Estructuras de datos en Kotlin.

Las estructuras de datos como en otros lenguajes nos permiten organizar y almacenar datos en la memoria del programa y gracias a esto nos dan la manera de acceder a información de forma mas eficiente y efectiva, y las estructuras de control más utilizadas son:

* Arreglos.
* Listas.
* Conjuntos.
* Mapas.
* Pares

**Algunas ventajas de utilizar las estructuras de datos en Kotlin son:**

1. **Organizar datos:** Nos permiten almacenar datos y acceder más fácil a estos.
2. **Eficiencia en el tiempo de ejecución:** Utilizando las estructuras de datos adecuadas, se pueden optimizar las búsqueda, inserción y eliminación de elementos.
3. **Facilidad de mantenimiento:** Hacen que el código sea más fácil de entender, depurar y mantener en el futuro.
4. **Reducción de errores:** pueden ayudar a evitar errores comunes de programación, como la corrupción de datos o la pérdida de elementos.

**Diferencias entre las estructuras de control de Java y Kotlin**

1. **Sintaxis más concisa:** La sintaxis de Kotlin es más concisa que la de Java, lo que facilita la lectura y escritura de código. Por ejemplo, la sintaxis de Kotlin para crear listas y mapas es más limpia y fácil de leer que la de Java.
2. Capacidad de nulidad: Kotlin tiene un sistema de tipos de nulidad que le permite especificar si las variables pueden contener valores nulos. Por lo tanto, las estructuras de datos de Kotlin son más seguras para evitar excepciones de puntero nulo. Por otro lado, Java no tiene un sistema de tipo anulable, lo que facilita que las estructuras de datos arrojen excepciones de puntero nulo.
3. **Funciones de extensión:** Kotlin le permite definir funciones de extensión que se pueden agregar a las estructuras de datos existentes sin heredar de las clases base ni crear subclases. Esto hace que las estructuras de datos sean más flexibles y extensibles que Java.
4. **Colecciones inmutables predeterminadas:** en Kotlin, las colecciones inmutables son el tipo predeterminado de estructura de datos, lo que brinda una mejor seguridad de concurrencia y evita la modificación accidental de datos. En Java, las colecciones mutables son el tipo predeterminado, lo que puede provocar problemas de simultaneidad y dificultar la implementación de estructuras de datos inmutables.

**ARREGOS O ARRAYS**

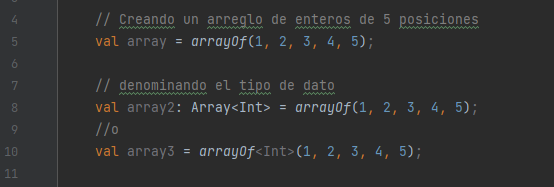
1. ¿Qué es un arreglo?

En Kotlin, una matriz es una estructura de datos utilizada para almacenar una colección de elementos del mismo tipo de datos. Se define como una secuencia de elementos del mismo tipo, almacenados en ubicaciones de memoria contiguas y accesibles mediante índices numéricos.

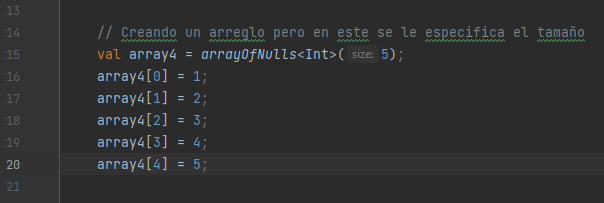
1. Como se crean los arreglos en Kotlin

En kotlin podemos crear arreglos de las siguientes formas

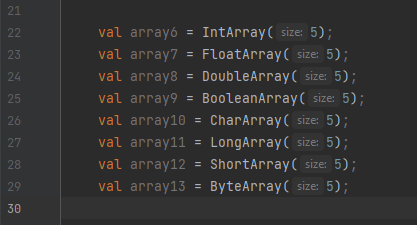
* 1. Con “arrayOf” con esta función lo que hacemos es crear un arreglo con unos datos específicos ya que nos exige llenarlo en ese mismo instante



* 1. Con “arrayOfNulls” con esta función creamos ya un arreglo con el tamaño definido y en el cual se nos exige definirle el tipo de dato ya que como se dijo anteriormente estos solo almacenan un tipo de dato

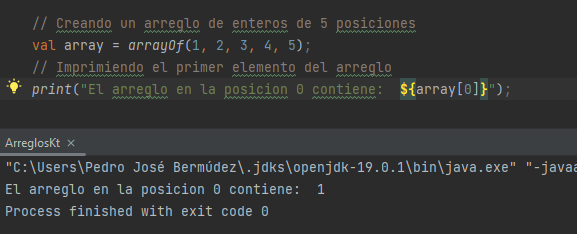


* 1. También se pueden crear arreglos con las siguientes funciones en las cuales está implícito el tipo de dato que este debe de almacenar, con estos se crea de las mismas maneras que las dos anteriores funciones solo que estos son menos usados



