



M3 - Programació

Tipus compostos - data class



Índex

- data class
 - Presentació
 - Definició
 - Declaració
 - Ús de les data class
 - Tasques comunes ja implementades

data class - Presentació



- Així com els arrays (i llistes) ens permeten manejar diferents dades homogènies (del mateix tipus), una data class ens permet manejar diferents dades heterogènies (de diferents tipus).
- En altres llenguatges de programació es diuen estructures (struct - C) o registres.
- En Kotlin, a més, s'implementen una sèrie de funcions de suport a aquestes data class



data class - Definició

- La definició d'una data class és així:

```
data class Persona (  
    var nom: String,  
    var cognoms: String,  
    var edat: Int  
)
```

Els camps de la data class poden ser variables (var) o constants (val)



data class - Declaració

- La declaració d'una data class seria així:

```
var persona1 : Persona  
var persona2 : Persona = Persona("Santi", "Rivas", 25)
```

- O la podem fer servir com el que és: un nou tipus de dades

```
var llistaPersones : MutableList<Persona> = mutableListOf<Persona>()
```

data class - Ús



- Per usar una data class ho farem amb el nom de la variable, seguit d'un punt i el nom del camp al que volem accedir.

```
var alumne: Persona = Persona("Joan", "Puig", 19)
alumne.edat = 20
println("L'edat de l'alumne és ${alumne.edat}")
```

- S'accedeix de igual forma tant per llegir com per escriure i cada camp es tracta de la mateixa forma que una variable.



data class - Tasques

còmunes

A diferència d'altres llenguatges, en Kotlin, el mecanisme class fa un munt de treball per nosaltres. Ex: comparacions.

```
var alumne1: Persona = Persona("Joan", "Puig", 19)
var alumne2: Persona = Persona("Joan", "Puig", 19)
if (alumne1 == alumne2) // cert
```

Altres llenguatges (ex: Java)

```
Persona alumne1 = new Persona("Joan", "Puig", 19);
Persona alumne2 = new Persona("Joan", "Puig", 19);
if (alumne1 == alumne2) // fals
if (alumne1.nom.equals(alumne2.nom) &&
    alumne1.cognoms.equals(alumne2.cognoms) &&
    alumne1.edat == alumne2.edat ) // cert
```



data class - Tasques comunes

- També ens permet fer còpies

```
var alumne1: Persona = Persona("Joan", "Puig", 19)
var alumne2: Persona = alumne1.copy()
```

Això genera un altre objecte del tipus Persona i amb els mateixos valors. Opcionalment li podem donar nous valors a alguns camps.

```
var alumne1: Persona = Persona("Joan", "Puig", 19)
var alumne2: Persona = alumne1.copy(cognoms = "Pons i Bofill")
```



data class - Tasques comunes

- Però compte amb les còpies que poden tenir sorpreses!!!
- Les data class poden tenir camps definits en el cos de la classe.

Aquests no es copien!!!

```
data class Persona (var nom: String,var cognoms: String,var edat: Int) {  
    var notaFinal:Int = 0  
}  
...  
var alumne1: Persona = Persona("Joan","Puig",19)  
alumne1.notaFinal = 8  
var alumne2: Persona = alumne1.copy()  
if (alumne2.notaFinal == 8)  // FALS!!!
```



data class - Tasques comunes

- Altres funcions implementades a les data class
 - **equals()** herència de Java, compara el contingut de les data class
 - **toString()** genera un String en format `Persona(nom=Joan, cognoms=Puig, edat=18)`
 - **componentN()** accedeix al camp *N* seguint l'ordre de la declaració. El primer camp és el **component1()**