TP 6

M.Adam – N.Delomez – JF.Kamp – L.Naert

10 août 2022

Objectifs du TP

- Construire des boucles imbriquées

Exercice 1 (reprise du TD) (**)

La méthode `saisirEtTrier()` a pour objectif d'insérer la valeur saisie dans le tableau. Les valeurs doivent être insérées de manière à ce que le tableau soit trié dans l'ordre croissant.

1. Compléter la boucle interne du programme. Évidemment, la méthode de construction des boucles doit être utilisée.

```
/**
 * Crée et saisit un tableau trié de LG_TAB entiers
 * @return tableau trié de LG-TAB entiers
 */
int[] saisirEtTrier () {
    int[] t = new int[LG_TAB];
    int i = 0;

    while (i < t.length) {
        t[i] = SimpleInput.getInt ("Entrer un entier");
        // insertion de la valeur en ordre croissant dans t
        i = i + 1;
    }
    return t;
}
```

2. **Rendre la conception méthodique de la méthode, son code, le code de la méthode de test et les tests d'éxécution.**

Exercice 2 (**)

La méthode `sontTousDiff()` vérifie si les deux tableaux n'ont aucun entier en commun. La méthode rend vrai si les 2 tableaux n'ont aucune valeur commune et rend faux s'il existe au moins une valeur présente dans les deux tableaux.

```
/**
 * vérifie si deux tableaux n'ont aucune valeur commune
 * @param tab1 premier tableau
 * @param tab2 deuxième tableau
 * @return vrai si les tableaux tab1 et tab2 n'ont aucune valeur commune, faux sinon
 */
boolean sontTousDiff (int[] tab1, int[] tab2)
```

1. Écrire de manière méthodique le code de la méthode `sontTousDiff()`.
2. **Rendre la conception méthodique de l'algorithme, le code de la méthode, le code de la méthode de test et les tests d'éxécution.**

Exercice 3 (****)

La méthode `eliminerDouble()` élimine, les valeurs en double d'un tableau. Ainsi, si la valeur 12, par exemple, est présente en 3 exemplaires, la méthode élimine 2 valeurs et rend un tableau avec une seule fois 12.

Évidemment, la taille du tableau, le nombre d'éléments, est modifiée en fonction des valeurs éliminées. Ainsi si 5 valeurs ont été supprimées, la taille est diminuée de 5.

```
/**
 * élimine les valeurs en plusieurs exemplaires dans un tableau
 * un élément présent plusieurs fois n'est plus qu'en un seul exemplaire
 * @param tab    tableau d'entiers
 * @return le nombre d'éléments du tableau sans double
 */
int eliminerDouble(int[] tab)
```

Par exemple, `eliminerDouble ({0, 0, 2, 3, 0, 2, 1, 3, 3, 0})` rend la valeur 4, car il existe 4 valeurs distinctes et un tableau `{0, 2, 3, 1, 0, 2, 1, 3, 3, 0}`. Attention, dans ce tableau seules les valeurs des indices 0 à 3 sont valides, les autres valeurs sont ignorées. Les valeurs distinctes sont placées en début du tableau passé en paramètre. L'ordre des valeurs dans le tableau après modification n'a pas d'importance.

1. Écrire de manière méthodique le code de la méthode `eliminerDouble()`.
2. **Rendre la conception méthodique de l'algorithme, le code de la méthode, le code de la méthode de test et les tests d'éxécution.**

Exercice 4 (***)

1. Écrire de manière méthodique une méthode `estSousChaine()` qui détermine si une chaîne est présente dans une autre.

```
/**
 * teste si une chaîne est une sous-chaîne d'une autre
 * @param mot    chaîne de caractères
 * @param phrase chaîne de carectères
 * @return vrai ssi la première chaîne est présente dans la seconde
 */
boolean estSousChaine (String mot, String phrase)
```

   La comparaison s'effectue caractère par caractère.

   Répondre aux questions :

   - la chaîne `ses` est-elle dans `abcdsesdef` ?
   - la chaîne `ses` est-elle dans `abcdef` ?
   - la chaîne `ses` est-elle dans `abcdefse` ?

2. **Rendre la conception méthodique de l'algorithme, le code de la méthode, la méthode de test et les tests d'exécution.**