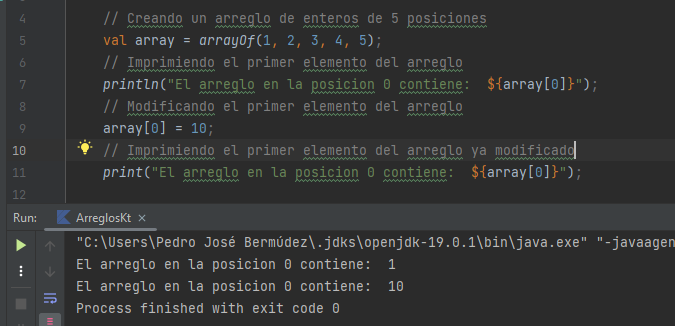
1. Acceder a los elementos de un arreglo

Para acceder a los elementos de un arreglo lo podemos hacer mediante su índice o con “get”



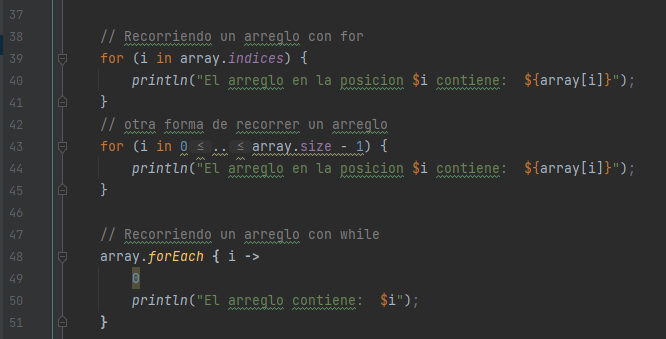
1. Modificando los elementos de un arreglo

Para modificar algún elemento del arreglo lo hacemos de la misma manera que para acceder a ellos que seria con el índice o con “set”



1. Recorrer un arreglo

Para recorrer un arreglo una de las mejores maneras es con arreglo como lo podría el “for” y “forEach”



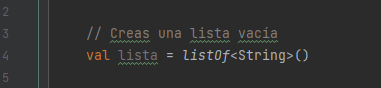
1. Funciones útiles para trabajar con arreglos en Kotlin
   1. “size”: Con esta función nos dice el tamaño del arreglo
   2. “get (index)”: Con esta función nos muestra el valor que está en ese índice
   3. “set (index, value)” con esta función le decimos en que índice deseamos cambiar por el valor que le mandamos
   4. “indexOf (element)”: Con esta función lo que haces es mostrarnos el índice del valor que le mandemos
   5. “contains(element)”: Con esta función nos devuelve un booleano, “true” si contine el valor o por lo contrario “false”
   6. “sorted()”: Esta función nos hace una copia del arreglo original

**Listas o list**

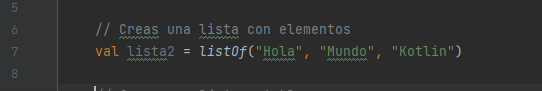
1. ¿Qué es una lista en kotlin?

Una lista es una colección ordenada de elementos a los que se puede acceder a través de un índice. Similares a los arreglos, pero a diferencia de los arreglos, las listas son estructuras de datos mutables.

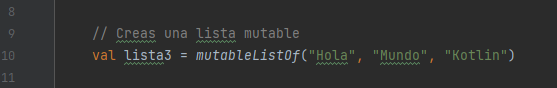
1. Las listas en Kotlin se crean de las siguientes formas
   1. Crear la lista vacía con “listOf”, toca definirle el tipo de dato por la convención de que deben de contener un solo tipo de dato y una vez creada no se podrá modificar



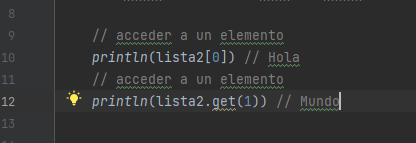
* 1. Crear una lista con elementos con “listOf” y una vez la creemos no se podrá modificar

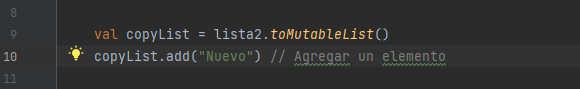


* 1. Creando una lista con “mutableListOf” el cual nos permite modificar la lista una vez la creemos



1. Acceder a los elementos de una lista lo hacemos con el índice o con “get”



1. Modificando los elementos de una lista
   1. Si es inmutable para agregar elementos lo haremos una copia de esta con “mutableListOf” y después se le realizaría el “add”En el cual básicamente lo que hacemos es decirle que a la lista copiaremos la misma lista anterior y la aremos mutable
   2. Si es mutable solo será

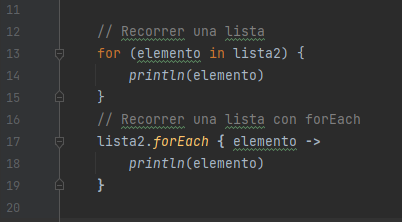
Texto

Descripción generada automáticamente

* 1. El resto de las modificaciones son iguales para las dos
     1. Borrar un valor ya sea por el mismo valor o su índice con “remove” o “remodeAt” Interfaz de usuario gráfica, Texto

        Descripción generada automáticamente

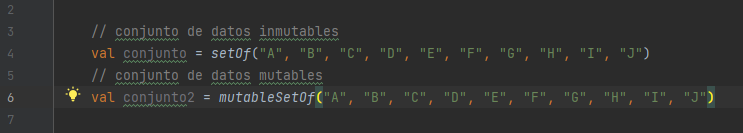
1. Recorriendo un arreglo con “for” y “forEach”



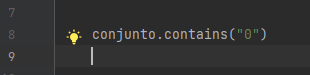
1. Funciones útiles para trabajar con listas en Kotlin
   1. “get (index)”: Con esta función nos muestra el valor que está en ese índice
   2. “size”: Con esta función nos dice el tamaño del arreglo
   3. “set (index, value)” con esta función le decimos en que índice deseamos cambiar por el valor que le mandamos
   4. “indexOf (element)”: Con esta función lo que haces es mostrarnos el índice del valor que le mandemos
   5. “contains(element)”: Con esta función nos devuelve un booleano, “true” si contine el valor o por lo contrario “false”

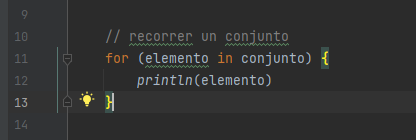
**Conjuntos**

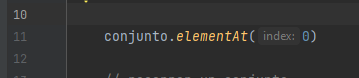
1. que son: Los conjuntos (sets) son una colección de elementos que no están duplicados y no tienen un orden específico. Los conjuntos se utilizan para almacenar elementos únicos, lo que los hace útiles para tareas como eliminar elementos duplicados de una lista.
2. Crear de conjuntos en Kotlin: los conjuntos al igual que las listas son inmutables al ser creados así que si se desea adicionar otro elemento se le debe de agregar la propiedad mutable que lo que hace es volver este mutable



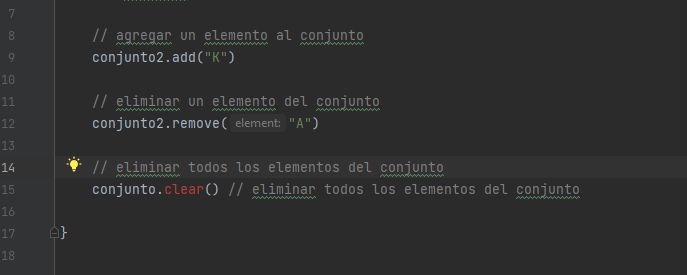
1. Accediendo a los elementos de un conjunto: Para acceder a los elementos de un conjunto seria directamente recorriéndolo todo o por defecto consultando si contine algún elemento, con sus índices



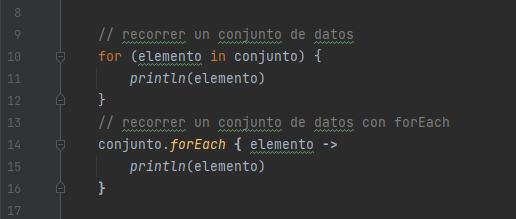




1. Modificando los elementos de un conjunto: Para modificar los datos almacenados en un conjunto usamos



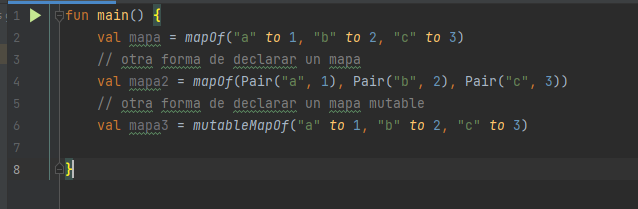
1. Recorrer un conjunto: la forma mas común de recorrerlo es con un “for” o “forEacht”



