M3 - Programació

Introducció a la Programació Orientada a Objectes (POO)

Índex

- Un món orientat a objectes
- Exemple Mario Kart
  - Galetes
  - Motlle
  - Package
  - Class
  - Data class
  - Constructor
  - Valors per defecte
  - Imprimir
  - Crear llista personatges

Un món orientat a objectes

- **En quina situació ens trobem?:** Els sistemes s'han fet més **complexos**, i per tant cal implementar solucions més complexes.

- **Què necessitem?** Necessitem mecanismes que permetin un **nivell d'abstracció més alt** en el procés de desenvolupament de programari i que la tasca de programar sigui **més propera al pensament i llenguatge humà**.

Un món orientat a objectes

- **Que ens aporta l'OO?** L'orientació a objectes és una veritable **aproximació entre la manera com un ésser humà pot estructurar un problema i com codificar-lo** en forma de programa, de manera que ambdós processos siguin tan semblants com sigui possible.

- **A tenir en compte:** Per a entendre com funciona l'orientació a objectes és necessari fer un ***"canvi de xip"*** respecte a la manera de pensar de la programació estructurada i modular.

Un món orientat a objectes

- L'**orientació a objectes** no és un tipus de llenguatge de programació sinó **és una metodologia de disseny** per a crear programes.

- Els llenguatges de programació orientats a objectes són els que apliquen aquesta metodologia (Java, C++, C#, Javascript, PHP, Python, Kotlin …)

Un món orientat a objectes

- L'orientació a objectes permet:
  - **Analitzar els problemes** que volem resoldre mitjançant aplicacions informàtiques **de la mateixa manera que altres problemes del món real** com altres disciplines de l'enginyeria: mecànica, arquitectura, etc.
  - **Dissenyar l'aplicació de manera totalment independent al llenguatge de programació** que s'utilitzarà per després implementar-la mitjançant un llenguatge de programació orientat a objectes.

Exemple Mario Kart

- Pensem en els personatges de Mario Kart que apareixen a la pantalla de selecció del joc i quina informació hi apareix...

Exemple Mario Kart - Personatge

- Quins aspectes **defineixen els personatges**?
- Si volem fer una aplicació que ens ajudi a triar el nostre personatge, com **guardariem** la seva **informació**?
- Crearem **una variable per a cada atribut** que volem guardar?
  - var name: String = “Mario”
  - var speed: Float = 5.5f
  - var acceleration: Float = 5.5f
  - var weight: Float = 5.5f
  - var handling: Float = 5.5f
  - var traction: Float = 5.5f
- I si volem **guardar TOTS els personatges**???
- Un vector? Una matriu amb tantes columnes com atributs??

Exemple Mario Kart - Galetes

- Ja que per defecte **existeixen múltiples tipus de dades** dins del JDK, seria ideal que existís un **tipus de dades** per a guardar els **personatges de Mario Kart**.
- Perquè no està fet?!
- I no el podem fer nosaltres??
- ...
- De moment anem a fer unes galetes que tinc gana!

Exemple Mario Kart - Galetes

- Què necessitem per fer les **galetes del Mario Kart**?
  - 125 g de mantega (a temperatura ambient)
  - 150 g de sucre (pot ser blanc, morè o una barreja)
  - 1 ou gran
  - 1 culleradeta d'essència de vainilla
  - 180 g de farina de blat
  - 30 g de cacau en pols sense sucre
  - 1/2 culleradeta de bicarbonat de sodi
  - 1/4 culleradeta de sal
  - 100-150 g de gotes de xocolata o trossos de xocolata
  - I **motlles de les figures dels personatges de Mario Kart !!!**

