Leçon 16
La notion de nullable

Objectif

Nous allons découvrir comment gérer les variables qui peuvent être null en Kotlin. Nous apprendrons à déclarer des variables nullable, utiliser les opérateurs ?, !!, et la fonction let

Les Variables en Kotlin

Déclaration des Variables :
Jusqu'à présent, vous avez appris à déclarer des variables comme ceci :

```
var text = "Super"
```

Dans ce cas, Kotlin peut deviner que text est une chaîne de caractères (String) car vous lui avez donné la valeur "Super". Vous n'avez pas besoin de spécifier le type explicitement, car Kotlin est intelligent et sait que c'est une String.

Déclaration sans Valeur Initiale :
Lorsque vous déclarez une variable sans lui donner de valeur initiale, comme une liste vide, vous devez préciser le type :

```
val animals = mutableListOf<String>()
```

Ici, vous indiquez clairement que animals est une liste mutable de chaînes de caractères (String). Sans cette spécification, Kotlin ne pourrait pas savoir quel type de liste vous voulez créer.

Déclaration avec Type Spécifié :
Vous pouvez aussi spécifier le type même si vous affectez une valeur initiale. Par exemple :

```
var text: String = "Super"
```

Dans ce cas, nous avons précisé que text est de type String. Toutefois, comme nous avons donné une valeur initiale, Kotlin aurait pu deviner le type sans cette spécification. Cependant, cette précision devient importante lorsqu'on veut déclarer des variables qui peuvent accepter des valeurs null.

La notion de nullable

Parfois, une variable peut ne pas avoir de valeur. Par exemple, si vous attendez une réponse ou un message qui peut ne pas arriver, votre variable doit pouvoir être null. Pour gérer ce cas, vous devez déclarer la variable comme nullable.

```
var text: String? = "Super"
```

Ici, text peut contenir une chaîne de caractères ou être null. Vous utilisez String? pour indiquer que text est nullable.

Pourquoi Utiliser Nullable ?

Imaginons que vous ayez une variable qui peut ou non avoir une valeur, comme une réponse d'un ami qui peut encore arriver. Vous devez gérer ce cas pour éviter des erreurs dans votre programme.

Opérateurs et Fonctions pour Variables Nullable

1. **Opérateur de Sécurité ?**
   - Utilisé pour accéder en toute sécurité aux membres d'une variable nullable sans provoquer d'erreur si la variable est null
   - Exemple :

```
var text: String? = "Super"
val length = text?.length
```

   - Ici, text?.length renvoie la longueur de text si text n'est pas null. Si text est null, length sera aussi null.

2. **Opérateur de Non-Null !!**
   - Utilisé lorsque vous êtes certain qu'une variable n'est pas null et que vous voulez accéder à sa valeur sans vérifier.

- Exemple :

```
var text: String? = "Super"
val length = text!!.length
```

- Important : Utiliser **!!** suppose que la variable n'est pas null. Si la variable devient null entre le moment où vous l'initialisez et le moment où vous l'utilisez, votre programme peut planter. Il est préférable de trouver des moyens alternatifs pour éviter les erreurs de null et utiliser !! avec prudence.

3. Opérateur ?: (Elvis Operator)

- Utilisé pour fournir une valeur par défaut si la variable est null.
- Exemple :

```
var text: String? = null
val message = text ?: "Valeur par défaut"
```

- Ici, si text est null, message prendra la valeur "Valeur par défaut". Sinon, il prendra la valeur de text.

4. Instruction let

- Exécute un bloc de code seulement si une variable nullable n'est pas null. Si la variable est null, le bloc ne s'exécute pas.
- Utilisation par rapport à if :
  - Avec if, vous feriez quelque chose comme :

```
var text: String? = "Super"
setContent {
    if (text != null) {
        Text( text: "La valeur de text est $text")
    }
}
```

- Avec let, cela devient plus concis :

```
var text: String? = "Super"
setContent {
    text?.let {
        Text( text: "La valeur de text est $it")
    }
}
```

- Ici, text?.let exécute le bloc seulement si text n'est pas null. La variable it représente text dans le bloc. C'est souvent plus lisible et évite des structures conditionnelles supplémentaires.

Exemples Concrets

1. Exemple avec une Map

Imaginons une Map qui associe des livres à leur nombre de copies :

```
val books = mutableMapOf(
    "Harry Potter" to 10,
    "Le Hobbit" to 5
)
```

Lorsque nous essayons de récupérer une valeur pour une clé qui n'existe pas, elle peut nous renvoyer null. Utilisons l'opérateur ?: pour fournir une valeur par défaut si la clé n'est pas trouvée :

```
val unknownBook = books["Livre Mystère"] ?: "Le livre n'est pas disponible."
```

Si "Livre Mystère" n'est pas présent dans la map, unknownBook prendra la valeur "Le livre n'est pas disponible.".

2. Exemple avec une Liste

Prenons une liste d'animaux :

```
val animals = listOf("Chat", "Chien", "Lapin")
```

Si nous cherchons un animal dans cette liste et qu'il n'est pas trouvé, nous pouvons également fournir une valeur par défaut :

```
val missingAnimal = animals.find { it == "Tigre" } ?: "Animal non trouvé."
```

Si "Tigre" est présent dans la liste, missingAnimal sera "Tigre". Sinon, il prendra la valeur "Animal non trouvé.".

Exercices

Exercice 1

1. Créez une variable userName de type nullable (String?) et assignez-lui la valeur "Alice".
2. Modifiez la variable userName pour qu'elle prenne la valeur null.
3. Utilisez l'opérateur ?: pour afficher le nom de l'utilisateur ou "Utilisateur inconnu" si userName est null.

Exercice 2

1. Créez une liste numbers contenant les nombres 1, 2, 3.
2. Utilisez find pour rechercher le nombre 4 dans la liste, qui n'existe pas.
3. Utilisez l'opérateur ?: pour afficher le nombre trouvé ou "Nombre non trouvé" si le nombre n'existe pas dans la liste.

Exercice 3

1. Créez une map mutable contacts avec les paires clé-valeur suivantes : "John" to "1234", "Jane" to "5678".
2. Recherchez le numéro de téléphone pour la clé "Bob" qui n'existe pas.
3. Utilisez l'opérateur ?: pour afficher le numéro trouvé ou "Numéro non disponible" si "Bob" n'existe pas dans la map.

Exercice 4

1. Créez un tableau fruits contenant les éléments "Pomme", "Banane", "Cerise".
2. Recherchez le fruit "Orange" dans le tableau.
3. Utilisez l'opérateur ?: pour afficher le fruit trouvé ou "Fruit non trouvé" si "Orange" n'est pas présent dans le tableau.

Exercice 5

1. Créez une map mutable books avec les clé-valeur suivantes : "Livre1" to 8, "Livre2" to 5.
2. Recherchez la clé "Livre3" dans la map.
3. Utilisez let pour afficher le nombre de pages si "Livre3" existe, sinon affichez "Livre non trouvé".

Exercice 6

1. Créez une liste mutable nommée animaux contenant les noms de quatre animaux : "Lion", "Tigre", "Ours", "Elephant".
2. Créez une map mutable nommée ages qui associe les noms des animaux aux âges suivants :
   - "Lion" à 5
   - "Tigre" à 3
   - "Ours" à 4
3. Pour chaque animal dans la liste animaux, récupérez son âge dans la map ages. Utilisez l'opérateur ?: pour afficher l'âge de l'animal ou "Âge non disponible" si l'animal n'a pas d'âge enregistré dans la map.