Juan Pablo Ramírez Osorio 2469285

Taller Estructuras de Datos en Kotlin

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

las estructuras de datos son aquellas que nos permiten, como desarrolladores, organizar la información de manera eficiente, y en definitiva diseñar la solución correcta para un determinado problema.

Ventajas de utilizar estructuras de datos en Kotlin

Las ventajas de utilizar estructuras de datos en Kotlin son muy similares a las ventajas de utilizar estructuras de datos en cualquier otro lenguaje de programación. Algunas de las ventajas específicas de utilizar estructuras de datos en Kotlin incluyen:

1. Kotlin cuenta con estructuras de datos integradas: Kotlin proporciona estructuras de datos integradas como listas, conjuntos, mapas, arreglos y otras estructuras que son muy fáciles de usar y se integran bien con el resto del lenguaje.
2. Kotlin es un lenguaje conciso: Kotlin está diseñado para ser un lenguaje conciso, lo que significa que se pueden escribir estructuras de datos complejas con menos líneas de código que en otros lenguajes de programación.
3. Kotlin es un lenguaje seguro: Kotlin es un lenguaje seguro que utiliza el sistema de tipos para evitar errores comunes, como la asignación de valores incorrectos a una estructura de datos. Esto hace que sea más fácil trabajar con estructuras de datos sin tener que preocuparse por errores inesperados.
4. Kotlin es compatible con Java: Kotlin es compatible con Java, lo que significa que se pueden utilizar estructuras de datos de Java en un programa Kotlin y viceversa. Esto proporciona a los programadores una amplia selección de estructuras de datos para elegir.

Diferencias entre las estructuras de datos en Kotlin y Java

1. Sintaxis simplificada: Kotlin cuenta con una sintaxis más simplificada que Java, lo que hace que sea más fácil escribir y trabajar con estructuras de datos. Por ejemplo, en Kotlin no es necesario escribir la palabra clave "new" al crear una instancia de una estructura de datos.
2. Tipos de datos inmutables: En Kotlin, muchos de los tipos de datos integrados son inmutables por defecto, lo que significa que no se pueden modificar una vez creados. Esto hace que sea más seguro trabajar con estructuras de datos y evita errores comunes de programación.
3. Sintaxis simplificada: Kotlin cuenta con una sintaxis más simplificada que Java, lo que hace que sea más fácil escribir y trabajar con estructuras de datos. Por ejemplo, en Kotlin no es necesario escribir la palabra clave "new" al crear una instancia de una estructura de datos.

Arreglos en Kotlin

1. ¿Qué es un arreglo?

un arreglo es una estructura de datos que almacena una colección de elementos del mismo tipo, en la que cada elemento se identifica por un índice numérico. En otras palabras, un arreglo es una colección ordenada de elementos que se pueden acceder mediante un índice.

Creación de arreglos en Kotlin

Crear un arreglo de valores predefinidos:

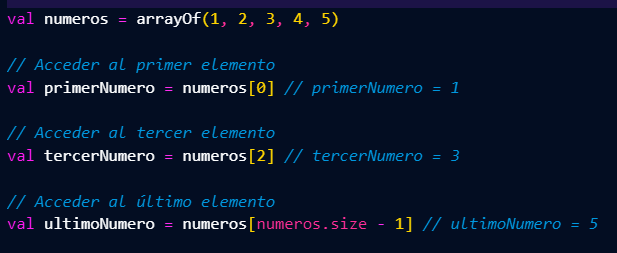


Crear un arreglo vacío con una longitud determinada:



1. Accediendo a los elementos de un arreglo

Para acceder a los elementos de un arreglo en Kotlin, se utiliza el operador de indexación [] seguido del índice del elemento deseado. El índice del primer elemento es 0 y el índice del último elemento es la longitud del arreglo menos uno.



También se pueden acceder a los elementos de un arreglo utilizando un ciclo for:

Interfaz de usuario gráfica

Descripción generada automáticamente

* Modificando los elementos de un arreglo en kotlin

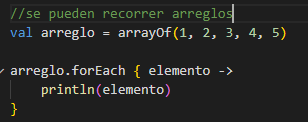
Para modificar los elementos de un arreglo en Kotlin, se utiliza el operador de indexación [] seguido del índice del elemento que se desea modificar, y luego se asigna un nuevo valor al elemento.

Texto

Descripción generada automáticamente

e. Recorriendo un arreglo

En Kotlin, se pueden recorrer arreglo utilizando diferentes formas



f. Funciones útiles para trabajar con arreglos en Kotlin **indexOf():** Devuelve el índice del primer elemento en el arreglo que es igual al elemento especificado.

**sort()**: Ordena los elementos del arreglo en orden ascendente.

**filter:** Esta función se utiliza para filtrar elementos de un arreglo que cumplan cierta condición.

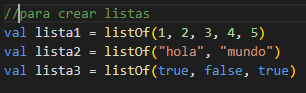
**map:** Esta función se utiliza para aplicar una transformación a cada elemento de un arreglo.

3. Listas en Kotlin

a. ¿Qué es una lista?

En Kotlin, una lista es una estructura de datos que permite almacenar una colección ordenada de elementos del mismo tipo. Los elementos de una lista pueden ser de cualquier tipo de datos, como enteros, cadenas de texto, objetos, entre otros.

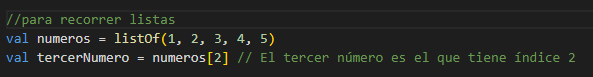
b. Creación de listas en KotlinEn Kotlin, se puede crear una lista utilizando la función listOf(), la cual recibe como argumento una serie de elementos separados por comas y devuelve una lista inmutable (es decir, una lista que no se puede modificar después de su creación). Por ejemplo:



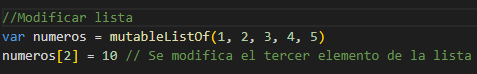
c. Accediendo a los elementos de una lista

En Kotlin, se puede acceder a los elementos de una lista utilizando su índice, que es un número entero que representa la posición del elemento en la lista. El primer elemento de la lista tiene un índice de 0, el segundo tiene un índice de 1 y así sucesivamente.

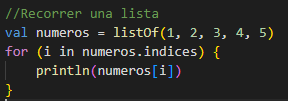
Para acceder a un elemento específico de la lista, se puede utilizar la sintaxis de corchetes [ ] seguida del índice del elemento. Por ejemplo, si se tiene una lista de enteros llamada numeros, se puede acceder al tercer elemento de la lista de la siguiente manera:



d. Modificando los elementos de una listaEn Kotlin, se pueden modificar los elementos de una lista utilizando la sintaxis de corchetes [ ] seguida del índice del elemento a modificar. Por ejemplo, si se tiene una lista de enteros llamada numeros, se puede modificar el tercer elemento de la lista de la siguiente manera:



e. Recorriendo una listaEn Kotlin, se pueden recorrer los elementos de una lista utilizando diferentes tipos de bucles y funciones. Algunas de las opciones más comunes son:



f. Funciones útiles para trabajar con listas en Kotlin-En Kotlin, existen diversas funciones útiles para trabajar con listas. Algunas de las más comunes son:

-size: devuelve la cantidad de elementos de la lista.

contains: devuelve true si la lista contiene un elemento específico, o false en caso contrario.

-indexOf: devuelve el índice del primer elemento que coincide con un valor específico, o -1 si no se encuentra.

-lastIndexOf: devuelve el índice del último elemento que coincide con un valor específico, o -1 si no se encuentra.

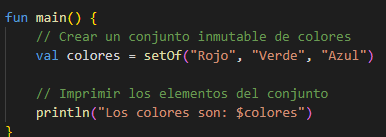
4-Conjuntos en Kotlin

a. ¿Qué es un conjunto?

