M3 - Programació

POO: Encapsulació i modificadors d'accés

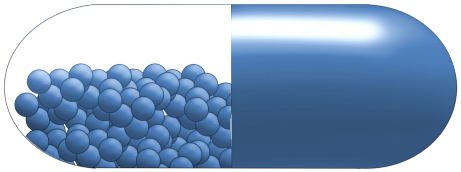
Índex

- Encapsulació i modificadors d'accés

Encapsulació i modificadors d'accés

- En POO, l'**encapsulació** és fonamental i es tracta de la **gestió dels permisos d'accés** als diferents punts del projecte.

- El seu propòsit és el d'**assegurar** que les **classes** s'**usin** tal i com van ser pensades i **dissenyades**.



- L'**encapsulació** protegeix els objectes, redueix la complexitat del projecte i assegura la coherència de dades dels objectes.

Encapsulació i modificadors d'accés

- En Programació Orientada a Objectes, la gestió dels **permisos d'accés** als **atributs** i **mètodes** de la classe en són una part molt important.

- L'**encapsulació** de les **classes** del projecte les **gestionarem** a través dels **modificadors d'accés**.

- De la mateixa manera que en un **electrodomèstic** o en una **consola de videojocs** només els podrem fer servir usant els **comandaments** visibles des de fora **amb els quals han estat dissenyats**, el mateix passarà amb les classes de dades.

Encapsulació i modificadors d'accés

- Podem usar la Switch II a través dels seus ***botons "públics"***.

- Però **la resta de components** de la consola que queden a dins són ***"privats"***, ja que, com a usuari, no hi podem accedir.

Encapsulació i modificadors d'accés

- En la **definició de les classes**, els modificadors d'accés els **aplicarem** als **atributs** i als **mètodes**.

- Disposem de **diversos modificadors d'accés** que apliquen **diferents nivells de permisos**.

Encapsulació i modificadors d'accés

- **public:** És el modificador d'accés per defecte i dóna accés al recurs (atribut o mètode) des de tot arreu del projecte
- **private:** Només dóna permís d'accés des de dins de la mateixa classe. Els atributs i mètodes privats no seràn visibles des de fora.
- **protected:** Només té sentit en un projecte on apliquem herència de classes o implementació d'*interfícies*. Es comporta igual que **private** amb la diferència que les classes filles també hi tindràn accés.
- **internal o *package protected*:** Dóna accés a un recurs a tots els fitxers que estiguin dins del mateix ***package***.

Encapsulació i modificadors d'accés

Modificador d'accés	Accessible dins de la mateixa classe	Accessible des de subclasses	Accessible dins del mateix fitxer .kt	Accessible des d'altres classes dins del mateix package	Accessible des fitxers d'altres packages *(Main.kt)*
`public`	✔	✔	✔	✔	✔
`private`	✔	✘	✔	✘	✘
`protected`	✔	✔	✘	✘	✘
`internal`	✔	✔	✔	✔	✘

Modificadors d'accés - Public

- El modificador d'accés **public** és el modificador d'accés que s'aplica per defecte i omissió. És a dir, si no n'especifiquem cap, s'aplicarà aquest.
- Només l'hauriem d'aplicar per a aquells atributs i mètodes que vulguem que siguin visibles i accionables des de tot arreu del projecte.
- Seràn els **"comandaments"** del nostre programa.

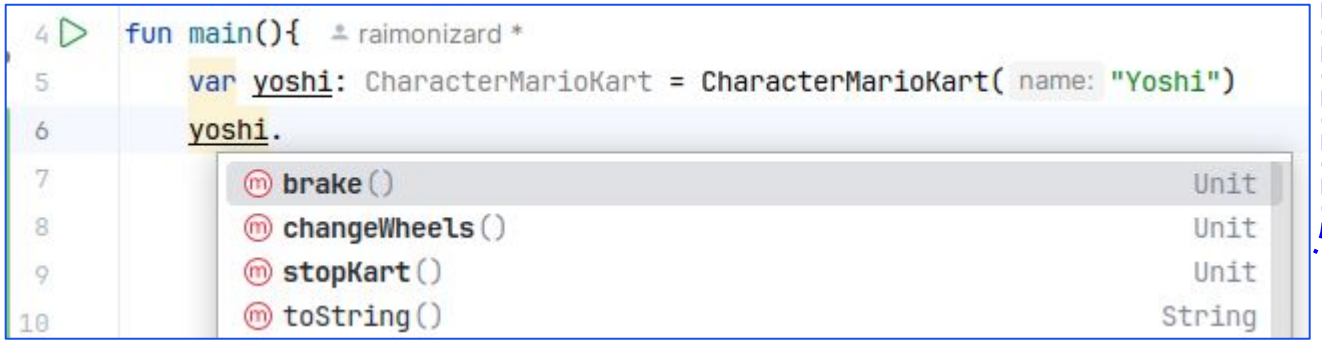
Modificadors d'accés - Public

- Exemples d'ús del modificador d'accés **public** en mètodes:

```
public fun brake(){
    this.acceleration /= 2.0f
}

public fun changeWheels(){
    this.traction = 100.0f
}
```

```
public fun stopKart(){
    this.acceleration = 0.0f
    this.speed = 0.0f
}
```



Modificadors d'accés - Private

- El modificador d'accés **private** és el modificador d'accés més restrictiu.

- L'usarem generalment per a tots els **atributs de la classe** i per als **mètodes *helpers*** que no volem que estiguin disponibles des de fora.

Modificadors d'accés - Private

- Exemples d'ús del modificador d'accés **private** aplicat a **atributs** i **mètodes** *(aquests no seràn disponibles des del Main.kt)*:

```
class CharacterMarioKart {
    private var name: String = "Unknown"
    private var speed: Float = 0.0f
    private var acceleration: Float = 0.0f
    …

                    private fun calcSpeed(time: Int){
                        this.speed = this.acceleration * time
                    }
```

Modificadors d'accés - Protected

- El modificador d'accés **protected** és un modificador d'accés que farem servir a les classes que puguin tenir **herència**.

- Serveix per tal de que una **classe filla** pugui **accedir als atributs i mètodes de la classe pare** com si fòssin seus.

Modificadors d'accés - Protected

- Exemple de modificador d'accés **protected** aplicat a atributs i mètodes que podran ser usat dins de la classe i també a les classes que l'heretin:

```
class CharacterMarioKart {
    protected var name: String = "Unknown"
    protected var speed: Float = 0.0f
    protected var acceleration: Float = 0.0f
    …
                        protected fun resetCharacter(){
                            this.name = "Unknown"
                            this.speed: Float = 0.0f
                            this.acceleration: Float = 0.0f
                        }
```

Modificadors d'accés - Internal

- El modificador d'accés **internal** és un modificador d'accés que farem servir poques vegades i només té sentit si tenim el projecte fortament modulat usant packages.

- Serveix per tal de que un mètode o atribut estigui disponible per a tots els arxius que es troben dins del mateix package i també a les classe filles en cas d'herència.

- A efectes pràctics es comporta igual que el modificador **public** amb l'única diferència que no permet l'accés des de fora del package *(El Main.kt no hi tindrà accés a no ser que estigui al mateix package)*.

Modificadors d'accés - Internal

- Exemple de classe estàtica amb un mètode **internal** definit i el seu **ús** des d'**una altra classe del mateix package**.

```
object Calculadora {
    internal fun suma(a: Float, b: Float): Float{
        return a + b
    }
}


public fun turbo(){
    this.speed = Calculadora.suma(this.speed, this.acceleration)
}
```