**Taller Estructuras de Datos en Kotlin**

**Presentado por:**

**Julian David Parra Ospina**

**Aprendiz**

**Ficha:**

**2469181**

**Análisis y Desarrollo de Sistemas de Información**

**Sena Centro de Comercio y turismo**

**Sede Galán**

**2022**

**Taller Estructuras de Datos en Kotlin**

**1.Introduccion a las estructuras de datos en Kotlin:**

1. **¿Qué son las estructuras de datos y para que se utilizan?**

**Respuesta:** Las estructuras de datos son aquellas que nos permiten gestionar de manera organizada cierta cantidad de datos y se utilizan para guardar estos datos en la memoria de la aplicación.

1. **¿Ventajas de utilizar estructuras de datos en Kotlin?**

**Respuesta:** las ventajas de utilizar las estructuras de datos en kotlin son la eficiencia con las que el lenguaje de programación los asimila, la facilidad de uso ya que la sintaxis en simple y concisa y la compatibilidad que tiene con Java.

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

**Respuesta:** Las diferencias entre las estructuras de control de kotlin y java es que en java existen las tuplas y en kotlin no existen, pero de resto todas las estructuras de datos son sus muy parecidas.

**2. Arreglos en kotlin:**

1. **¿Qué es un arreglo?**

**Respuesta:** El arreglo es un tipo de estructura de datos que nos permite almacenar elementos del mismo tipo.

1. **¿Creación de arreglos en kotlin?**

**Respuesta:** Para crear un arreglo en kotlin existen varias maneras de crear, la primera es mediante la función **arrayOf()**, que principalmente se utiliza para un arreglo de objetos de tipo entero, y para trabajar con objetos primitivos se pueden utilizar las siguientes funciones:

**ByteArray(), ShortArray(), IntArray(), LongArray(), FloatArray(), DoubleArray(), BooleanArray(), CharArray().**

****

1. **Accediendo a los elementos de un arreglo:**

**Respuesta:** Para poder acceder a los elementos de un arreglo se puede hacer mediante su numero de índice, recordemos que los índices comienzan desde el numero cero que representa al primer elemento de el arreglo.



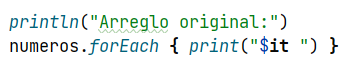
1. **Modificando los elementos de un arreglo:**

**Respuesta:** Para modificar los elementos de un arreglo, podemos utilizar su índice y después de esto agregarle un valor nuevo, lo que hará que el elemento del arreglo se actualice.



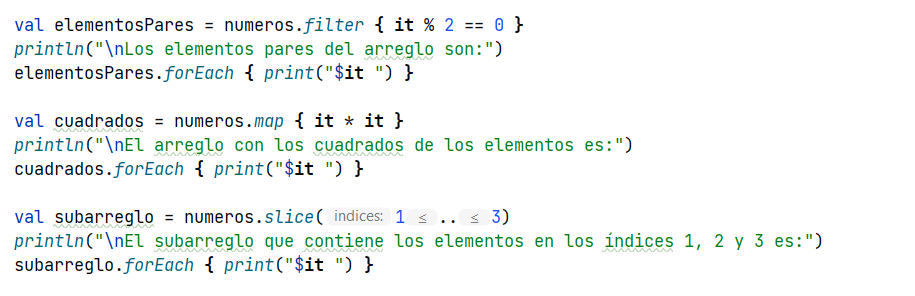
1. **Recorriendo un arreglo:**

**Respuesta:** Para recorrer el arreglo se puede utilizar un ciclo ForEach o el ciclo For, lo que hacen estos ciclos es iterar cada elemento del arreglo e imprimirlo.



1. **Funciones útiles para trabajar con arreglos en Kotlin:**

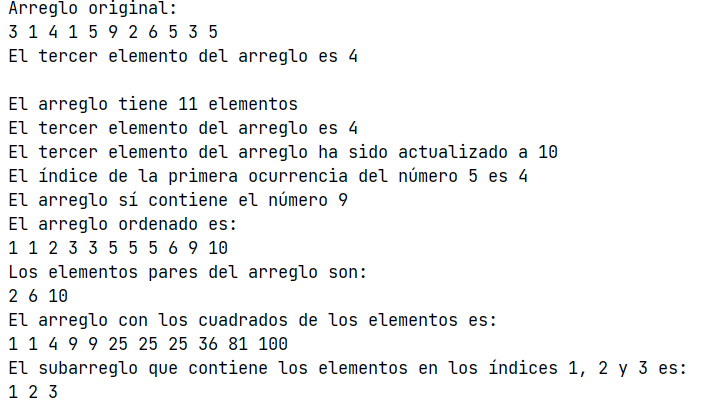
**Respuesta:** Algunas de las funciones útiles para trabajar con arreglos en kotlin son las siguientes funciones: **“size”:**que nos devuelve el numero de elementos que hay en el arreglo, **“set, get”:** que nos permiten acceder a algún elemento del arreglo y actualizar su valor, **“ Contains”:** nos devuelve **true** o **False,**  dependiendo si el elemento que buscamos este en el arreglo o no.



**Ejemplo Practico de un arreglo en Kotlin:**

**Texto

Descripción generada automáticamente**

****

**Nota:** En resumen, la variable implícita $it en un arreglo de Kotlin se utiliza para referirse a cada elemento del arreglo mientras se itera sobre él en algunas funciones de orden superior, lo que simplifica el código al evitar la necesidad de declarar una variable de bucle explícita.

**3. Listas en Kotlin:**

1. **¿Qué es una lista?**

**Respuesta:** Una lista es una estructura de datos que permite almacenar elementos de cualquier tipo y a diferencia de los arreglos, las listas puede crecer o decrecer dinámicamente

1. **Creación de listas en Kotlin**

**Respuesta:** Para crear las listas en kotlin se utiliza la función listOf()



1. **Accediendo a los elementos de una lista**

**Respuesta:** Para acceder a los elementos de una lista se utiliza el índice.

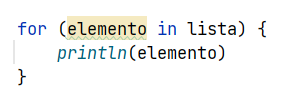


1. **Modificando los elementos de una lista**

**Respuesta:** Los elementos de una lista no se pueden modificar, entonces si usted desea modificar los elementos de una lista, debe proceder a crear nuevamente la lista.

1. **Recorriendo una lista**

**Respuesta:** Para recorrer los elementos de una lista se puede utilizar los ciclos forEach o for.



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

**Respuesta:** Las funciones útiles para trabajar las listas en kotlin son las siguientes;

**Map():**recibe los elementos de la lista y devuelve el resultado de la transformación

**Filter():** sirve para filtrar los elementos de la lista

**Reduce():**sirve para reducir los elementos de la lista a un único valor.

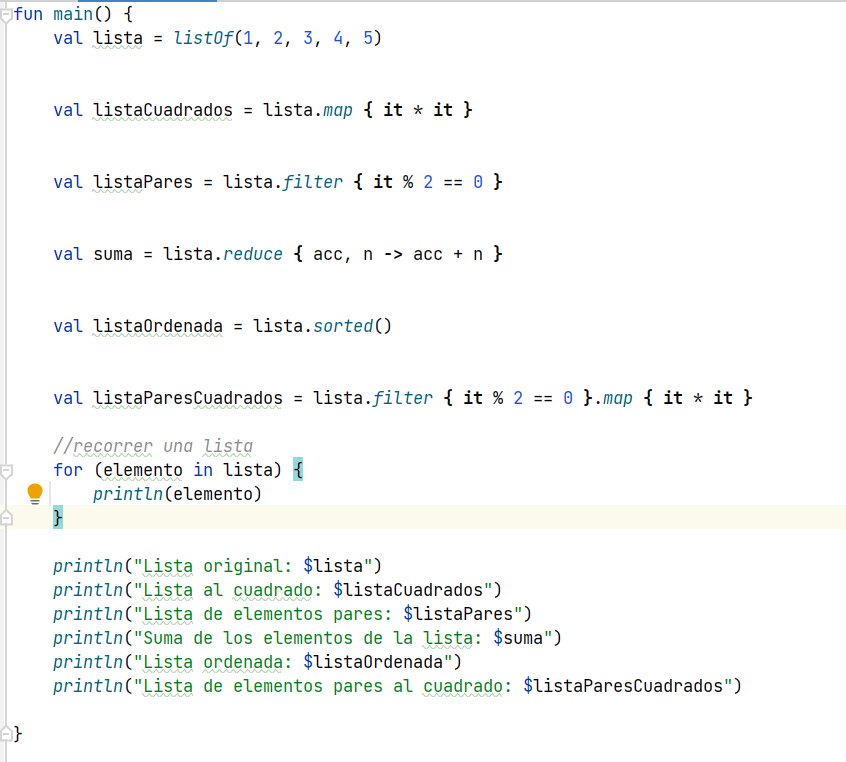
**Sorted():** se utiliza para ordenar los elementos de una lista

**Distinc():** se utiliza para eliminar elementos duplicados de una lista

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Ejemplo Practico de Listas en kotlin**

****

**Texto, Carta

Descripción generada automáticamente**

**4.** **Conjuntos en Kotlin**

1. **¿Qué es conjunto?**

**Respuesta:** Es una estructuras de datos que permiten almacenar solamente elementos únicos.

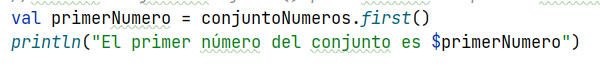
1. **Creación de conjuntos en Kotlin**

**Respuesta:** Para crear los conjuntos en kotlin se utiliza la función SetOf()



1. **Accediendo a los elementos de un conjunto**

**Respuesta:** Para acceder a los elementos de un conjunto se puede utilizar la función elementAt() que permite acceder a un elemento específico del conjunto mediante su indice.



1. **Modificando los elementos de un conjunto**

**Respuesta:** Al igual que en las listas los elementos no pueden ser modificados, entonces si se desea modificar algún elemento del conjunto lo debe crear de nuevo y ya con el elemento modificado.

1. **Recorriendo un conjunto**

**Respuesta:** Al igual que como en los arreglos y en las listas, los conjuntos se pueden recorrer con un ciclo forEach o for.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

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

**Respuesta:** Las funciones útiles que se pueden utilizar en los conjuntos son:

**Union():** Sirve para unir dos conjuntos y devuelve el conjunto que se forma de la unión

**intersect():** Esta función devuelve un nuevo conjunto que es la intersección de dos conjuntos

**subtract():** Esta función devuelve un conjunto que es la diferencia entre dos conjuntos.

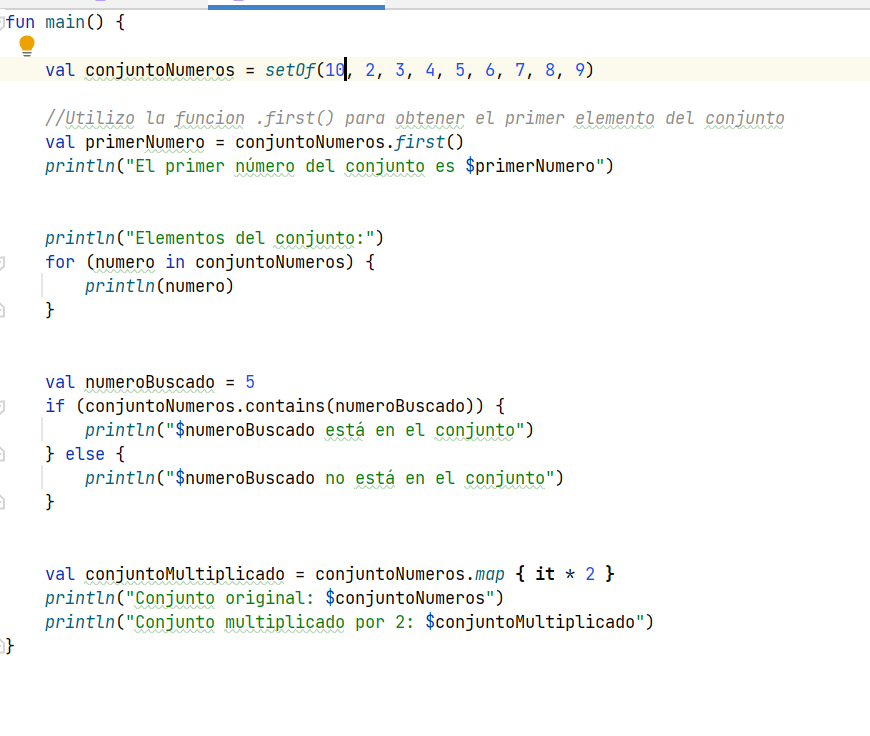
**contains():** Esta función devuelve un booleano que indica si un conjunto contiene un elemento específico

