Leçon 15
Les Maps

Objectif

Nous allons explorer les Maps en Kotlin. Une Map est comme une grande boîte avec des tiroirs étiquetés, permettant de stocker des valeurs que vous pouvez retrouver rapidement en utilisant des clés.

Qu'est-ce qu'une Map ?

Une Map est une collection d'éléments où chaque élément est constitué d'une clé et d'une valeur associée. Pensez à une Map comme une boîte de rangement avec des tiroirs étiquetés.

Types de Maps

Une Map est une collection d'éléments où chaque élément est constitué d'une clé et d'une valeur associée. Pensez à une Map comme une boîte de rangement avec des tiroirs étiquetés.

Les Maps en Kotlin se divisent en deux types principaux :

1. **Map Immuable**
   - Une fois créée, vous ne pouvez pas ajouter, modifier ou supprimer des éléments.
   - Création :

```
val contacts = mapOf(
    "Jean" to "123-456",
    "Sophie" to "789-101"
)
```

Ici, nous utilisons le mot clé to pour créer des paires clé-valeur de manière concise. Par exemple, "Jean" to "123-456" associe la clé "Jean" à la valeur "123-456".

2. **Map Mutable**
   - Vous pouvez modifier, ajouter ou supprimer des éléments.
   - Création :

```
val ages = mutableMapOf(
    "Alice" to 25,
    "Bob" to 30
)
```

   De la même manière, "Alice" to 25 associe "Alice" à 25 dans une Map mutable.

Explication :
Pour que ce soit simple à comprendre, quand vous lisez l'instruction, remplacez "to" par "contient", ce qui donnerait "le tiroir étiqueté Alice contient 25"
Techniquement, Alice correspond à la clé et 25 la valeur.

Accès aux Éléments d'une Map

Pour accéder à une valeur en utilisant une clé :

```
val bobAge = ages["Bob"]
```

Explication : Imaginez que vous avez une boîte avec des tiroirs étiquetés. Pour trouver l'âge de Bob, vous cherchez le tiroir étiqueté "Bob" et lisez l'âge à l'intérieur.

Vérifier l'Existence d'une Clé

Pour vérifier si une clé est présente :

```
val hasCharlie = ages.containsKey("Charlie")
```

Explication : Cherchez si le tiroir étiqueté "Charlie" est présent dans la boîte. containsKey vous dit si le tiroir existe.

Manipulations des Maps Mutable

1. **Ajouter un Élément**
- Vous pouvez ajouter un nouvel élément en utilisant une clé et une valeur :

```
val mutableAges = mutableMapOf(
    "Alice" to 25,
    "Bob" to 30
)
mutableAges["David"] = 34  // Ajoute David avec 34 ans
```

2. **Modifier un Élément**
- Vous pouvez modifier la valeur d'une clé existante :

```
mutableAges["Bob"] = 36
```

3. **Supprimer un Élément**
- Vous pouvez supprimer un élément en utilisant sa clé :

```
mutableAges.remove( key: "Bob")  // Supprime Bob de la Map
```

Exercices

Exercice 1

1. Créez une Map nommée contacts contenant les paires suivantes :
   - "Jean" -> "123-456"
   - "Sophie" -> "789-101"
2. Affichez le numéro de téléphone de "Sophie" dans le format suivant :
   "Le numéro de Sophie est : 789-101."

Exercice 2

1. Créez une Map nommée produits contenant les paires suivantes :
   - "Pommes" -> 3.50
   - "Bananes" -> 2.20
   - "Oranges" -> 4.00
2. Ajoutez un produit "Poires" à 3.00 à la Map.
3. Affichez tous les produits et leurs prix sous la forme :
   "Produits : Pommes - 3.50, Bananes - 2.20, Oranges - 4.00, Poires - 3.00."

Note : Pour l'affichage de la map, vous pouvez utiliser l'instruction suivante :

```
produits.entries.joinToString( separator: ", ") { "${it.key} - ${it.value}" }}
```

Explication :

1. produits.entries : Imagine que chaque élément de la Map produits est comme une petite carte avec deux côtés : un côté pour le nom du produit (la clé) et un côté pour son prix (la valeur). L'ensemble de ces cartes est contenu dans entries.
2. joinToString(", ") : Ensuite, on prend toutes ces petites cartes et on les aligne en une seule ligne. Pour séparer chaque carte, on place une virgule (", ") entre elles, comme si on écrivait une liste d'éléments.
3. { "${it.key} - ${it.value}" } : À chaque carte, on décide comment afficher ses deux côtés. Pour chaque carte (ou entrée), it est le mot magique qui nous permet d'accéder à la clé (nom du produit) et à la valeur (prix du produit). On choisit de les montrer sous la forme "nom - prix", comme ceci : "Pommes - 3.50".

Cette instruction prend tous les produits de la Map, les transforme en texte sous la forme "nom - prix", et les enchaîne en une seule phrase, avec des virgules pour les séparer

Exercice 3

1. Créez une Map nommée éléments avec les paires suivantes :

- "Hydrogène" -> 1
- "Hélium" -> 2
- "Carbone" -> 6

2. Supprimez "Hélium" de la Map.
3. Affichez les éléments restants sous la forme :
   "Éléments : Hydrogène - 1, Carbone - 6."

Exercice 4

1. Créez une Map nommée inventaire avec les paires suivantes :
   - "Chaussures" -> 10
   - "Chapeaux" -> 5
   - "Gants" -> 7
   - "Chaussettes" -> 4
2. Vérifiez si "Chapeaux" est dans la Map et affichez un texte indiquant sa présence : "Les Chapeaux sont dans l'inventaire." sinon affichez "Les Chapeaux ne sont pas dans l'inventaire."

Exercice 5

1. Créez une Map nommée notes avec les paires suivantes :
   - "Mathématiques" -> 15
   - "Français" -> 12
   - "Histoire" -> 18
2. Modifiez la note de "Français" à 14.
3. Affichez les notes sous la forme :
   "Notes : Mathématiques - 15, Français - 14, Histoire - 18."

Exercice 6

1. Créez une Map nommée prix avec les paires suivantes :
   - "Lait" -> 1.20
   - "Pain" -> 0.90
   - "Fromage" -> 2.50
2. Modifiez le prix du "Pain" à 1.00 et ajoutez un nouveau produit "Beurre" à 1.50.
3. Affichez les prix sous la forme :
   "Prix : Lait - 1.20, Pain - 1.00, Fromage - 2.50, Beurre - 1.50."

Exercice 7

1. Créez une Map nommée villes avec les paires suivantes :
   - "Paris" -> 2148000
   - "Londres" -> 8982000
   - "Tokyo" -> 13929000
2. Affichez toutes les valeurs de la Map sous la forme :
   "Populations : 2148000, 8982000, 13929000."