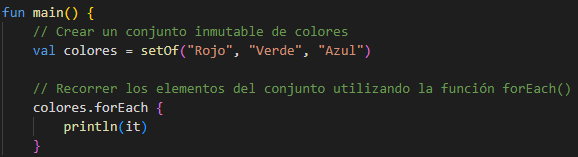
En Kotlin, un conjunto (también conocido como set) es una colección de elementos que no tiene un orden definido y no permite elementos duplicados. Esto significa que no se puede garantizar el orden en que se almacenan los elementos en el conjunto y que cada elemento que se agregue al conjunto debe ser único. Los conjuntos se definen con la interfaz Set y pueden ser mutables (MutableSet) o inmutables (Set).

b. Creación de conjuntos en Kotlin

En Kotlin, se pueden crear conjuntos usando la interfaz Set y sus implementaciones, como HashSet o LinkedHashSet (en el caso de conjuntos mutables) o setOf o linkedSetOf (en el caso de conjuntos inmutables).

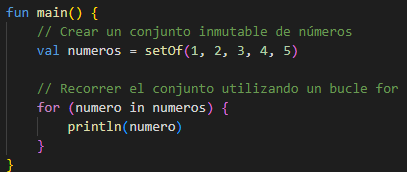


c. Accediendo a los elementos de un conjunto Para acceder a un conjunto en kotlin se utiliza la función ‘contains()’.



d. Modificando los elementos de un conjunto En Kotlin, los conjuntos son estructuras de datos inmutables, lo que significa que no se pueden modificar los elementos existentes de un conjunto. Sin embargo, se pueden crear nuevos conjuntos a partir de uno existente que contengan elementos adicionales o eliminados.

e. Recorriendo un conjunto



f. Funciones útiles para trabajar con conjuntos en KotlinEn Kotlin, existen varias funciones útiles para trabajar con conjuntos. A continuación, se describen algunas de ellas:

add(elemento: E): agrega un elemento al conjunto.

addAll(coleccion: Collection<E>): agrega una colección de elementos al conjunto.

remove(elemento: E): elimina un elemento del conjunto.

removeAll(coleccion: Collection<E>): elimina una colección de elementos del conjunto.

clear(): elimina todos los elementos del conjunto.

5. Mapas en Kotlin

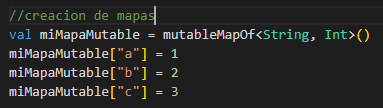
a. ¿Qué es un mapa?

En Kotlin, un mapa es una estructura de datos que almacena pares de elementos (clave-valor), donde cada clave es única y se utiliza para acceder a su valor correspondiente. Es similar a un diccionario en otros lenguajes de programación.

b. Creación de mapas en Kotlin

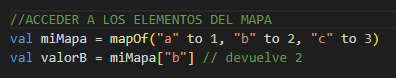
En Kotlin, se pueden crear mapas utilizando la clase Map y sus diferentes implementaciones, como HashMap, TreeMap, LinkedHashMap, entre otras.

La forma más común de crear un mapa es utilizando la función mapOf, que crea un mapa inmutable con los pares clave-valor proporcionados como argumentos. Por ejemplo:



Accediendo a los elementos de un mapaPara acceder a los elementos de un mapa en Kotlin, se utiliza la clave correspondiente. Dado que las claves son únicas, se puede acceder al valor correspondiente de manera eficiente.

Para acceder al valor de un elemento del mapa, se utiliza el operador de indexación [] con la clave correspondiente. Por ejemplo:



d. Modificando los elementos de un mapa

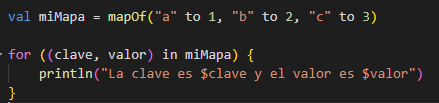
En Kotlin, los mapas inmutables no se pueden modificar después de su creación. En cambio, para modificar un mapa, se debe utilizar una implementación mutable como MutableMap.

Para modificar un elemento existente en un mapa mutable, se utiliza el operador de indexación [] para actualizar el valor correspondiente a una clave. Por ejemplo:



e. Recorriendo un mapa

Para recorrer un mapa en Kotlin, se puede utilizar un bucle for con la sintaxis for ((clave, valor) in mapa). Esto permite acceder tanto a la clave como al valor de cada elemento del mapa.



f. Funciones útiles para trabajar con mapas en Kotlin

mapOf(): crea un mapa inmutable a partir de una lista de pares clave-valor.

mutableMapOf(): crea un mapa mutable vacío.

put(): agrega un nuevo elemento al mapa.

6. Pares en Kotlin

a. ¿Qué es un par?

En Kotlin, un par es una estructura de datos que contiene dos elementos. Es similar a una tupla en otros lenguajes de programación. Cada elemento en el par puede ser de un tipo diferente. Los pares se utilizan a menudo para devolver múltiples valores de una función o para representar un par de elementos relacionados. En Kotlin, los pares son inmutables, lo que significa que no se pueden cambiar después de su creación.

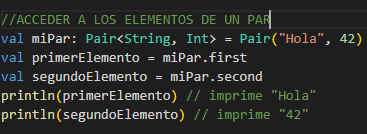
b. Creación de pares en Kotlin

En Kotlin, se puede crear un par utilizando la función Pair() o el operador to. La función Pair() toma dos argumentos y devuelve un objeto Pair que contiene ambos valores. El operador to es una forma abreviada de llamar a Pair().

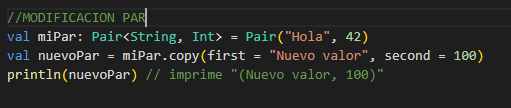


c. Accediendo a los elementos de un par

Para acceder a los elementos de un par en Kotlin, se puede utilizar la función first y second que devuelven el primer y segundo elemento del par, respectivamente.

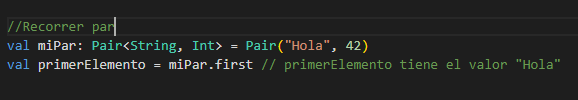


d. Modificando los elementos de un parLos pares en Kotlin son inmutables, lo que significa que una vez creados, sus valores no se pueden modificar. Si es necesario modificar los valores de un par, se debe crear un nuevo par con los nuevos valores.



e. Recorriendo un par

Los pares en Kotlin no se recorren, ya que son una estructura de datos que solo contiene dos elementos. En su lugar, se accede a los elementos individuales del par utilizando la sintaxis de punto o la desestructuración de variables.



f. Funciones útiles para trabajar con pares en Kotlin

to: esta función se utiliza para crear un par a partir de dos valores. Por ejemplo:



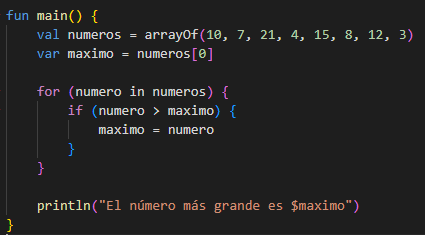
first y second: estas propiedades se utilizan para acceder al primer y segundo elemento de un par, respectivamente

equals: esta función se utiliza para comparar dos pares y determinar si son iguales. Dos pares son iguales si tienen los mismos elementos en el mismo orden.

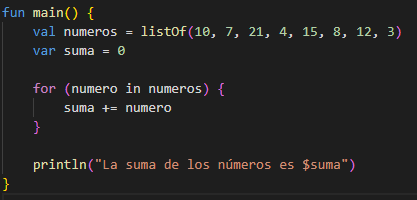
7- Prácticas de estructuras de datos en Kotlin

a. Ejercicios prácticos para aplicar los conceptos aprendidos

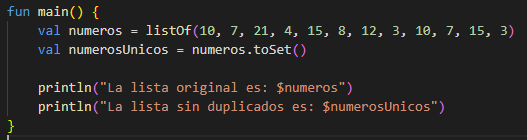
encontrar el número más grande en ese arreglo



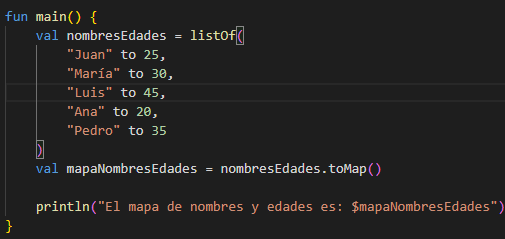
calcular la suma de todos los números enteros



tenemos una lista de números enteros y necesitamos eliminar los duplicados. Podemos usar un conjunto para lograrlo, ya que los conjuntos no permiten elementos duplicados.



tenemos una lista de nombres de personas y sus edades en años, y necesitamos crear un mapa que muestre los nombres de las personas como claves y sus edades como valores.



encontrar todos los números pares en una lista de enteros:

