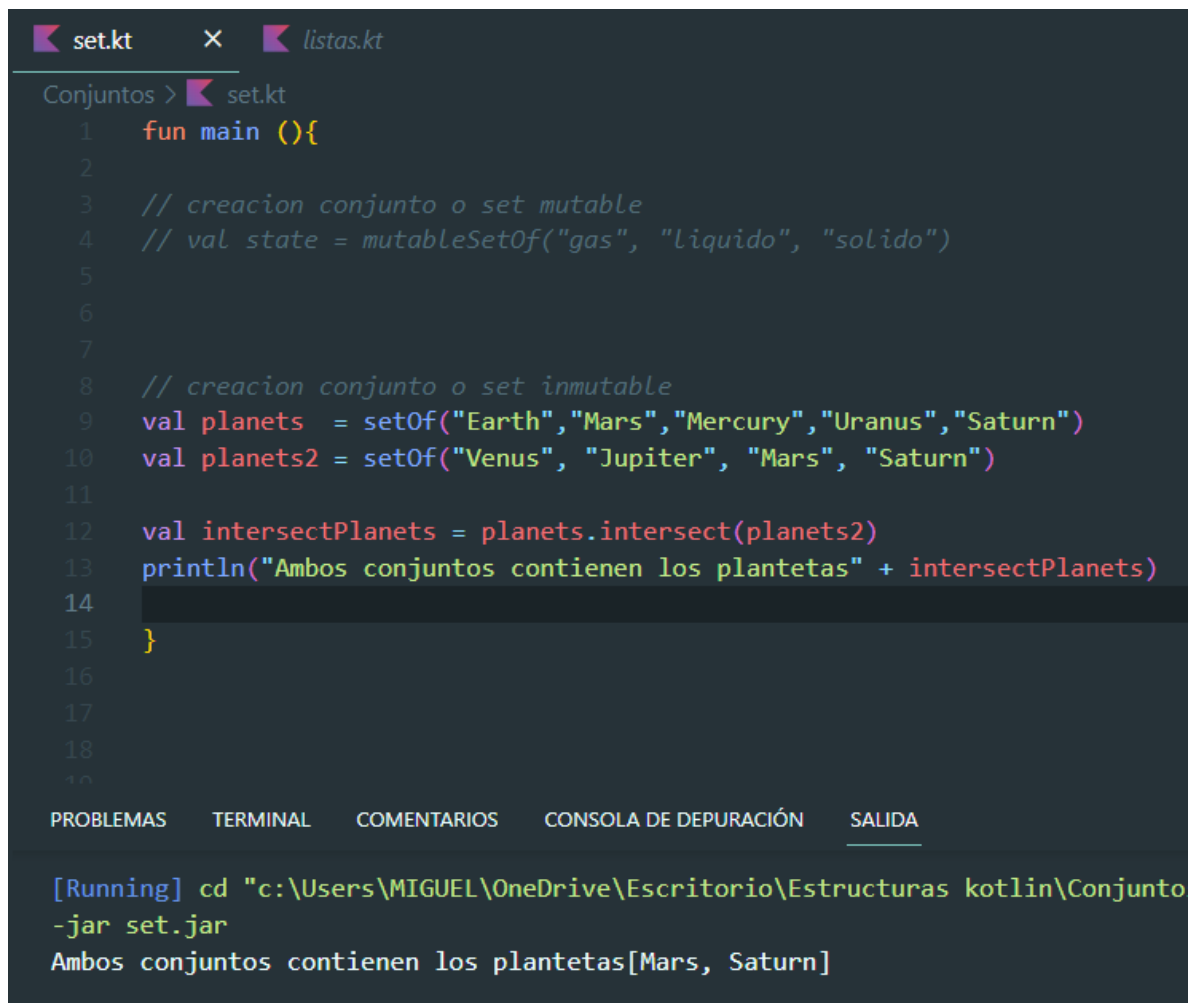
PROBLEMAS TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN SALIDA

```
[Running] cd "c:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Conjuntos"
List of state:
gas
liquido
solido
-----
List of planets:
Earth
Mars
mercury
Uranus
Saturn
```

f. Funciones útiles para trabajar con conjuntos en Kotlin

R/ Algunas de las funciones más útiles para trabajar conjuntos o sets en kotlin son:

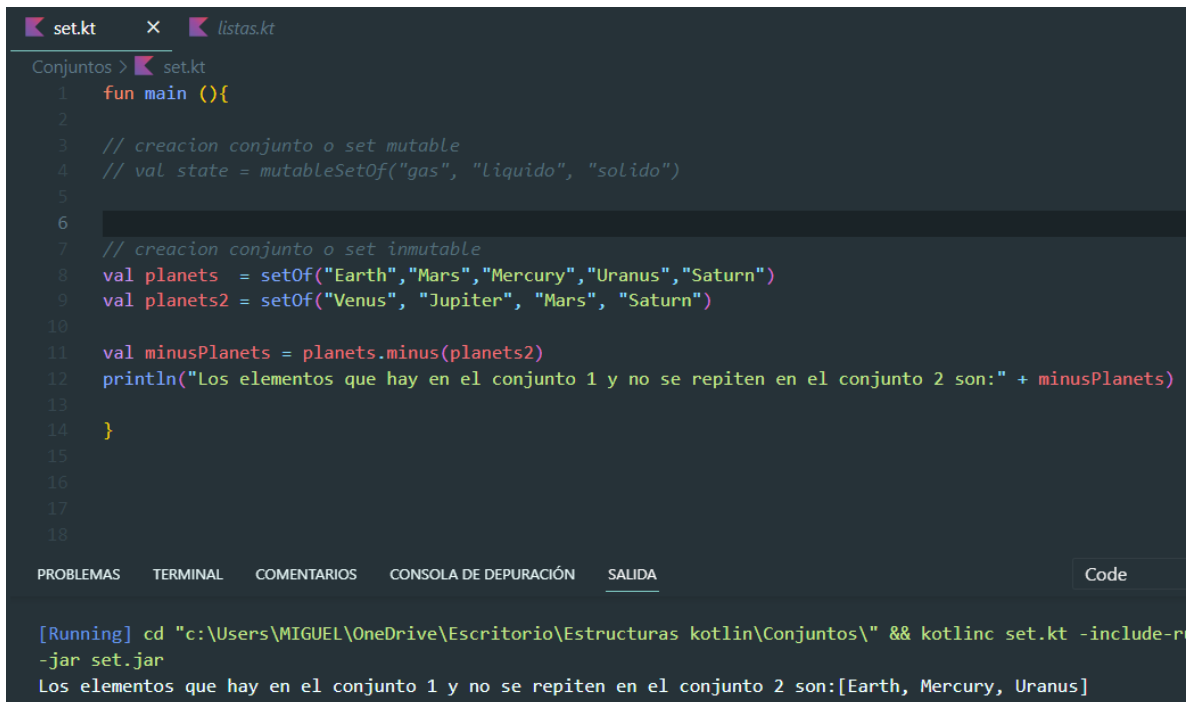
intersect (): sirve para comparar dos conjuntos y crear uno nuevo con elementos que pertenezcan a ambos conjuntos.



```
set.kt  X  listas.kt
Conjuntos > set.kt
1  fun main (){
2
3  // creacion conjunto o set mutable
4  // val state = mutableSetOf("gas", "liquido", "solido")
5
6
7
8  // creacion conjunto o set immutable
9  val planets = setOf("Earth","Mars","Mercury","Uranus","Saturn")
10 val planets2 = setOf("Venus", "Jupiter", "Mars", "Saturn")
11
12 val intersectPlanets = planets.intersect(planets2)
13 println("Ambos conjuntos contienen los planetas" + intersectPlanets)
14
15 }
16
17
18
19
20
PROBLEMAS  TERMINAL  COMENTARIOS  CONSOLA DE DEPURACIÓN  SALIDA

[Running] cd "c:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Conjunto
-jar set.jar
Ambos conjuntos contienen los planetas[Mars, Saturn]
```

minus (): Crea un nuevo conjunto en el cual sus elementos estén en el primer conjunto, pero no en el segundo



```
set.kt x listas.kt
Conjuntos > set.kt
1 fun main (){
2
3 // creacion conjunto o set mutable
4 // val state = mutableSetOf("gas", "liquido", "solido")
5
6
7 // creacion conjunto o set immutable
8 val planets = setOf("Earth", "Mars", "Mercury", "Uranus", "Saturn")
9 val planets2 = setOf("Venus", "Jupiter", "Mars", "Saturn")
10
11 val minusPlanets = planets.minus(planets2)
12 println("Los elementos que hay en el conjunto 1 y no se repiten en el conjunto 2 son:" + minusPlanets)
13
14 }
15
16
17
18
```

PROBLEMAS TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN SALIDA Code

[Running] cd "c:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Conjuntos\" && kotlinc set.kt -include-r
-jar set.jar
Los elementos que hay en el conjunto 1 y no se repiten en el conjunto 2 son:[Earth, Mercury, Uranus]

union (): Crea un nuevo conjunto que incluye todos los elementos de ambos conjuntos,



```
set.kt x listas.kt
Conjuntos > set.kt
1 fun main (){
2
3 // creacion conjunto o set mutable
4 val state1 = mutableSetOf("gas", "liquido", "liquido", "plasma")
5 val state2 = mutableSetOf("solido", "plasma", "liquido", "liquido")
6
7 val unionState = state1.union(state2)
8
9 println(unionState)
10 // creacion conjunto o set immutable
11 // val planets = setOf("Earth", "Mars", "Mercury", "Uranus", "Saturn")
12 // val planets2 = setOf("Venus", "Jupiter", "Mars", "Saturn")
13 }
14
```

PROBLEMAS TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN SALIDA

PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Conjuntos> java -jar set.jar
[gas, liquido, plasma, solido]
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Conjuntos> █

plus (): Crea un nuevo conjunto que incluye todos los elementos del primer conjunto más los elementos del segundo conjunto.



```
set.kt x listas.kt
Conjuntos > set.kt
1 fun main (){
2
3 // creacion conjunto o set mutable
4 val state1 = mutableSetOf("gas", "liquido", "liquido", "plasma")
5 val state2 = mutableSetOf("solido", "plasma", "liquido", "liquido")
6
7 val plusState = state1.plus(state2)
8
9 println(plusState)
10 // creacion conjunto o set immutable
11 // val planets = setOf("Earth", "Mars", "Mercury", "Uranus", "Saturn")
12 // val planets2 = setOf("Venus", "Jupiter", "Mars", "Saturn")
13 }
14

PROBLEMAS TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN SALIDA
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Conjuntos> kotlinc set.kt -include-runtime -d set.jar
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Conjuntos> java -jar set.jar
[gas, liquido, plasma, solido]
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Conjuntos>
```

La principal diferencia entre las funciones `union()` y `plus()` es que `union()` devuelve un nuevo conjunto que incluye todos los elementos de ambos conjuntos, mientras que `plus` devuelve un nuevo conjunto que incluye todos los elementos del primer conjunto más los elementos del segundo conjunto, sin incluir elementos duplicados.

filter (): crea un nuevo conjunto que incluye solo los elementos que cumplen con una condición determinada.

```
set.kt  x  listas.kt
Conjuntos > set.kt
1  fun main (){
2
3  // creacion conjunto o set mutable
4  val state = mutableSetOf("gas", "liquido", "solido", "plasma")
5
6  val filterState = state.filter{it.startsWith("g")}
7
8  println("El estado de la materia que inicia con g es: " + filterState)
9
10
11 // creacion conjunto o set immutable
12 val planets = setOf("Earth","Mars","Mercury","Uranus","Saturn")
13 val filterPlanets = planets.filter{it.endsWith("s")}
14 println("Los planetas que terminan en es son: " + filterPlanets)
15 }
16
PROBLEMAS  TERMINAL  COMENTARIOS  CONSOLA DE DEPURACIÓN  SALIDA


PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Conjuntos> java -jar set.jar
El estado de la materia que inicia con g es: [gas]
Los planetas que terminan en es son: [Mars, Uranus]
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Conjuntos> 
```

toMutableSet (): esta función se utiliza para crear una copia mutable de un conjunto inmutable.

```
set.kt  x  listas.kt
Conjuntos > set.kt
1  fun main (){
2  // creacion conjunto o set inmutable
3  val planets = setOf("Earth","Mars","Mercury","Uranus","Saturn")
4
5  val mutablePlanets = planets.toMutableSet()
6
7  mutablePlanets.add("Venus")
8
9  println(mutablePlanets)
10 }
11
12
13
14
15
16
PROBLEMAS  TERMINAL  COMENTARIOS  CONSOLA DE DEPURACIÓN  SALIDA

PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Conjuntos> java -jar set.jar
[Earth, Mars, Mercury, Uranus, Saturn, Venus]
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Conjuntos> 
```


map (): Devuelve un conjunto que contiene los resultados de aplicar la función de transformación a cada elemento del conjunto original.



```
set.kt  X  listas.kt
Conjuntos > set.kt
1  fun main () {
2
3  // creacion conjunto o set mutable
4  val state = mutableSetOf("gas", "liquido", "solido", "plasma")
5  val mapState = state.map { it.length }
6  println("La cantidad de letras de cada estado son: " + mapState)
7
8
9  // creacion conjunto o set immutable
10 val planets = setOf("Earth", "Mars", "Mercury", "Uranus", "Saturn")
11 val mapPlanets = planets.map { it.uppercase() }
12 println(mapPlanets)
13 }
14

PROBLEMAS  TERMINAL  COMENTARIOS  CONSOLA DE DEPURACIÓN  SALIDA

PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Conjuntos> java -jar set.jar
La cantidad de letras de cada estado son: [3, 7, 6, 6]
[EARTH, MARS, MERCURY, URANUS, SATURN]
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Conjuntos>
```

5. Mapas en Kotlin

a. ¿Qué es un mapa?