**size():** Esta función devuelve el número de elementos en un conjunto.

Captura de pantalla de un celular

Descripción generada automáticamente

**Ejemplo Practico de Conjuntos en Kotlin**

****

**Texto, Carta

Descripción generada automáticamente**

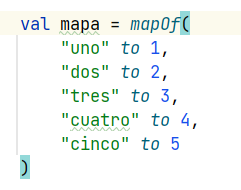
**5. Mapas en kotlin:**

1. **¿Qué es un mapa?**

**Respuesta:** Un mapa es una estructura de datos en la que se permite almacenar elementos clave-valor, en donde el valor es representado por la clave que es única.

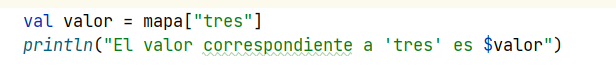
1. **Creación de mapas en kotlin**

**Respuesta:** Para crear mapas en kotlin se utiliza la función mapOf()



1. **Accediendo a los elementos de un mapa.**

**Respuesta:** Para acceder a los elementos de un mapa se debe llamar a la clave, lo que imprimirá solamente el valor que corresponde con la clave



1. **Modificando los elementos de un mapa**

**Respuesta:** En los mapas no se pueden modificar los elementos del mapa, por lo tanto si de desea modificar los elementos de un mapa se debe volver a crear de nuevo.

1. **Recorriendo un mapa**

**Respuesta:** Para recorrer el mapa se debe utilizar un ciclo for

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

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

**Respuesta:** Las funciones útiles que se pueden utilizar en los mapas son:

**Put()**: Agrega un par clave-valor al mapa mutable.

**Get()**: Obtiene el valor correspondiente a una clave del mapa.

**Keys()**: Devuelve una colección de las claves en el mapa.

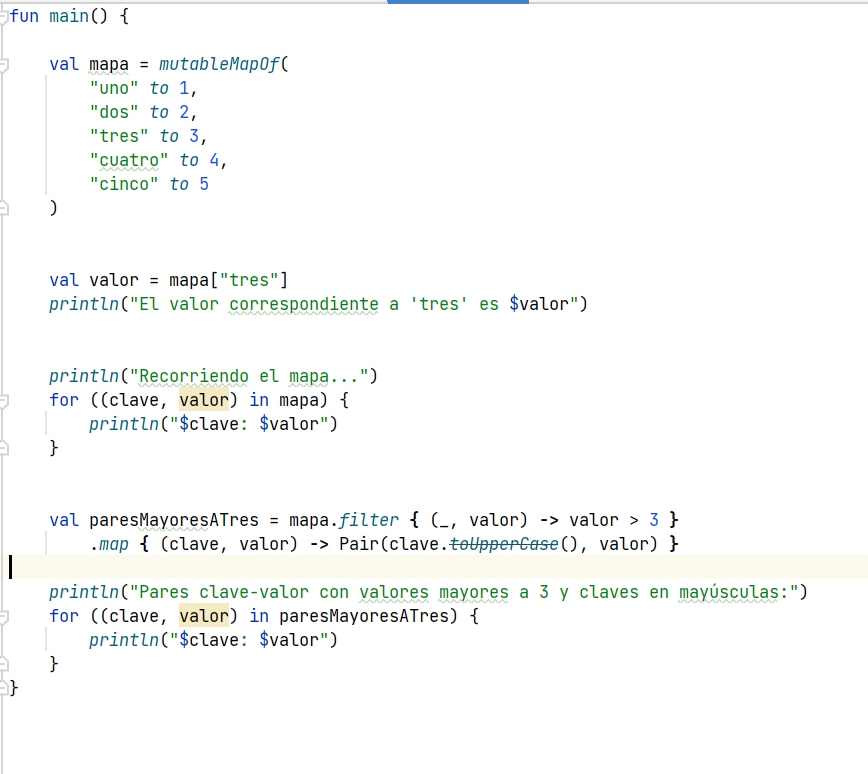
**Values()**: Devuelve una colección de los valores en el mapa.

**Filter()**: Devuelve un nuevo mapa que contiene solo las entradas que satisfacen la condición dada.

Texto

Descripción generada automáticamente

**Ejemplo practico de mapas en kotlin:**

****

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

**Nota: La función toUpperCase() es un método de la clase String en Kotlin que convierte todas las letras de una cadena en mayúsculas.**

**6. Pares en Kotlin:**

1. **¿Qué es un par?**

**Respuesta:** Un par es una estructura de datos que representa un conjunto de dos valores.

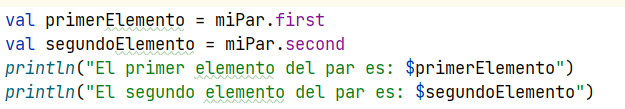
1. **Creación de pares en Kotlin**

**Respuesta:** Para crear un par en kotlin se puede utilizar la función pair()



1. **Accediendo a los elementos de un par**

**Respuesta:** Para acceder a los elementos se pueden utilizar las propiedades **first** Para acceder al primer elemento, y **second** Para acceder al segundo elemento.



1. **Modificando los elementos de un par**

**Respuesta:** Los elementos de un par son inmutables, por lo tanto si quiere modificar algún elemento de un par, lo debe crear nuevamente con los elementos actualizados.

1. **Recorriendo un par**

**Respuesta:** Se puede recorrer una lista de pares utilizando un bucle for y acceder a los elementos de cada par dentro del cuerpo del bucle



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

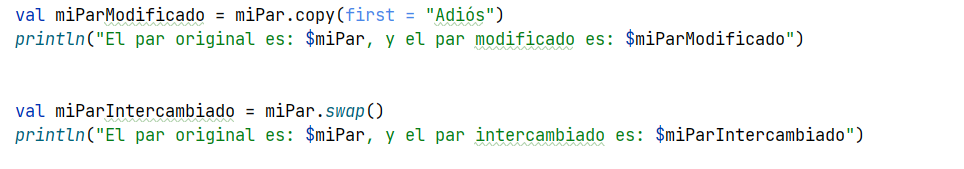
**Respuesta:** Las funciones útiles que se pueden utilizar en los pares son:

**to():** La función to() se utiliza para crear un par a partir de dos elementos**.**

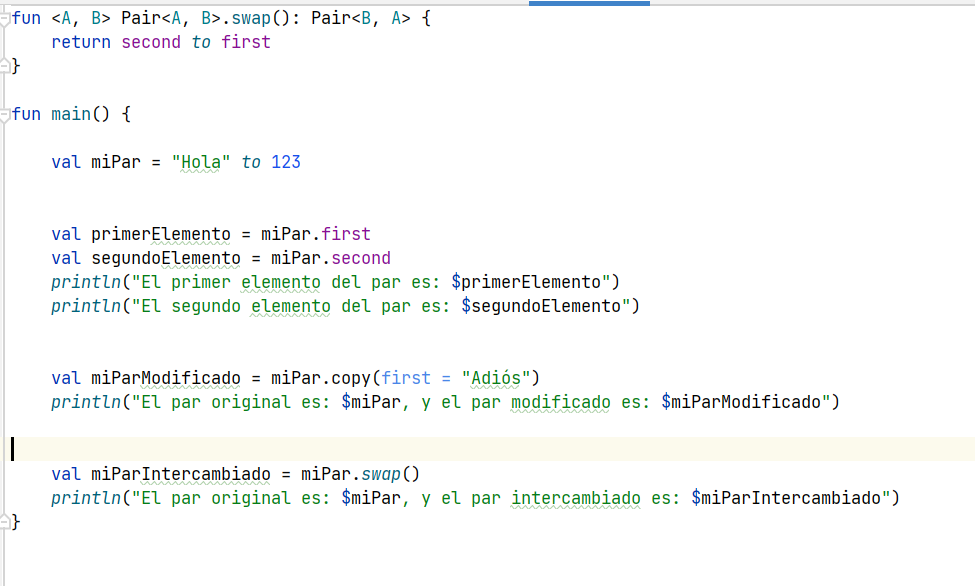
**component1() y component2():** Estas funciones se utilizan para acceder al primer y segundo elemento de un par

