TD : Manipulation des Collections en Java

Objectifs :

- Comprendre les différentes collections (`List`, `Set`, `Queue`, `Map`, `Deque`) et les appliquer en Java.
- Développer des solutions adaptées à chaque type de collection en fonction des besoins spécifiques.

Exercice 1 : Utilisation de `List` pour Analyser des Données Météo

Objectif : Utiliser une `List` pour traiter des séquences de données et déterminer la plus longue occurrence consécutive d'un même type.

1. **Création de la Classe `WeatherOccurrence`**
    - Créez une classe `WeatherOccurrence` avec les attributs :
        - `String weather` : type de météo (pluie, soleil, etc.)
        - `int occurrences` : nombre de fois où ce type de météo apparaît consécutivement
        - `int startIndex` : indice de départ de cette séquence.
2. **Implémentation de la Méthode de Recherche**
    - Implémentez une méthode `WeatherOccurrence longestSequenceOfSameWeather(List<String> weather)` qui prend une liste de types de météo en paramètre et identifie la séquence la plus longue de jours consécutifs avec le même type de météo.
    - La méthode doit retourner un objet `WeatherOccurrence` représentant la météo, le nombre d'occurrences et l'indice de début.
3. **Exemple de Test**
    - Utilisez la liste suivante pour tester votre méthode : `["Rain", "Sun", "Rain", "Rain", "Hail", "Snow", "Storm"]`.
    - Affichez le type de météo, le nombre de jours consécutifs, et l'indice de départ de la séquence la plus longue.

Exercice 2 : Comparaison de Loisirs avec un `Set`

Objectif : Utiliser un `Set` pour identifier les éléments communs et calculer un pourcentage de similitude.

1. **Création des Ensembles de Loisirs**
    - Déclarez deux `Set<String>` pour représenter les loisirs de deux personnes.

    - Ajoutez quelques loisirs (ex. "Camping", "Pêche", "Lecture") dans chaque ensemble, en incluant certains loisirs communs.
2. **Calcul de la Similitude**
    - Écrivez une méthode qui prend deux ensembles de loisirs en paramètres et :
        - Affiche les intérêts communs.
        - Calcule et affiche le pourcentage de similarité en divisant le nombre d'intérêts communs par le nombre total d'intérêts uniques.
3. **Exemple de Test**
    - Testez votre méthode avec différents ensembles de loisirs et affichez le pourcentage de compatibilité.

Exercice 3 : Simulation d'un Calculateur RPN avec une `Queue`

Objectif : Utiliser une `Queue` pour évaluer une expression en notation polonaise inverse (RPN).

1. **Introduction à la Notation Polonaise Inverse (RPN)**
    - En RPN, les opérateurs suivent leurs opérandes. Par exemple, `3 4 +` signifie `3 + 4`, et `3 4 + 5 *` signifie `(3 + 4) * 5`.
2. **Implémentation de la Méthode**
    - Créez une méthode `evaluateRPN(String expression)` qui prend une chaîne RPN en paramètre, la décompose en tokens et utilise une pile (ou `Queue`) pour évaluer l'expression.
    - Pour chaque nombre, ajoutez-le dans la pile. Pour chaque opérateur, retirez les deux derniers éléments de la pile, appliquez l'opération, et replacez le résultat dans la pile.
3. **Exemple de Test**
    - Utilisez l'expression `"12 34 23 + *"` pour tester votre méthode, et vérifiez que le résultat est correct.

Exercice 4 : Conversion de Tableaux 2D en `Map`

Objectif : Convertir un tableau en `Map` pour gérer des paires clé-valeur uniques.

1. **Création de la Méthode de Conversion**
    - Créez une méthode `Map<String, String> convertToMap(String[][] array)` qui prend un tableau bidimensionnel et le convertit en `Map`.
    - Dans le tableau, la première colonne représente la clé et la seconde colonne représente la valeur.
2. **Gestion des Duplication**
    - Si une clé apparaît plusieurs fois dans le tableau, la dernière occurrence doit écraser les valeurs précédentes.

3. **Exemple de Test**
    - Utilisez un tableau tel que `{{"Rouge", "#FF0000"}, {"Vert", "#00FF00"}, {"Rouge", "#FF1111"}}` et vérifiez que le résultat final dans la `Map` est `{Rouge=#FF1111, Vert=#00FF00}`.

Exercice 5 : Gestion d'une Rampe de Chargement avec un `Deque`

Objectif : Utiliser une `Deque` pour simuler une file d'attente de chargement.

1. **Mise en Place de la Structure de Données**
    - Utilisez une `ArrayBlockingQueue<String>` de capacité 5 pour représenter une rampe de chargement.
2. **Création des Tâches de Chargement et de Déchargement**
    - Implémentez deux classes `Loader` et `Unloader` qui ajoutent et retirent des éléments de la rampe respectivement.
    - Le `Loader` doit ajouter des articles de manière aléatoire dans la `Deque` avec un délai entre 1 et 2 secondes.
    - Le `Unloader` retire les articles de la rampe avec le même délai.
3. **Exécution des Tâches**
    - Créez cinq threads `Unloader` et dix threads `Loader` pour tester le système de chargement et de déchargement simultané.