1. Introducción a las estructuras de datos en Kotlin

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

**R=/** La estructura de datos es idéntica en concepto a una pila física de objetos. Cuando agregue un artículo a una pila, colóquelo encima de la pila. Cuando retire un elemento de una pila, siempre retire el elemento superior. Las pilas son útiles y también muy simples. El objetivo principal de crear una pila es hacer cumplir la forma en que accede a sus datos. Sirven para almacenar varios valores, generalmente del mismo tipo de datos, de manera organizada.

**b.** Ventajas de utilizar estructuras de datos en Kotlin

**R=/** Nos permite llevar un control ordenado de los datos que ingresamos, nos deja conocer su orden y manipular los datos fácilmente.

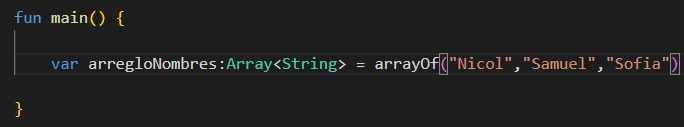
**c.** Diferencias entre las estructuras de datos en Kotlin y Java

**R=/** Ambos tienen casi la misma velocidad de codificación, pero Java requiere escribir más código. Sin embargo, se dedica menos tiempo a pensar la solución con Java. Kotlin tiene construcciones más concisas, por lo que requiere escribir menos código. Sin embargo, se tarda más tiempo en encontrar la solución a una tarea.

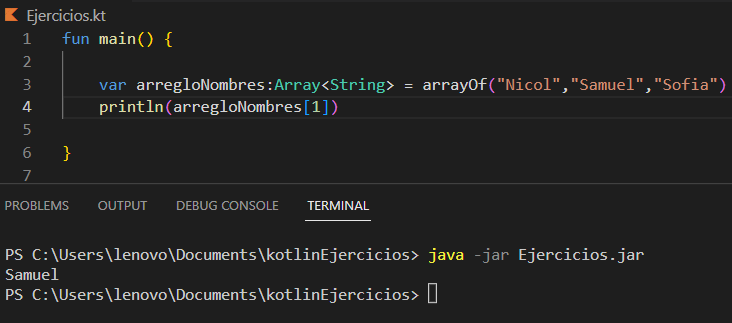
1. Arreglos en Kotlin
2. ¿Qué es un arreglo?

**R=/** Un array es la forma más sencilla de agrupar un número arbitrario de valores en tus programas. Al igual que una agrupación de paneles solares se denomina array solar, o del mismo modo en que aprender sobre Kotlin abre un conjunto (array) de posibilidades para tu carrera de programación, un Array representa más de un valor. Específicamente, un array es una secuencia de valores que tienen el mismo tipo de datos.

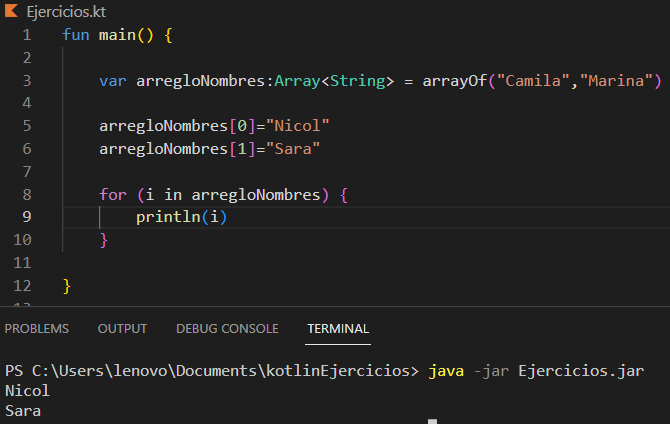
1. Creación de arreglos en Kotlin



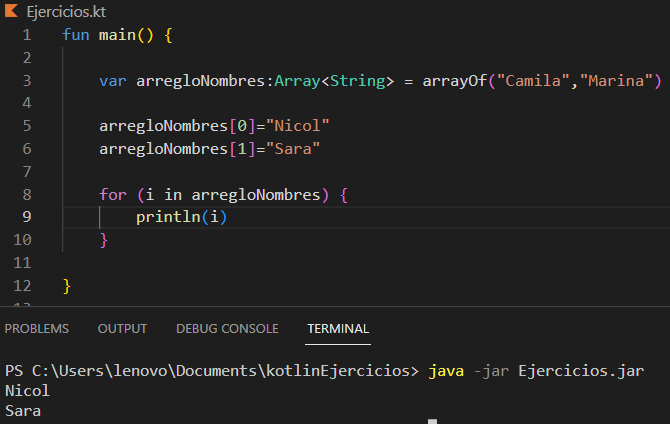
1. Accediendo a los elementos de un arreglo



1. Modificando los elementos de un arreglo



1. Recorriendo un arreglo



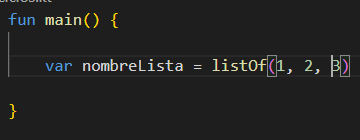
1. Funciones útiles para trabajar con arreglos en Kotlin

* get()
* set()
* arrayOf()

1. Listas en Kotlin
2. ¿Qué es una lista?

**R=/** Una lista es una colección redimensionable y ordenada que, por lo general, se implementa como un array que puede cambiar de tamaño. Cuando el array alcanza su capacidad máxima y tratas de insertar un nuevo elemento, el array se copia en un nuevo array más grande.

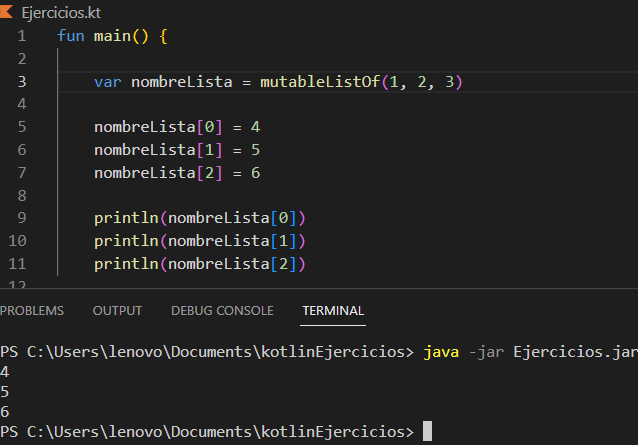
1. Creación de listas en Kotlin



1. Accediendo a los elementos de una lista



1. Modificando los elementos de una lista



1. Recorriendo una lista



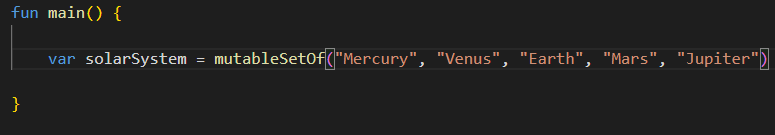
**f.** Funciones útiles para trabajar con listas en Kotlin

* setOf()
* listOf()
* add()
* addAll()
* celar()
* set()
* mutableListOf()
* remove()
* removeAt()
* contains()

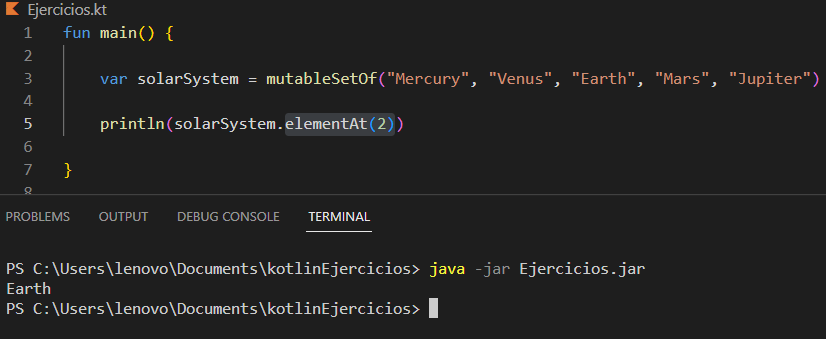
1. Conjuntos en Kotlin
2. ¿Qué es un conjunto?