**toList():** La función toList() se utiliza para convertir un par en una lista de dos elementos.

**swap()**: La función **swap()** se utiliza para intercambiar los elementos de un par



**Ejemplo Practico de Pares en kotlin**

****

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Nota: Lo que creamos al principio es una extensión simple que ayuda a que la función Swap() funcione de manera correcta.**

**7. Prácticas de estructuras de datos en Kotlin**

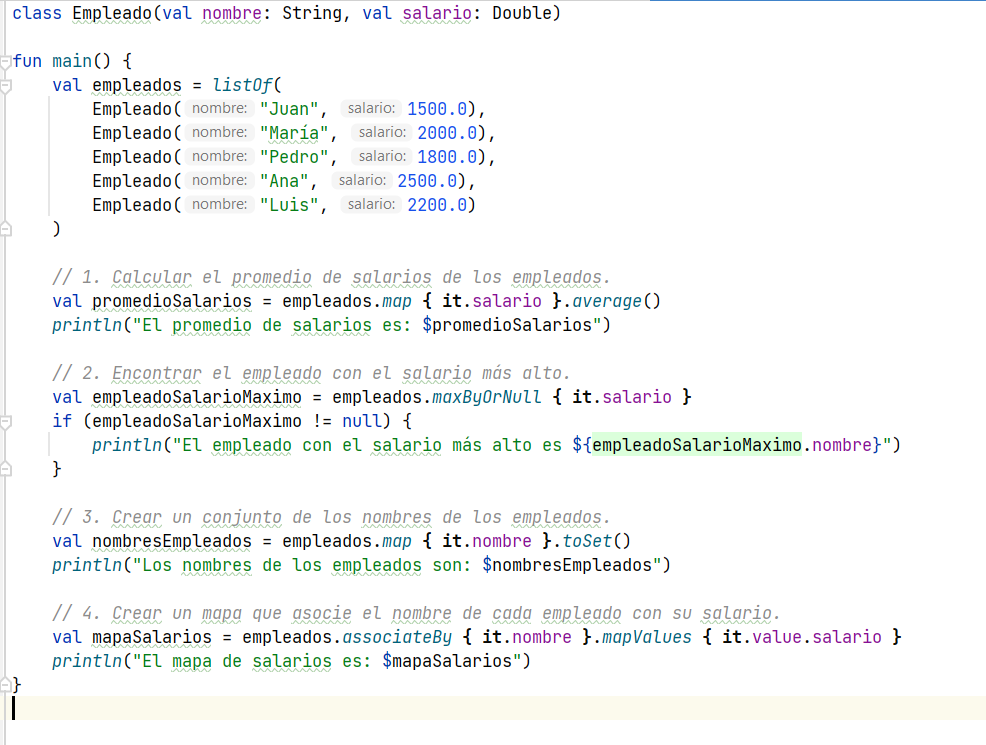
**A. Ejercicios prácticos para aplicar los conceptos aprendidos**

**Respuesta:**

Tienes una lista de empleados de una empresa, y deseas realizar una serie de operaciones utilizando diferentes estructuras de datos. Para ello, puedes crear una clase Empleado que tenga un nombre y un salario, y luego crear una lista de objetos Empleado que representen a los empleados de la empresa. A continuación, puedes realizar las siguientes operaciones:

1. Calcular el promedio de salarios de los empleados.
2. Encontrar el empleado con el salario más alto.
3. Crear un conjunto de los nombres de los empleados.
4. Crear un mapa que asocie el nombre de cada empleado con su salario.

**B. Solución a los ejercicios prácticos:**



**Texto

Descripción generada automáticamente**