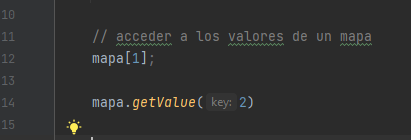
1. Funciones útiles para trabajar con conjuntos en Kotlin
   1. “union ()”: Se usa para unir dos conjuntos
   2. “intersect()”: Se usa para hacer un conjunto pero solo de los valores que estén presentes en los dos conjuntos
   3. subtract(): Retorna un nuevo conjunto que contiene solo los elementos que están presentes en el primer conjunto original, pero no en el segundo conjunto original.
   4. “contains()”: Nos devuelve True si contiene el valor o false si es lo contrario
   5. “isEmpty ()”: Nos devuelve true si este está vacio o false si es lo contrario

**Mapas**

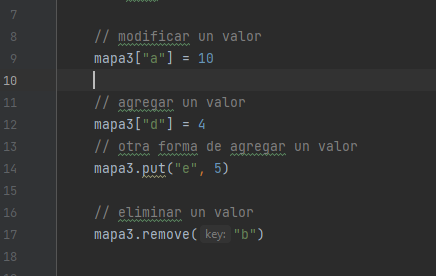
1. Que son los mapas: Un mapa es una estructura de datos que puede almacenar pares de clave-valor. Cada clave es única y se utiliza para acceder al valor correspondiente.
2. Para crear un mapa en Kotlin usamos “mapOf” y con esta sentencia ya solo sería pasarle los valores y al igual que los otras estructuras estos también son mutables e inmutables



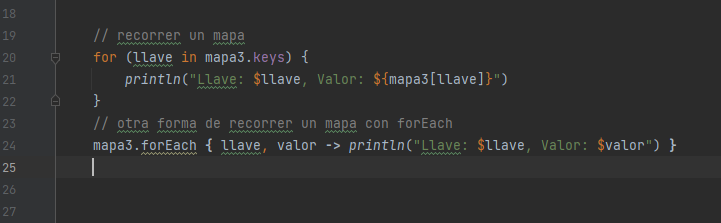
1. Para acceder a los a los elementos almacenados gracias a las lleves



1. Para modificar elementos de un mapa se usa “put” para agregar si esta es mutable, “remove” para eliminar y ya para cambiar el valor de un elemento lo que debemos hacer es renombrarlo



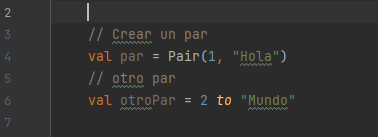
1. Para recorrer el mapa lo podemos hacer con las siguientes estructuras



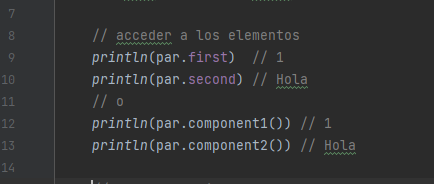
1. Funciones útiles para trabajar con mapas en Kotlin
   1. “mapOf”: Nos permite crear un nuevo mapa y este es inmutable
   2. “mutableMapOf”: También podemos crear un mapa, pero este si será mutable
   3. “put”: Podemos agregar datos a un mapa pero este debe de ser mutable
   4. “get”: Nos trae el valor de la llave que le entregemos
   5. “containsKey”: Nos devuelve un boolean dependiendo si la llave esta o no en el mapa
   6. “containsValue”: No devuelve un boolean dependiendo si el valor está en el mapa
   7. “keys”: Esta función nos devuelve todas las llaves del mapa
   8. “values”: Nos trae todos los valores que están el mapa

**Pares**

1. En Kotlin, un par es una estructura de datos que representa un conjunto de dos valores. Los pares se pueden usar para unir dos valores y pasarlos como un solo objeto.
2. Para crear un par lo hacemos con “pair” o dando los dos valores



1. Para acceder a los elementos de un par lo podemos hacer con las siguientes funciones



1. Para modificar los elementos de un par lo hacemos con una copia en la cual le pasamos el nuevo valor

Texto

Descripción generada automáticamente

1. Para recorrer es

Texto

Descripción generada automáticamente

1. Funciones útiles para trabajar con pares en Kotlin
   1. “to”: Con esta función nos indica que se van a adjuntar dos elementos
   2. “first, second”: Nos ayuda a traer los elementos del par, siendo correspondiente a la pociones que estos nos indican
   3. “component1, component2”: Hace lo mismo que las dos funciones anteriores
   4. “toList”: Se usa para convertir el par en una lista
   5. “unzip”: Se utiliza para separa un conjunto de pares en ya solo pares