**R=/** Un conjunto es una colección que no tiene un orden específico y no permite valores duplicados.

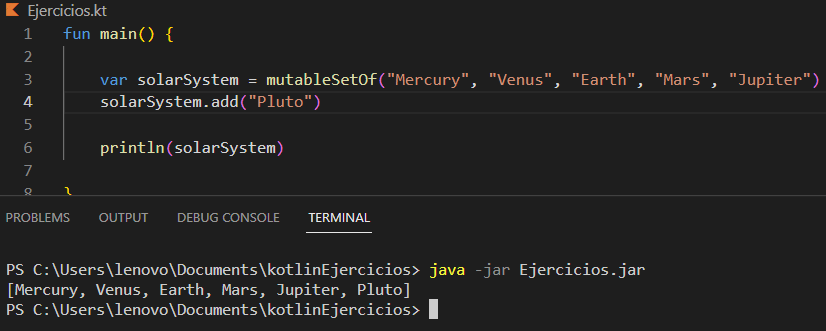
1. Creación de conjuntos en Kotlin



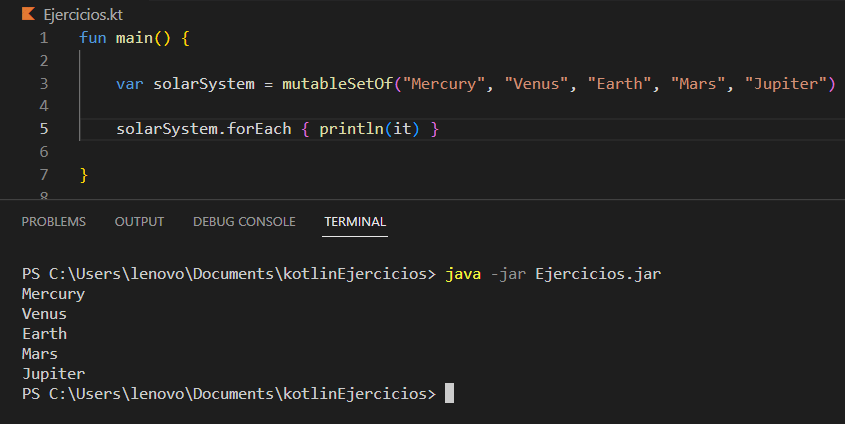
1. Accediendo a los elementos de un conjunto



1. Modificando los elementos de un conjunto



1. Recorriendo un conjunto



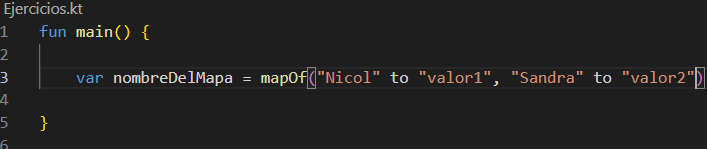
**f.** Funciones útiles para trabajar con conjuntos en Kotlin

* size
* indexOf()
* hashCode()
* add()
* contains()
* containsAll()
* elementAt()
* elementAtOrNull()
* addAll()
* clear()
* first()
* last()
* sum()
* average()
* random()

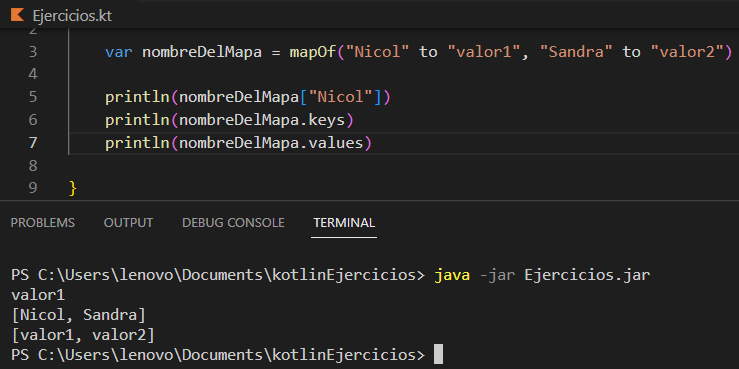
1. Mapas en Kotlin
2. ¿Qué es un mapa?

