Taller Estructuras de Datos en Kotlin

El objetivo de este taller es que los aprendices sean capaces de comprender y utilizar las principales estructuras de datos en Kotlin, incluyendo arreglos, listas, conjuntos, mapas y pares.

El aprendiz deberá realizar un informe donde se evidencien los siguientes puntos:

1. Introducción a las estructuras de datos en Kotlin
   a. ¿Qué son las estructuras de datos y para qué se utilizan?

      R=Las estructuras de datos son medios para manejar grandes cantidades de información de manera facil y nos permiten, como desarrolladores, organizar la información de manera eficiente

   b. Ventajas de utilizar estructuras de datos en Kotlin

      R=1. Kotlin proporciona una gran variedad de estructuras de datos integradas, como List, Set, Map, etc., que son fáciles de usar y comprender.

      2. Las estructuras de datos de Kotlin son flexibles y pueden adaptarse a diferentes necesidades. Por ejemplo, las listas se pueden utilizar para almacenar cualquier tipo de datos, y las colas se pueden utilizar para implementar lógica FIFO (primero en entrar, primero en salir).

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

      R=En kotlin las estructuras de datos se pueden crear sin definir el tipo de datos que tendrá, en cambio en Java no

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

      R= un arreglo es una colección de datos homogéneos

   b. Creación de arreglos en Kotlin

      R=

      kotlin:

      val kotlin = arrayOf(1, 5, 7, 3)

      java:

      int[] numeros = new int[5];

   c. Accediendo a los elementos de un arreglo

      R= kotlin:

      println(arreglo[1])

      java:

      println(arreglo[1])

   d. Modificando los elementos de un arreglo

      R=Kotlin:

      arreglo[0]=”hola”

      java:

      arreglo[0]=”hola”

   e. Recorriendo un arreglo

      R=kotlin:

      for(i in 0 until arreglo.size){

      print(i)

      }

      java:

      for(int i=0;i<arreglo.lenght;i++){

      system.out.println(arreglo[i])

      }

   f. Funciones útiles para trabajar con arreglos en Kotlin

      R=size : devuelve el tamaño del arreglo

      get : devuelve el valor en una posición específica del arreglo

set : establece el valor en una posición específica del arreglo

indexOf : devuelve el índice de la primera ocurrencia de un valor en el arreglo, o -1 si no se encuentra

3. Listas en Kotlin
   a. ¿Qué es una lista?
      R=Una lista es una colección de elementos ordenados
   b. Creación de listas en Kotlin
      R= Kotlin:
      val lista= mutableListOf()
      Java:
      ArrayList lista=new Arraylist()
   c. Accediendo a los elementos de una lista
      R=kotlin:
      lista.get(0)
      java:
      lista.get(0)
   d. Modificando los elementos de una lista
      R=kotlin:
      lista.set(2,”juan”)
      java:
      lista.set(3,”juan”)
   e. Recorriendo una lista
      R= kotlin:
      for(i in lista){
      print(i)
      }
      java:
      for(int i=0;i<lista.size;i++){
      system.out.println(lista[i])
      }
   f. Funciones útiles para trabajar con listas en Kotlin
      R= size: devuelve la cantidad de elementos en la lista.
      get: devuelve el elemento en el índice especificado.
      add: agrega un elemento al final de la lista.
      addAll: agrega todos los elementos de otra lista al final de la lista.

4. Conjuntos en Kotlin
   a. ¿Qué es un conjunto?
      R=Un conjunto es una colección que no tiene un orden específico y no permite valores duplicados.
   b. Creación de conjuntos en Kotlin
      R=Kotlin:
      toMutableSet() también es útil para modificar un conjunto
      val dias= setOf("lunes", "martes", "miercoles")
      Java:
      HashSet dias=new HashSet()

   c. Accediendo a los elementos de un conjunto
      R=Kotlin:
      print(dias)
      java:
      print(dias)
   d. Modificando los elementos de un conjunto
      R=kotlin:
      val dias = setOf(“lunes”, “martes”, “miercoles”) val newDias = dias.map { if (it == 2) 4 else it }
      java:
      Set dias = new HashSet<>(Arrays.asList(“lunes”, “martes”, “miercoles”)); set.remove(“martes”); set.add(“viernes”);
   e. Recorriendo un conjunto
      R=kotlin:

for (i in dias) { println(i) }

java:

for(object dia:dias){ system.out.println(dias) }

f. Funciones útiles para trabajar con conjuntos en Kotlin

size:devuelve el tamaño del conjunto

contains:verifica si un elemento se encuentra en el conjunto

union:une dos conjuntos

intersect:devuelve un conjunto solo con los elementos que coincidan en dos

subtract:devuelve un conjunto solo con los elementos que están en el primer conjunto y no en el segundo

map:devuelve un conjunto después de aplicar una operación

5. Mapas en Kotlin

a. ¿Qué es un mapa?

R=Un Map es una colección que consta de claves y valores

b. Creación de mapas en Kotlin

R=kotlin:

val materias = mutableMapOf(“matematicas” to “Ejemplo”, “religion” to “tulio”)+

java:

hashMap materias=new HashMap ()

c. Accediendo a los elementos de un mapa get()

R=Para acceder a un elemento de un mapa utilizaremos

d. Modificando los elementos de un mapa

R=kotlin println(“llaves ${materias.keys})

java:

System.out.println(materias.keyset())

e. Recorriendo un mapa

R=

kotlin:

for ((clave, valor) in materias) { println("$clave = $valor") }

java:

IteratorIterator=materias.keyset().iterator() while(iterator.hasNext()){ Integer lave=iterator.next() sistem.out.println(llave+”-”+materias.get(llave)}

f. Funciones útiles para trabajar con mapas en Kotlin

R= get:recuperar el valor de una clave

remove:eliminar un valor por medio de su clave

6. Pares en Kotlin

a. ¿Qué es un par?

R= Un par es una una estructura que permite guardar dos valores

b. Creación de pares en Kotlin

R=kotlin

val num = Pair("Hello", 42)

java

Pair<String, Integer> num = new Pair<>("Hello", 42);

c. Accediendo a los elementos de un par

R=

kotlin: pair.first

java: pair.first

d. Modificando los elementos de un par

R=

Kotlin:

val newNum = Num(num.first + 1, num.second - 1)

java:

Pair newNum= new Pair<>(num.first + 1, num.second - 1);

e. Recorriendo un par

R=for ((a, b) in num) { println("$a, $b") }

f. Funciones útiles para trabajar con pares en Kotlin

R=first y second: estas son propiedades de solo lectura que devuelven el primer y segundo elemento del par, respectivamente.
copy(): esta función devuelve una copia del par original con los mismos valores de first y second.
toString(): esta función devuelve una cadena que representa el par, en el formato "(first, second)".

7. Prácticas de estructuras de datos en Kotlin
   a. Ejercicios prácticos para aplicar los conceptos aprendidos
   b. Solución a los ejercicios

prácticos Recursos adicionales:

- Documentación oficial de Kotlin: **c**

Entrega.

Se deberá realizar la entrega de un informe con la solución de los puntos anteriores, el aprendiz acompañará la investigación con ejemplos practicos de cada estructura y deberá publicar el código fuente en un repositorio en GitHub.