Sujet de TP N°1 - Manipulation de collections en Kotlin (Android)

Modalité de remise :

- **Non noté**

VUE D'ENSEMBLE

Dans ce TP, vous allez compléter des méthodes qui manipulent des collections pour obtenir et afficher des résultats spécifiques. Vous utiliserez plusieurs fonctions d'ordre supérieur fournies par Kotlin. Ce TP peut être réalisé dans Android Studio ou directement sur le Kotlin Playground.

OBJECTIFS

- **Apprendre à utiliser les fonctions d'ordre supérieur** : vous allez découvrir comment manipuler et transformer des collections grâce à des fonctions comme `forEach`, `map`, `filter`, `groupBy`, `fold` et `sortedBy`.
- **Développer des compétences en manipulation de données** : à partir d'une liste d'objets `User`, vous effectuerez des transformations et des analyses pour extraire des informations pertinentes.
- **Renforcer la pratique de Kotlin** : en travaillant directement sur des exemples concrets, vous apprendrez à mieux utiliser les collections et les expressions lambda.

Contexte

Vous disposez d'une liste d'utilisateurs avec les propriétés suivantes :

- **name** : nom de l'utilisateur (de type `String`)
- **age** : âge de l'utilisateur (de type `Int`)
- **city** : ville de l'utilisateur (de type `String`)

Voici la liste de départ :

```
data class User(val name: String, val age: Int, val city: String)

val users = listOf(

    User("Alice", 25, "Paris"),
```

```
    User("Bob", 30, "Lyon"),
    User("Charlie", 35, "Marseille"),
    User("David", 28, "Paris"),
    User("Eve", 40, "Lyon")
)
```

1. **Utiliser *forEach***

   Compléter la méthode pour afficher tous les utilisateurs.

   ```
   fun printUsers(users: List<User>) {
       // Complétez le corps ici
   }
   ```

2. **Utiliser *map***

   Transformer la liste d'utilisateurs en une liste de leurs noms.

   ```
   fun getUserNames(users: List<User>): List<String> {
       // Complétez le corps ici
   }
   ```

3. **Utiliser *filter***

   Récupérer uniquement les utilisateurs ayant plus de 30 ans.

   ```
   fun getUsersOver30(users: List<User>): List<User> {
       // Complétez le corps ici
   }
   ```

4. Utiliser *groupBy*

Grouper les utilisateurs par ville.

```
fun groupUsersByCity(users: List<User>): Map<String, List<User>>
{

   // Complétez le corps ici

}
```

5. Utiliser *fold*

Calculer l'âge total de tous les utilisateurs.

```
fun getTotalAge(users: List<User>): Int {

   // Complétez le corps ici

}
```

6. Utiliser *sortedBy*

Trier les utilisateurs par âge.

```
fun sortUsersByAge(users: List<User>): List<User> {

   // Complétez le corps ici

}
```

7. Calculer la moyenne d'âge

Compléter la méthode pour calculer la moyenne d'âge des utilisateurs.

```
fun getAverageAge(users: List<User>): Double {

   // Complétez le corps ici

}
```

8. Trouver l'utilisateur le plus âgé

Compléter la méthode pour retourner l'utilisateur le plus âgé.

```
fun getOldestUser(users: List<User>): User? {

   // Complétez le corps ici

}Trier les utilisateurs par nom
```

9. Trier la liste des utilisateurs par ordre alphabétique des noms.

```
fun sortUsersByName(users: List<User>): List<User> {

   // Complétez le corps ici

}
```

10. Trier les utilisateurs par ville puis par âge

Trier les utilisateurs d'abord par ville, puis par âge.

```
fun sortUsersByCityAndAge(users: List<User>): List<User> {

   // Complétez le corps ici

}
```