**R=/** Un Map es una colección que consta de claves y valores. Se llama un mapa porque las claves únicas se asignan a otros valores. Una clave y su valor complementario suelen llamarse key-value pair.

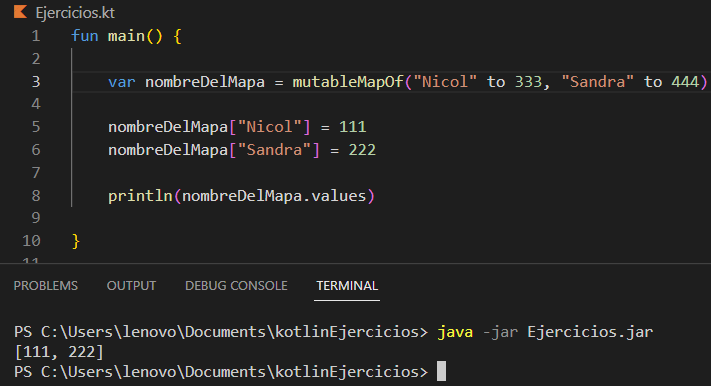
1. Creación de mapas en Kotlin



1. Accediendo a los elementos de un mapa



1. Modificando los elementos de un mapa



1. Recorriendo un mapa



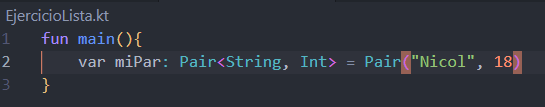
**f.** Funciones útiles para trabajar con mapas en Kotlin

* values
* keys
* put()
* remove()
* size
* containsKeys()
* containsValue()
* get()

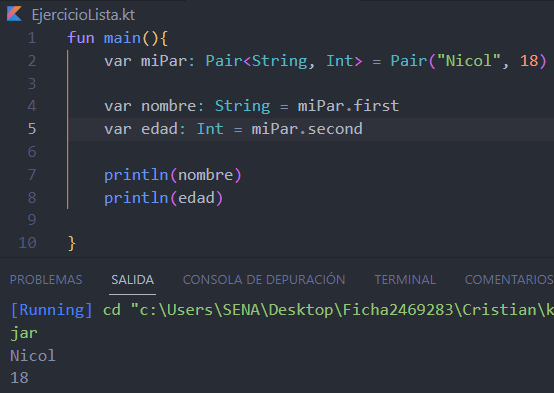
1. Pares en Kotlin
2. ¿Qué es un par?

**R=/** es un tipo de dato que permite almacenar dos valores de tipos distintos en una sola variable. Es una clase que contiene dos propiedades, first y second, que pueden contener cualquier tipo de dato.

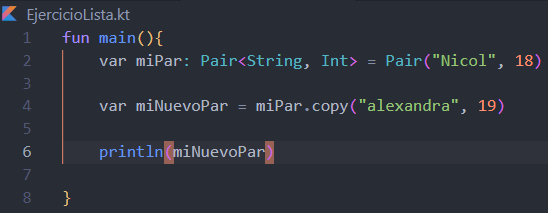
1. Creación de pares en Kotlin



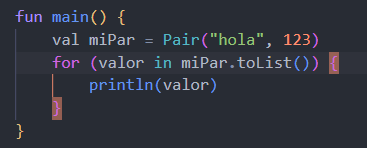
1. Accediendo a los elementos de un par



1. Modificando los elementos de un par



1. Recorriendo un par

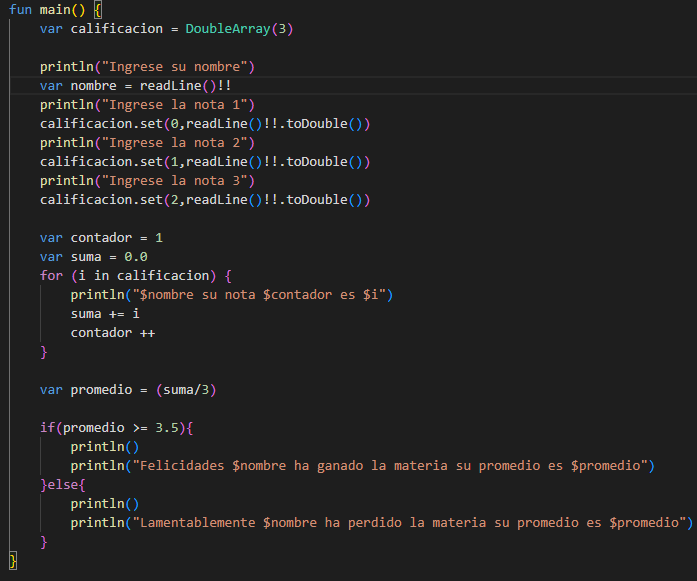


**f.** Funciones útiles para trabajar con pares en Kotlin

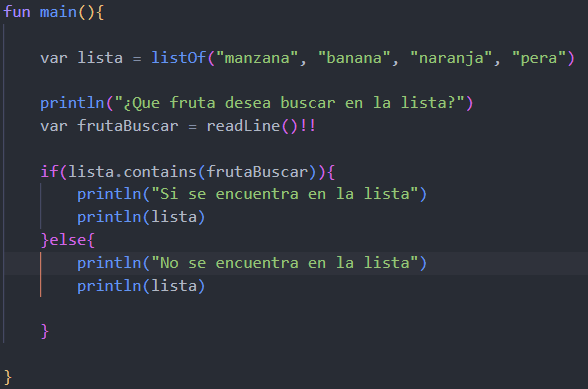
* Copy()
* First()
* Second()

1. Prácticas de estructuras de datos en Kotlin
2. Ejercicios prácticos para aplicar los conceptos aprendidos

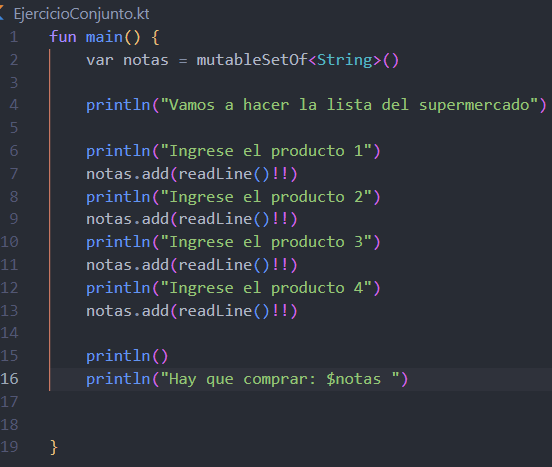
**Arreglos**

****

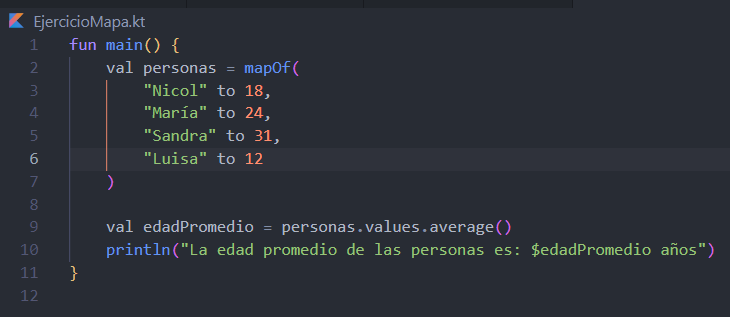
**Listas**

****

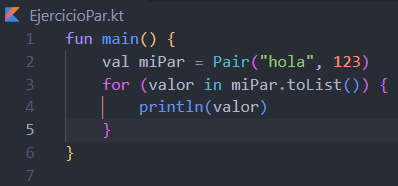
**Conjuntos**

****

**Mapas**

****

**Pares**

****