# Leçon 21
# Les data class

Niveau 1

## Introduction

Imaginez que vous travaillez dans un bureau où vous devez constamment manipuler des fiches d'informations. Par exemple, des fiches de clients avec leur nom, leur âge, et leur adresse. Vous voulez pouvoir les comparer, les dupliquer, et les afficher facilement. En Kotlin, une data class est spécialement conçue pour ces types de tâches.

Une **data class** est une classe spécialement conçue pour gérer des données de manière efficace. Elle offre des fonctionnalités automatiques pour comparer, copier, et afficher des objets, sans que vous ayez à écrire tout ce code vous-même.

## Qu'est-ce qu'une data class ?

Une data class en Kotlin est une classe dont le but principal est de stocker des données. Lorsqu'une classe est marquée comme data, Kotlin génère automatiquement plusieurs fonctions utilitaires très pratiques.

Prenons un exemple simple :

```
data class Produit(val nom: String, val prix: Double, val enStock: Boolean)
```

Dans cet exemple, nous avons créé une data class nommée Produit avec trois propriétés : nom, prix, et enStock. Voici ce que Kotlin génère automatiquement pour vous :

- equals() : Pour comparer deux objets Produit en fonction de leurs propriétés.
- hashCode() : Pour obtenir un code de hachage basé sur les propriétés.
- toString() : Pour obtenir une représentation textuelle claire de l'objet.
- copy() : Pour copier un objet en modifiant certaines propriétés si nécessaire.

## Différences entre une Classe Normale et une data class

Une data class est conçue pour simplifier le travail avec des données. Voici ce qui la distingue d'une classe normale :

- Classe Normale : Vous devez définir manuellement des méthodes comme equals(), hashCode(), et toString() si vous avez besoin de ces fonctionnalités.

```
class Produit(val nom: String, val prix: Double, val enStock: Boolean)

val produit1 = Produit( nom: "Ordinateur",  prix: 999.99,  enStock: true)
val produit2 = Produit( nom: "Ordinateur",  prix: 999.99,  enStock: true)

val identique = produit1 == produit2 // Cela compare les références, pas les valeurs
```

Dans cette exemple, nous définissons une classe normal et nous créons 2 variables contenant les mêmes propriétés. Cependant, lorsque nous les comparons, nous vérifions si ce sont les mêmes objets. Donc la variable identique vaut false.

- data class : Kotlin génère les méthodes equals(), hashCode(), et toString() automatiquement, basées sur les propriétés définies.

```
data class Produit(val nom: String, val prix: Double, val enStock: Boolean)

val produit1 = Produit( nom: "Ordinateur",  prix: 999.99,  enStock: true)
val produit2 = Produit( nom: "Ordinateur",  prix: 999.99,  enStock: true)

val identique = produit1 == produit2 // True, car les valeurs sont égales
```

Sur cette exemple, le code est le même sauf que nous avons ajouté le mot clé data à la class. Lorsque nous les comparons, nous vérifions si chaque propriété du constructeur sont égaux. Ici, la variable identique sera donc à true.

La data class permet une comparaison des valeurs, une copie facile, et une représentation textuelle sans effort supplémentaire.

## Cas d'Utilisation des data class

- Modélisation de données : Les data class sont idéales pour représenter des entités du monde réel, comme des utilisateurs, des produits, ou des transactions. Elles sont particulièrement utiles lorsque l'égalité des objets, la possibilité de les copier, et leur affichage sont des fonctionnalités essentielles.

```
data class Utilisateur(val nom: String, val age: Int, val email: String)
```

- Échange de données : Elles sont souvent utilisées pour structurer les données échangées entre différentes parties de votre application, comme les couches de présentation, de logique métier, et de stockage.
- Collections : Lorsqu'elles sont utilisées dans des collections (listes, ensembles, etc.), les data class facilitent la recherche, le tri, et la comparaison d'objets. En effet, les collections se basent sur la fonction equals() générée automatiquement dans les data class

```
val utilisateurs = listOf(
    Utilisateur( nom: "Alice",  age: 28,  email: "alice@example.com"),
    Utilisateur( nom: "Bob",  age: 34,  email: "bob@example.com")
)
val contains = utilisateurs.contains(Utilisateur( nom: "Alice",  age: 28,  email: "alice@example.com"))
```

Vous l'aurez sans doute compris, les data class sont très utilisées en Kotlin.

## Copier un objet

Un des avantages des data class est la possibilité de copier un objet facilement. Il suffit d'utiliser la fonction copy() et de mentionner en paramètre seulement les propriétés que nous voulons modifier par rapport à l'original.

Voici un exemple :

```
val utilisateur1 = Utilisateur( nom: "Alice",  age: 28,  email: "alice_patoch@example.com")
val utilisateur2 = utilisateur1.copy(nom = "Patrick")
```

Dans cette exemple, l'utilisateur2 aura les même propriété que l'utilisateur1 mise à part le nom.

## Exercices

### Exercice 1

1. Créez une data class nommée Livre qui représente un livre avec les propriétés suivantes :

- titre (String)
- auteur (String)
- anneePublication (Int)
- disponible (Boolean)

2. Créez trois objets de type Livre avec des valeurs différentes.
3. Comparez deux objets pour vérifier s'ils sont identiques (c'est-à-dire que leurs propriétés sont les mêmes).
4. Utilisez la méthode copy() pour créer un nouvel objet Livre en changeant uniquement la valeur de disponible.

## Exercice 2

1. Créez une data class nommée Produit qui possède les propriétés suivantes :
   - nom (String)
   - prix (Double)
   - categorie (String)
2. Créez une liste de 5 produits différents.
3. Trouvez et affichez tous les produits appartenant à une catégorie spécifique (par exemple, "Électronique").

Note : Vous pouvez utiliser la fonction filter d'une liste

4. Utilisez copy() pour créer une nouvelle liste où tous les produits de la catégorie "Électronique" ont une réduction de 10 % sur leur prix.

Note : Vous pouvez utiliser la fonction map d'une liste qui permet d'appliquer une transformation, et retourne une nouvelle collection avec les résultats.

5. Affichez le résultat final dans un Text.

## Exercice 3

1. Créez une data class nommée Contact qui représente un contact avec les propriétés suivantes :
   - nom (String)
   - numero (String)
   - email (String)
2. Créez une liste de contacts avec au moins 5 entrées.
3. Recherchez un contact spécifique par son nom et affichez ses informations.
4. Ajoutez une fonctionnalité pour mettre à jour l'adresse e-mail d'un contact en utilisant la méthode copy().
5. Affichez les informations mises à jour dans un Text.

## Exercice 4

1. Créez une data class nommée Commande qui représente une commande avec les propriétés suivantes :
   - numeroCommande (String)
   - produit (Produit, une autre data class qui correspond à celle de l’exercice 2)
   - quantite (Int)
   - status (String, avec une valeur par défaut "En cours")
2. Créez plusieurs objets Commande en utilisant différents produits.
3. Modifiez le status d'une commande à "Expédiée" en utilisant copy().
4. Affichez le détail mis à jour dans un Text.