Set

Un set es una operación que regresa una serie de elementos únicos, es decir, que no se repiten. Por ejemplo, el alfabeto es un set de letras. Almacena elementos únicos, incluso con valores null.

- Para declarar un set inmutable usamos la función **setOf()**. Esto nos creará una instancia de un conjunto, en el cual no podremos realizar operaciones de escritura.
- Para realizar operaciones de escritura sobre un set lo creamos con la función **mutableSetOf()**

A continuación se muestran las operaciones más utilizadas:

```
val setEnteros = setOf(2,6,4,29,5)
val setVariado: Set<Any> = setOf(2,6,4,29,5,"ASD","EXY")

// Ciclo for con SET
for(elemento in setEnteros){
    println(elemento)
}

println(".....")

for(elemento in setVariado){
    println(elemento)
}

println(".....")

// Buscar elementos
println("Buscar 1 en setVariado")
println(setVariado.contains(1))
println("Buscar 'ASD' en setVariado")
println(setVariado.contains("ASD"))
println("Buscar set dentro de otro")
println(setVariado.containsAll(setEnteros))

println(".....")

// El SET contiene datos
println("El Set está vació")
println(setEnteros.isEmpty())
```

```
    println("El Set no está vació")
    println(setEnteros.isNotEmpty())

    println("..............")

    // Obtener lista a partir de la posición deseada
    val nuevoSetEnteros = setEnteros.drop(2)
    println(nuevoSetEnteros)

    println("Imprimir posición 1")
    println(nuevoSetEnteros.elementAt(1))

    println("Imprimir posición 5 -> Si no existe no retorna excepción, solo
imprime null")
    println(nuevoSetEnteros.elementAtOrNull(5))

    println(".......MutableSet.........")

    val setMutableEnteros = mutableSetOf(2,6,4,29,5)
    val setMutableVariado: MutableSet<Any> =
mutableSetOf(2,6,4,29,5,"ASD","EXY")

    println(setMutableEnteros)
    println(setMutableVariado)

    // Agregar
    setMutableEnteros.add(7)
    setMutableVariado.add(7)
    // Nota: Es posible agregar un set dentro de otro,
    // para eso tenemos que utilizar
    // setMutableVariado.addAll(setMutableEnteros)

    println(setMutableEnteros)
    println(setMutableVariado)

    // Remover
    setMutableEnteros.remove(7)
    setMutableVariado.remove(7)
    // Nota: Es posible remover un set dentro de otro,
    // para eso tenemos que utilizar
    // setMutableVariado.removeAll(setMutableEnteros)

    println(setMutableEnteros)
    println(setMutableVariado)

    // El resto de operaciones implementadas en la lista inmutable se
puede aplicar en las mutables
```

El código anterior puede probarse en el siguiente link

Map

El mapa (o diccionario) es un conjunto de pares clave-valor. Las claves son únicas y cada una de ellas se asigna exactamente a un valor. Los valores pueden ser duplicados. Los mapas son útiles para almacenar conexiones lógicas entre objetos, por ejemplo, la identificación de un empleado y su posición.

Para crear una colección Map inmutable o de solo lectura en Kotlin, usamos la función **mapOf()**. Creamos un mapa con esta función dándole una lista de pares. El primer valor es la clave, y el segundo es el valor. Llamar a esta función devuelve una interfaz tipo Kotlin Map.

Supongamos que queremos crear un mapa en el que asociemos el nombre de usuario con la edad, es decir, cada edad corresponde a un nombre de usuario. Recordando tipos de datos, es importante notar que el nombre de usuario es un String y que la edad es un Int, de modo que nuestro mapa quedaría de la siguiente forma.

```
val namesToAges: Map<String, Int> = mapOf("unser_one" to 20,
"user_two" to 23)
```

Observa que como definimos nuestro mapa, el primer elemento es un String y el segundo es un Int, por esto, cuando creamos la instancia de nuestro mapa con la función **mapOf()** asignamos un elemento Int a una clave String con la palabra reservada to.

Un punto importante es que cuando iteramos sobre un mapa, o sea, recorrer cada uno de sus elementos, podremos acceder tanto a la clave como al valor, un ciclo for nos permite hacerlo de la siguiente forma.

```
   val namesToAges: Map<String, Int> = mapOf("Andrés" to 20, "Carlos" to
23)

   for ((key, value) in namesToAges) {
      println("$key tiene $value años")
   }

   println(".....")
```

```
// Imprimir solo valores
for(key in namesToAges.keys){
   println(namesToAges[key])
}

println(".....")

// Imprimir solo llaves
for(key in namesToAges.keys){
   println(key)
}
```

Al igual que las listas (list) y los conjuntos (set) la función **mapOf()** nos crea un mapa inmutable. Si queremos hacer operaciones de escritura sobre un mapa tenemos que crearlo con la función **mutableMapOf()**.

A continuación se muestran las operaciones más utilizadas:

```
// Crear map que no sea genérico -> Nótese que no definimos los tipos de datos que utilizará
val myMap = mapOf(20 to "Andrés", 23 to "Carlos", "Ramirez" to "Andrés", "Ramírez" to 28)

   for(key in myMap.keys){
      println("$key = ${myMap.get(key)}")
   }

   println("Obtener valor de la llave 23")
   println(myMap.getValue(23))
   println(myMap.get(23))

   println("Existe la llave o valor 3")
   println(myMap.contains(3))
   println("Existe la llave o valor 20")
   println(myMap.contains(20))

   println("Existe la llave 23")
   println(myMap.containsKey(23))

   println("Existe el valor Andrés")
   println(myMap.containsValue("Andrés"))

   println("Retornar default si no encuentra la llave")
   println(myMap.getOrDefault(3, "Bedu"))
```

```
println("..............")

// Recorer elementos con iterator, a diferencia de los
// ciclos pasados, con éste obtenemos llave y valor en la
// variable elemento
for(elemento in myMap.iterator()){
   println("key = ${elemento.key} value = ${elemento.value}")
}

println(".....Mutable.....")

val users = mutableMapOf(15 to "Luis", 20 to "Juan", 32 to "Manuel")

println(users)

users.put(28, "Andrés")
users.putAll(setOf(4 to "Isaac", 5 to "Lorena"))

println(users)

users.remove(15)
users.keys.remove(20)
users.values.remove("Manuel")

println(users)
```

El código anterior puede probarse en el siguiente link

Los siguientes enlaces muestran ejemplos y documentación oficial:

Documentación Set

Documentación Map

Documentación List