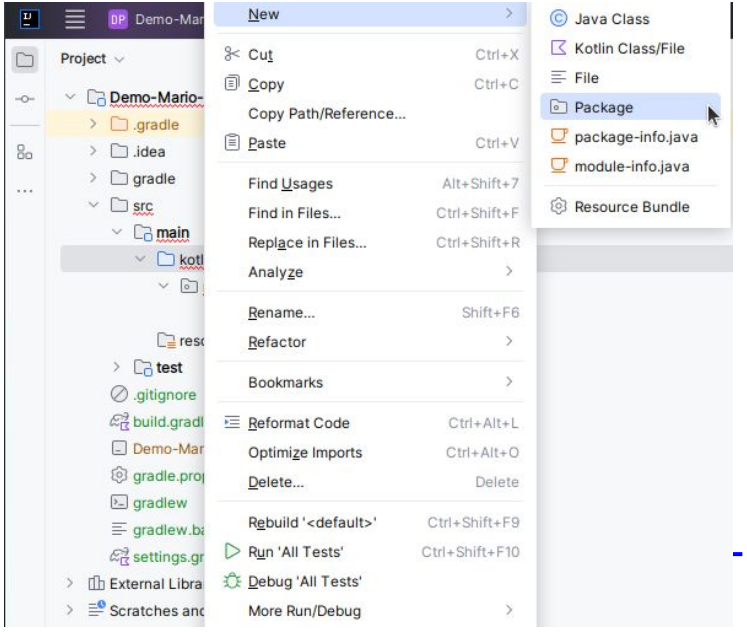
Exemple Mario Kart - Motlle

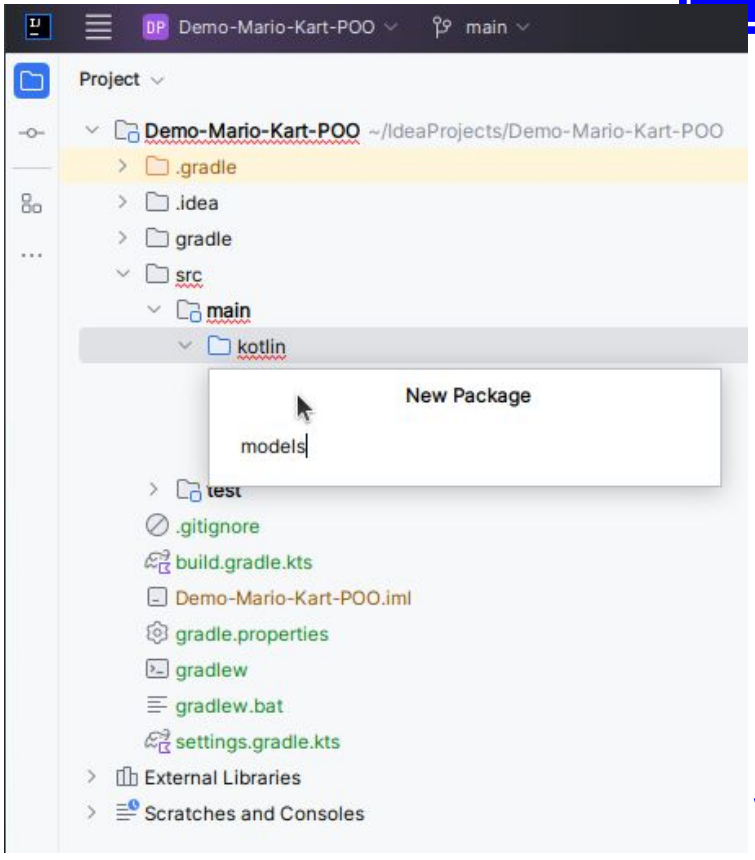
- Amb aquests ingredients, quantes galetes puc fer de cada personatge?
- Resposta: m'és igual, moltes! Però necessito el motlle per poder fer les figures...

- Anem a fer el motlle doncs !

Exemple Mario Kart - Package

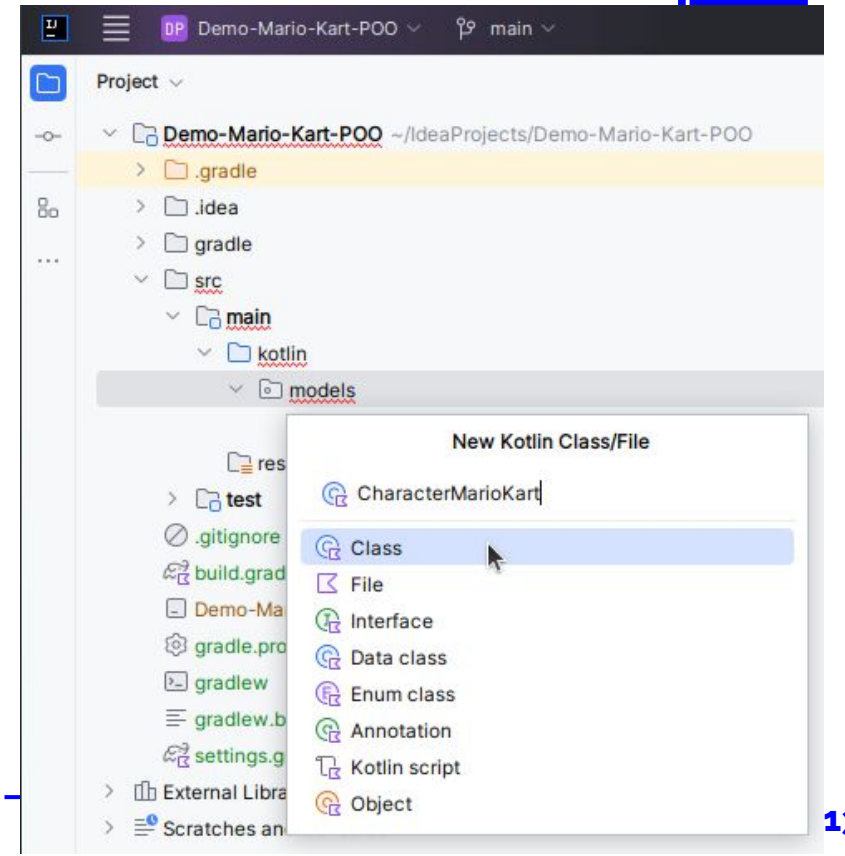
- Crearem un **package** nou dins de la carpeta de **src** i li direm **models:**





Exemple Mario Kart - Class

- Crearem un nou arxiu de kotlin dins del package **models**
- Aquest nou arxiu, serà un arxiu especial de tipus **Class**
- Aquest fitxer serà el nostre **motlle** ja que definirà un **nou tipus de dades**!
- Dins del package de models, hi podem definir tants tipus de dades nous *(classes)* com vulguem!
- Aquests tipus de dades, estaran disponibles dins del nostre projecte.



Exemple Mario Kart - data class

- Tal com hem vist, en Kotlin podriem crear un **data class** nou.
- Però un **data class** és només una aproximació a les **classes de dades** que ens permetràn fer moltes més coses en el futur!

```
package models

data class CharacterMarioKart(
    var name: String,
    var speed: Float,
    var acceleration: Float,
    var weight: Float,
    var handling: Float,
    var traction: Float
)
```

Exemple Mario Kart - Class

- Anem a veure una aproximació del codi de la **nova classe de dades** que volem crear:

```
package models

class CharacterMarioKart {
    var name: String
    var speed: Float
    var acceleration: Float
    var weight: Float
    var handling: Float
    var traction: Float
}
```

Package on es troba

Nom del nou tipus de dades

Atributs de la classe. Per a cada personatge guardarem tot això.

Exemple Mario Kart - Constructor

- Ara tenim un nou tipus de dades definit amb els seus atributs similar a com ho fariem amb un **data class**.
- Però encara no ens permet usar-lo com a motlle per a crear personatges de Mario Kart.
- Necessitem donar-li les **indicacions** sobre com posar la massa de la galeta dins del motlle.
- Per a fer-ho definirem els **constructors.**
- Els constructors son els encarregats d'**inicialitzar els els objectes** de la classe.

Exemple Mario Kart - Constructor

- Les **classes de dades poden contenir**:
  - **Atributs ->** per a guardar informació
  - **Mètodes ->** per a definir les seves funcionalitats

- Els **constructors** són un **mètode especial** que permeten determinar com s'**inicialitzaràn els objectes** de la classe.

Exemple Mario Kart - Constructor

- Definirem els constructors just a sota dels atributs.

- Podem **definir més d'un constuctor** per permetre inicialitzar els objectes de la classe de diferents maneres.

```
constructor(name: String, speed: Float,
acceleration: Float, weight: Float,
handling: Float, traction: Float) {
    this.name = name
    this.speed = speed
    this.acceleration = acceleration
    this.weight = weight
    this.handling = handling
    this.traction = traction
}
```

Exemple Mario Kart - Constructor

- El constructor creat, rep tants paràmetres com atributs té la classe.
- Però això pot canviar en funció de quines maneres volem poder inicialitzar els nostres objectes.
- La paraula reservada ***this*** s'utilitza en kotlin per tal de **fer referència als elements de la classe** i trencar així l'ambigüetat amb el nom dels paràmetres d'entrada del constructor.

Exemple Mario Kart - Constructor

- Exemple d'implementació d'un constructor:

```
constructor(name: String, speed: Float, acceleration: Float,
        weight: Float, handling: Float, traction: Float) {
    this.name = name
    this.speed = speed
    this.acceleration = acceleration
    this.weight = weight
    this.handling = handling
    this.traction = traction
}
```

Exemple Mario Kart - Crear objectes

- Amb els atributs i el(s) constructor(s) definits, ja podem fer un primer ús del nou tipus de dades definit:

```
import models.*

fun main(){
    var mario: CharacterMarioKart
    mario = CharacterMarioKart("Mario", 5.5f, 5.5f, 5.5f, 5.5f, 5.5f)

    println(mario.name)
}
```

Exemple Mario Kart - Valors per defecte

- Si volem, podem donar valors per defecte als atributs de la classe:

```
class CharacterMarioKart {
    var name: String = "Unknown"
    var speed: Float = 0.0f
    var acceleration: Float = 0.0f
    var weight: Float = 0.0f
    var handling: Float = 0.0f
    var traction: Float = 0.0f

...
```

Exemple Mario Kart - Segon constructor

- Si volem, podem **definir un altre constructor** que ens permeti **inicialitzar objectes** de la classe **especificant només el valor per l'atribut nom** del personatge:

```
constructor(name: String){
    this.name = name
}
```

Exemple Mario Kart - Yoshi

- Ara tenim **dues maneres** d'**inicialitzar *instàncies*** de la classe.

```
import models.*

fun main(){
    var mario: CharacterMarioKart
    mario = CharacterMarioKart("Mario", 5.5f, 5.5f, 5.5f, 5.5f, 5.5f)
    println(mario.name)

    var yoshi: CharacterMarioKart = CharacterMarioKart("Yoshi")
    println(yoshi.name)
    println(yoshi.acceleration)
}
```

Exemple Mario Kart - Imprimir objectes

- Com ho farem per imprimir tots els detalls dels personatges per pantalla?

Main.kt

```
1  import models.*
2
3  fun main(){  new *
4      var mario: CharacterMarioKart
5      mario = CharacterMarioKart( name: "Mario",  speed: 5.5f,  acceleration: 5.5f,  weight: 5.5f,  handling: 5.5f,  traction: 5.5f)
6
7      println(mario)
8      println(mario.toString())
9  }
```

Run MainKt

```
/home/rai/.jdks/corretto-17.0.13/bin/java -javaagent:/snap/intellij-idea-ultimate/557/lib/idea_rt.jar=42499:/snap/intellij-
models.CharacterMarioKart@279f2327
models.CharacterMarioKart@279f2327
```

Exemple Mario Kart - Imprimir objectes

- **Per defecte**, si l'objecte existeix, ens mostra la **posició hexadecimal de la memòria RAM** on es troba allotjat.

- Si volem que els objectes de la classe creada s'imprimeixin d'una altra manera, ho haurem de definir a la classe.

- Per a fer-ho, afegirem el mètode ***toString*** dins del codi de la classe, després dels atributs i dels constructors.

Exemple Mario Kart - Override toString

- Tal com hem comentat, el mètode ***toString*** ja ve amb un comportament predefinit que imprimeix la posició hexadecimal.

- Com que volem sobreescriure el seu comportament, usem la paraula clau ***override*** just al principi de la seva declaració***.***

Exemple Mario Kart - Override toString

- Podem especificar el format del text i la info a mostrar que vulguem:

```
override fun toString(): String {
    return "CharacterMarioKart(name='$name',
                               speed=$speed,
                               acceleration=$acceleration,
                               weight=$weight,
                               handling=$handling,
                               traction=$traction)"
}
```

Exemple Mario Kart - Imprimir objectes

- Ara, quan intentem **imprimir** els detalls dels **personatges**, obtindrem una sortida diferent a l'anterior:

Exemple Mario Kart - Llista personatges

- Podem guardar tots els personatges en una **llista**: