

Las Data Class y su uso en las Apps

Nelson Ricardo Baquero

28 de julio de 2021

1. ¿CUÁNTOS CONSTRUCTORES PUEDE TENER UNA DATA CLASS?

Tiene un constructor principal donde se puede definir valores predeterminados para las variables.

```
1 data class Person(val name: String, val age: Int)
```

Se pueden definir también constructores secundarios que tomen variables arbitrarias, con la condición que siempre deben delegarse al constructor principal.

```
1 data class Person(val name: String) {  
2     constructor(name: String, age: Int) : this(name)  
3 }
```

Además, si trabaja en un proyecto con compatibilidad con java, es posible especificar la anotación **@JvmOverloads** para generar varios constructores en bytecode que puedan ser utilizados por código Java.

```
1 data class Person @JvmOverloads constructor(val name: String,  
2                                             val age: Int? = 0)
```

2. NUMERE LAS VENTAJAS QUE TIENEN LAS DATA CLASS SOBRE LAS CLASES REGULARES

1. Requiere escribir mucho menos código por parte del programador.
2. Son inmutables, lo que significa que comprobar la igualdad en dos instancias con el mismo contenido siempre retornará *verdadero*.
3. Posee varias utilidades basadas en su contenido: `equals()`, `toString()`, `componentN()`, `copy()`.

3. ¿CUÁLES SON LAS LIMITACIONES QUE TIENEN EN LA PROGRAMACIÓN ORIENTADA A OBJETOS?

- Las data classes no pueden ser marcadas como: abstract, sealed, open o inner.
- Los constructores de las data classes siempre deben recibir al menos un parámetro.

4. ¿QUÉ SON LAS DESTRUCTURING DECLARATIONS Y CÓMO FUNCIONAN?

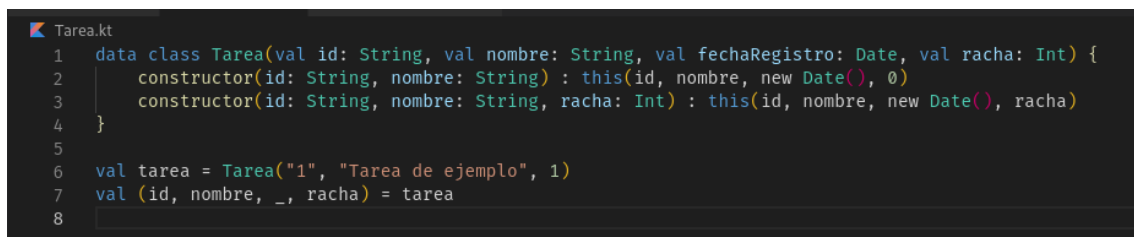
Las *destructuring declarations* sirven para declarar múltiples variables al mismo tiempo. Es especialmente útil para ahorrar tiempo al escribir las declaraciones de variables que posee un objeto intermedio.

```
1 val (name, age) = person
```

Ejemplo, retornar dos variables desde una función y utilizar una *destructuring declaration* para obtenerlas:

```
1 data class Result(val result: Int, val status: Status)
2 fun function(...): Result {
3     // codigo
4     return Result(result, status)
5 }
6
7 // Para utilizar la funcion y deestructurar el resultado:
8 val (result, status) = function(...)
```

5. CREE UNA DATA CLASS COMO EJEMPLO



```
Tarea.kt
1 data class Tarea(val id: String, val nombre: String, val fechaRegistro: Date, val racha: Int) {
2     constructor(id: String, nombre: String) : this(id, nombre, new Date(), 0)
3     constructor(id: String, nombre: String, racha: Int) : this(id, nombre, new Date(), racha)
4 }
5
6 val tarea = Tarea("1", "Tarea de ejemplo", 1)
7 val (id, nombre, _, racha) = tarea
8
```

Figura 5.1: Ejemplo de data class en Kotlin