R/= Los mapas o diccionarios son una estructura que se componen de una clave y un valor, por lo tanto, accederemos por medio de la clave y no de un índice, siendo la clave el único identificador del valor almacenado en ese mapa.

b. Creación de mapas en Kotlin

R/= Como en las demás estructuras también existen mapas mutables e inmutables, cabe resaltar que la clave y el valor pueden ser tanto String como un entero:

```
Map.kt X
Mapas > Map.kt
1 fun main() {
2     // Mapa inmutable
3     val mapPeople = mapOf (1 to "Juan", 2 to "Nicolas", 3 to "Fabi")
4     println("Mapa inmutable: $mapPeople")
5
6     //Mapa mutable
7     val mapAnimals = mutableMapOf("Perro" to 1, "Gato" to 2, "Panda" to 3)
8     println("Mapa mutable: $mapAnimals")
9 }
```

PROBLEMAS TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN SALIDA

```
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Mapas> java -jar Map.jar
Mapa inmutable: {1=Juan, 2=Nicolas, 3=Fabi}
Mapa mutable: {Perro=1, Gato=2, Panda=3}
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Mapas>
```

c. Accediendo a los elementos de un mapa

R/= Para acceder al valor de un mapa existen 2 formas:

Con corchetes: `${nombremapa[clave]}`

```
Map.kt X
Mapas > Map.kt
1 fun main() {
2     // Mapa inmutable
3     val mapPeople = mapOf (1 to "Juan", 2 to "Nicolas", 3 to "Fabi")
4     println("Para la clave 1 el valor es: ${mapPeople[1]}")
5
6     //Mapa mutable
7     // val mapAnimals = mutableMapOf("Perro" to 1, "Gato" to 2, "Panda" to 3)
8     // println("Para la clave Panda el valor es: ${mapAnimals.get("Panda")}")
9 }
```

PROBLEMAS TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN SALIDA

```
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Mapas> java -jar Map.jar
Para la clave 1 el valor es: Juan
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Mapas>
```

Con función. Get: `${nombremapa.get(clave)}`

```
Map.kt
Mapas > Map.kt
1 fun main() {
2     // Mapa inmutable
3     // val mapPeople = mapOf(1 to "Juan", 2 to "Nicolas", 3 to "Fabi")
4     // println("Para la clave 1 el valor es: ${mapPeople[1]}")
5
6     // Mapa mutable
7     val mapAnimals = mutableMapOf("Perro" to 1, "Gato" to 2, "Panda" to 3)
8     println("Para la clave Panda el valor es: ${mapAnimals.get("Panda")}")
9 }
```

PROBLEMAS TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN SALIDA

PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Mapas> java -jar Map.jar
Para la clave Panda el valor es: 3
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Mapas> █

d. Modificando los elementos de un mapa

R/= Gracias a la flexibilidad que kotlin nos brinda existen varias formas de actualizar los valores de un mapa:

```
Map.kt
Mapas > Map.kt
1 fun main() {
2     // Mapa inmutable
3     // val mapAnimals = mapOf("Perro" to 1, "Gato" to 2, "Panda" to 3)
4
5
6     // Mapa mutable
7     val mapPeople = mutableMapOf(1 to "Juan", 2 to "Nicolas", 3 to "Fabi")
8     // seactualizar los valores existentes utilizando el operador de indexación []
9     mapPeople[2] = "Nico"
10    // Por medio de la funcion.set()
11    mapPeople.set(1, "Juan Jose")
12    // Por medio del .put()
13    mapPeople.put(3, "Fabiola")
14    println(mapPeople)
15 }
```

PROBLEMAS TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN SALIDA

PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Mapas> java -jar Map.jar
{1=Juan Jose, 2=Nico, 3=Fabiola}
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Mapas> █

e. Recorriendo un mapa

R/= Para recorrer un mapa haremos uso de un ciclo for:

```
Mapas > Map.kt
1 fun main() {
2     // Mapa inmutable
3     val mapAnimals = mapOf("Perro" to 1, "Gato" to 2, "Panda" to 3)
4     // Para acceder al elemento completo
5     for (element in mapAnimals) {
6         println(element)
7     }
8
9     // Mapa mutable
10    val mapPeople = mutableMapOf (1 to "Juan", 2 to "Nicolas", 3 to "Fabi")
11    // Para acceder a la clave y valor por separado.
12    // utilizamos el entries para obtener un conjunto de pares clave-valor que representan los elementos del mapa.
13    for ((key,value) in mapPeople) {
14        println("En la clave $key el valor es $value")
15    }
16 }
17
18
```

PROBLEMAS TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN SALIDA

```
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Mapas> java -jar Map.jar
Perro=1
Gato=2
Panda=3
En la clave 1 el valor es Juan
En la clave 2 el valor es Nicolas
En la clave 3 el valor es Fabi
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Mapas>
```

f. Funciones útiles para trabajar con mapas en Kotlin

R/=

put (): Agrega un nuevo par clave-valor al mapa y devuelve el valor anterior asociado con la clave, si existe.

```
Mapas > Map.kt
1 fun main() {
2     // Mapa inmutable
3     // val mapAnimals = mapOf("Perro" to 1, "Gato" to 2, "Panda" to 3)
4
5
6     // Mapa mutable
7     val mapPeople = mutableMapOf (1 to "Juan", 2 to "Nicolas", 3 to "Fabi")
8     // funcion put:
9     mapPeople.put(4,"Sophia")
10    println(mapPeople)
11 }
12
13
```

PROBLEMAS TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN SALIDA

```
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Mapas> java -jar Map.jar
{1=Juan, 2=Nicolas, 3=Fabi, 4=Sophia}
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Mapas>
```

Remove (): Elimina un elemento indicando la clave.

Clear (): elimina todos los pares clave-valor del mapa.

```
Mapas > Map.kt
1 fun main() {
2     // Mapa inmutable
3     val mapAnimals = mutableMapOf("Perro" to 1, "Gato" to 2, "Panda" to 3)
4     println(mapAnimals)
5     // funcion clear:
6     mapAnimals.clear()
7     println(mapAnimals)
8
9     //Mapa mutable
10    val mapPeople = mutableMapOf (1 to "Juan",2 to "Nicolas", 3 to "Fabi")
11    // funcion put:
12    mapPeople.put(4,"Sophia")
13    // funcion remove:
14    mapPeople.remove(2)
15    println(mapPeople)
16 }
--
PROBLEMAS  TERMINAL  COMENTARIOS  CONSOLA DE DEPURACIÓN  SALIDA

PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Mapas> java -jar Map.jar
{Perro=1, Gato=2, Panda=3}
{}
{1=Juan, 3=Fabi, 4=Sophia}
```

filterKeys (): Devuelve un nuevo mapa que contiene solo los pares clave-valor cuyas claves cumplen con la condición especificada.

filterValues (): Devuelve un nuevo mapa que contiene solo los pares en los cuales los valores cumplen con la condición especificada:

```
Mapas > Map.kt
1 fun main() {
2     // Mapa inmutable
3     val mapAnimals = mutableMapOf("Perro" to 1, "Gato" to 2, "Panda" to 3)
4     val filterAnimals = mapAnimals.filterKeys { it.contains("a")}
5     println(filterAnimals)
6
7     //Mapa mutable
8     val mapPeople = mutableMapOf (1 to "Juan",2 to "Nicolas", 3 to "Fabi")
9     val filterPeople = mapPeople.filterValues { it.length == 4 }
10    println(filterPeople)
11
12 }
--
PROBLEMAS  TERMINAL  COMENTARIOS  CONSOLA DE DEPURACIÓN  SALIDA

PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Mapas> java -jar Map.jar
{Gato=2, Panda=3}
{1=Juan, 3=Fabi}
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Mapas>
```

Pares en Kotlin

a. ¿Qué es un par?

R/= Los pares son un tipo de datos que permite almacenar dos valores de diferentes tipos en una sola variable. Los pares son inmutables, lo que significa que una vez que se crean, no se pueden cambiar sus valores.

b. Creación de pares en Kotlin

R/= Existen dos formas de crear estructuras pair:

```
Pair.kt
Pares > Pair.kt
1 fun main(){
2     val items = Pair("Agua", "Aceite")
3     val numbers = 1 to 100
4     println(items)
5     println(numbers)
6
7 }
```

PROBLEMAS TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN SALIDA

```
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Pares> java -jar Pair.jar
(Agua, Aceite)
(1, 100)
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Pares>
```

c. Accediendo a los elementos de un par

R/= Para acceder a los elementos de un pair se utilizan las propiedades first y second, que corresponden al primer y segundo valor del par,

```
Pair.kt
Pares > Pair.kt
1 fun main(){
2     val items = Pair("Agua", "Aceite")
3     val numbers = 1 to 100
4     // Acceder a los elementos par
5     println("Todos sabemos que el " + items.first + " y el " + items.second + " son una mezcla heterogenea")
6     println("Nate de 5 años ya sabe contar del " + numbers.first + " al " + numbers.second)
7
8 }
```

PROBLEMAS TERMINAL COMENTARIOS CONSOLA DE DEPURACIÓN SALIDA

```
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Pares> java -jar Pair.jar
Todos sabemos que el Agua y el Aceite son una mezcla heterogenea
Nate de 5 años ya sabe contar del 1 al 100
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Pares>
```

d. Modificando los elementos de un par

R/= Los pares son estructuras inmutables, es decir no se pueden modificar sus valores.

e. Recorriendo un par

R/= Debido a que el par contiene solo dos elementos o valores no es necesario recorrerlo, para acceder a estos se utiliza el `first` y el `second` explicados anteriormente.

f. Funciones útiles para trabajar con pares en Kotlin

R/=

component1 y **component2**: Son funciones que permiten acceder al primer y segundo elemento del par al igual que el `first` y el `second`

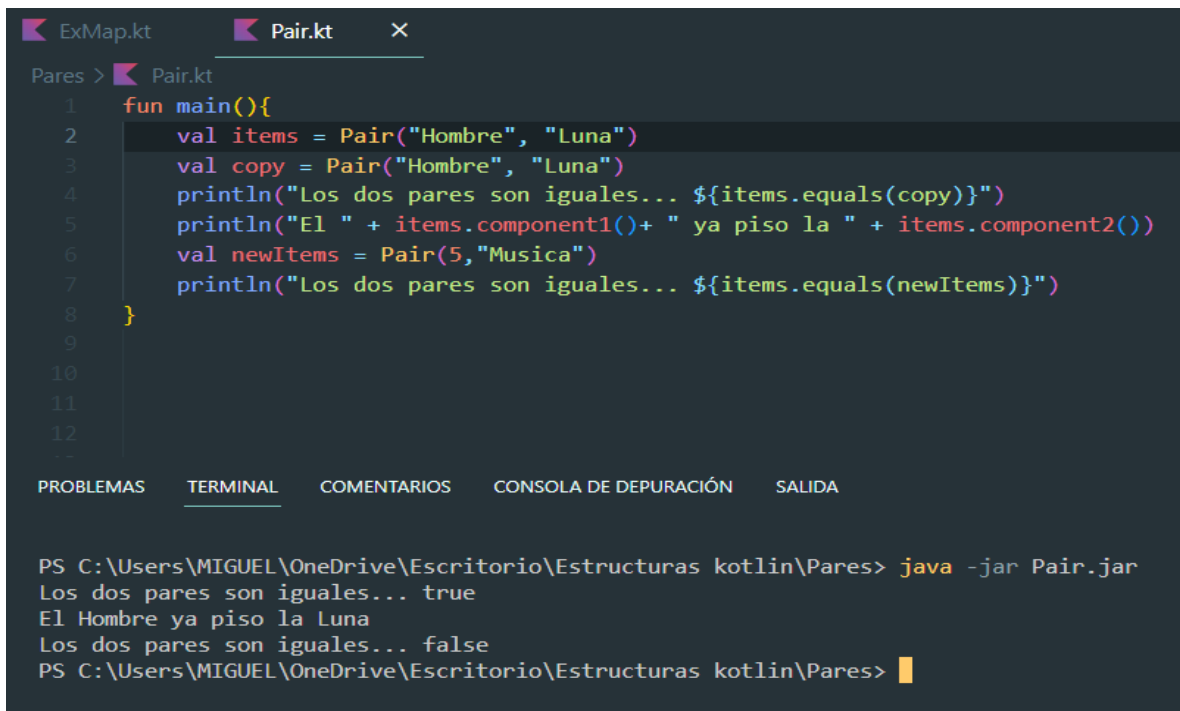


```
Pair.kt
Pares > Pair.kt
1 fun main(){
2     // val numbers = 1 to 100
3     val items = Pair("Hombre", "Luna")
4     println("El " + items.component1()+ " ya piso la " + items.component2())
5
6 }
7
8
9

PROBLEMAS  TERMINAL  COMENTARIOS  CONSOLA DE DEPURACIÓN  SALIDA

PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Pares> java -jar Pair.jar
El Hombre ya piso la Luna
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Pares>
```

equals (): Compara si dos pares son iguales.



```
ExMap.kt  Pair.kt  X
Pares > Pair.kt
1 fun main(){
2     val items = Pair("Hombre", "Luna")
3     val copy = Pair("Hombre", "Luna")
4     println("Los dos pares son iguales... ${items.equals(copy)}")
5     println("El " + items.component1()+ " ya piso la " + items.component2())
6     val newItems = Pair(5,"Musica")
7     println("Los dos pares son iguales... ${items.equals(newItems)}")
8 }
9
10
11
12
13
PROBLEMAS  TERMINAL  COMENTARIOS  CONSOLA DE DEPURACIÓN  SALIDA

PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Pares> java -jar Pair.jar
Los dos pares son iguales... true
El Hombre ya piso la Luna
Los dos pares son iguales... false
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Pares> 
```

7. Prácticas de estructuras de datos en Kotlin

a. Ejercicios prácticos para aplicar los conceptos aprendidos

1. Haga un arreglo que permita almacenar el valor de 5 números enteros y obtenga la sumatoria.

2. Haga una lista que permita almacenar el valor de 3 notas y obtenga el promedio de ellas.

3. Crea un nuevo conjunto que incluye solo los elementos que sean múltiplos de 3 a partir de otros dos conjuntos kotlin

4. Dado un mapa con nombres y edades, crea un nuevo mapa con los nombres en mayúsculas y las edades duplicadas.

5. Dado un par de valores enteros, crear una lista que contenga todos los números enteros entre el primer y el segundo valor del par.

b. Solución a los ejercicios prácticos

1. Ejercicio de arreglos.

```
ExArray.kt X ArrayOf.kt
Examples > ExArray.kt
1 // 1. Haga un arreglo que permita almacenar el valor de 5 números enteros y obtenga la sumatoria.
2
3 fun main() {
4     // declaramos una variable number para almacenar el numero ingresado por el usuario para guardarlo en el arreglo
5     // y para sumarlo en la variable addition
6     var number: Int
7     var addition = 0
8     var array = IntArray(5)
9     // creamos un ciclo para iterar sobre los índices del arreglo y preguntar al usuario por cada valor utilizando
10    la función readLine().
11    for (i in array.indices) {
12        println("Ingrese el numero en la posicion $i")
13        number = readLine()!!.toInt()
14        // cada dato es almacenado en el arreglo y se suma en la variable addition
15        array[i] = number
16        addition += number
17    }
18    // mostramos todos los elementos dentro del arreglo
19    println("Los numeros del arreglo son: ")
20    for (i in array) {
21        println(i)
22    }
23    // imprimimos la suma de los elementos
24    println("La suma de los elementos es de: $addition")
25 }
```

```
PROBLEMAS    TERMINAL    COMENTARIOS    CONSOLA DE DEPURACIÓN    SALIDA

Ingrese el numero en la posicion 0
5
Ingrese el numero en la posicion 1
4
Ingrese el numero en la posicion 2
6
Ingrese el numero en la posicion 3
5
Ingrese el numero en la posicion 4
4
Ingrese el numero en la posicion 3
4
Ingrese el numero en la posicion 4
5
Los numeros del arreglo son:
5
4
6
4
5
La suma de los elementos es de: 24
```

2. Haga una lista que permita almacenar el valor de 3 notas y obtenga el promedio de ellas.

```
ExArray.kt  ExList.kt  X  listas.kt
Examples > ExList.kt
1  // 2. Haga una lista que permita almacenar el valor de 3 notas y obtenga el promedio de ellas.
2  fun main(){
3      // declaracion de variables
4      // Declaramos una lista mutable de notas de tipo Double
5      val notes = mutableListOf<Double>()
6      var note: Double
7      var addition=0.0
8      var average: Double
9      // Pedimos al usuario que ingrese las 3 notas y las agregamos a la lista
10     for (i in 1..3) {
11         println("Ingrese la nota ${i}")
12         note= readLine()!!.toDouble()
13         if (note>5 || note<0){
14             println("Valor invalido")
15             break
16         }
17         else{
18             notes.add(note)
19             addition+= note
20         }
21     }
22     // imprimimos las notas
23     println(notes)
24     // Calculamos el promedio de las notas
25     average= notes.average()
26     // imprimimos el promedio
27     println("El promedio de sus notas es de: $average")
28 }
```

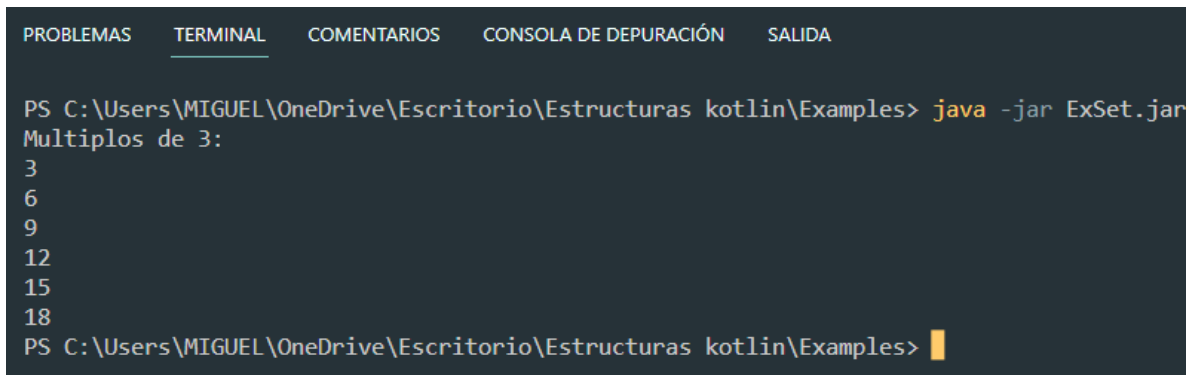
```
PROBLEMAS  TERMINAL  COMENTARIOS  CONSOLA DE DEPURACIÓN  SALIDA

PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Examples> java -jar ExList.jar
Ingrese la nota 1
4
Ingrese la nota 2
5
Ingrese la nota 3
3
[4.0, 5.0, 3.0]
El promedio de sus notas es de: 4.0
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Examples>
```

3. Crea un nuevo conjunto que incluye solo los elementos que sean múltiplos de 3 a partir de otros dos conjuntos kotlin



```
ExSet.kt  X  set.kt
Examples > ExSet.kt
1  fun main() {
2      // creamos dos conjuntos, uno de numeros pares y otro de impares
3      val peers = mutableSetOf(2,4,6,8,10,12,14,16,18,20)
4      val odd = mutableSetOf(1,3,5,7,9,11,13,15,17, 19)
5
6
7      //se utiliza la función "union" para combinar los dos conjuntos y crear uno nuevo que contenga
      // todos los elementos. A continuación, se utiliza la función "filter" para seleccionar solo los
      // elementos que sean múltiplos de 3, se utiliza la función "sorted" para organizar los numeros en
      // orden ascendente
8      val multiplesOfThree = peers.union(odd).filter{it % 3 ==0}.sorted()
9
10     // se recorre el conjunto utilizando un bucle "for" para imprimir cada elemento en la consola.
11     println("Multiplos de 3:")
12     for (element in multiplesOfThree) {
13         println(element)
14     }
15
16 }
```



```
PROBLEMAS  TERMINAL  COMENTARIOS  CONSOLA DE DEPURACIÓN  SALIDA
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Examples> java -jar ExSet.jar
Multiplos de 3:
3
6
9
12
15
18
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Examples>
```

4. Dado un mapa con nombres y edades, crea un nuevo mapa con los nombres en mayúsculas y las edades duplicadas.

```
ExMap.kt x Map.kt
Examples > ExMap.kt
1 fun main() {
2     //Creamos un mapa mutable de nombres y edades
3     val people = mutableMapOf("Natalia" to 20, "Mariana" to 19, "Leonel" to 34, "Miguel" to 17)
4     // Le preguntamos al usuario de que persona quiere saber la edad
5     print("De que persona deseas saber la edad: ")
6     // Almacenamos el nombre para utilizarlo como llave
7     val name = readline()
8     // guardamos la edad a traves de la llave en este caso el nombre, si el nombre no esta el resultado de age es null
9     val age = people[name]
10    // Validamos que la persona este registrada en el hasmap para poder brindar la informacion
11    if (age != null) {
12        println("La edad de $name es $age")
13    } else {
14        println("No se encontró el nombre $name en la lista")
15    }
16 }
```

```
PROBLEMAS  TERMINAL  COMENTARIOS  CONSOLA DE DEPURACIÓN  SALIDA

PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Examples> java -jar ExMap.jar
De que persona deseas saber la edad:
Miguel
La edad de Miguel es 17
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Examples> java -jar ExMap.jar
De que persona deseas saber la edad:
Margarita
No se encontró el nombre Margarita en la lista
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Examples>
```

5. Dado un par de valores enteros, crear una lista que contenga todos los números enteros entre el primer y el segundo valor del par

```
Pair.kt    ExPair.kt    X
Examples > ExPair.kt
1  // Dado un par de valores enteros, crear una lista que contenga todos los números enteros entre el primer y el
   // segundo valor del par.
2
3  fun main() {
4      // Creamos el par con ambos valores enteros, tambien la lista para almacenar los numeros
5      val pair = Pair(2, 19)
6      val numbers = mutableListOf<Int>()
7      // Creamos un ciclo for el cual empezara desde el primer elemento del par hasta el ultimo, para recorrer el
   // rango entre estos dos, dentro del ciclo almacenaremos los numeros en la lista
8      for (i in pair.first..pair.second) {
9          numbers.add(i)
10     }
11     // Imprimimos la lista para verificar que se hayan almacenado los numeros dentro del rango
12     println("Los numeros entre ${pair.component1()} y ${pair.component2()} son:")
13     println(numbers)
14 }
15
```

```
PROBLEMAS    TERMINAL    COMENTARIOS    CONSOLA DE DEPURACIÓN    SALIDA
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Examples> java -jar ExPair.jar
Los numeros entre 2 y 19 son:
[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
PS C:\Users\MIGUEL\OneDrive\Escritorio\Estructuras kotlin\Examples>
```