Haute École de Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

2018 – 2019
DEV1

DEV1 – Laboratoires Java I

TD 10 – Tableaux

Dans ce TD vous trouverez une introduction aux tableaux.

Les codes sources et les solutions de ce TD se trouvent à l'adresse :

https://git.esi-bru.be/dev1/labo-java/tree/master/td10-tableaux

Table des matières

Version du document
31 octobre 2018

1 Créer et manipuler des tableaux

Un tableau est une collection indexée d'éléments de même type. Un tableau est crée avec une taille donnée et ne peut plus en changer par la suite. Dans l'exemple ci-dessous nous créons et manipulons un tableau de taille 5 contenant des entiers. Comme cela est illustré dans l'exemple, on accède aux éléments d'un tableau par son indice.

```java
package esi.dev1.td10;

/**
 * Exemple d'utilisations des tableaux
 * @author fservais
 */
public class Tableau {

   /**
    * Exemple d'utilisations de tableaux d'entier.
    * @param args pas utilisé.
    */
   public static void main(String[] args) {
      int[] tab = {1, 2, 3, 4, 5};
      System.out.println("taille: "+tab.length);
      System.out.println("1er élément: "+tab[0]);
      System.out.println("dernier élément: "+tab[tab.length-1]);
      System.out.println("autre élément: "+tab[2]);
      System.out.println("affiche un drôle de truc: "+tab);

      afficherTab(tab);
      incrémenter(tab);
      afficherTab(tab);

      int[] tab2 = new int[10];
      afficherTab(tab2);
   }

   /**
    * Affiche un tableau.
    * @param tab le tableau à afficher
    */
   static void afficherTab(int[] tab) {
      for (int valeur : tab) {
         System.out.print(valeur+" ");
      }
      System.out.println(); // on passe à la ligne
   }

   /**
    * Incrémente chaque composante du tableau.
    * @param tab le tableau à incrémenter.
    */
   static void incrémenter(int[] tab) {
      for (int i = 0; i < tab.length; i++) {
         tab[i]++;
      }
   }
}
```

Tableau.java

Exercice 1 **Doubler**

Reprenez le code ci-dessus et ajoutez-y une méthode `void multiplier(int[] tab, int valeur)` qui multiplie chaque élément du tableau par la valeur donnée.

Exercice 2 **Afficher**

Modifiez la méthode `afficherTab` afin que l'affichage soit par exemple pour un tableau contenant les valeurs 1, 2, 3, 4, 5 :

```
[1, 2, 3, 4, 5]
```

2 Tableaux de différents types

Les tableaux peuvent contenir des éléments de tous les types. L'exemple suivant illustre l'utilisation d'un tableau de `String`.

```java
package esi.dev1.td10;

/**
 * Exemple d'utilisation de tableaux de chaines
 * @author fservais
 */
public class TableauChaines {
    public static void main(String[] args) {
        String[] mots = {"the", "quick", "brown", "fox", "jumps", "over", "the",
            "lazy", "dog"};

        for (String mot : mots) { // pour chaque mot dans le tableau 'mots'
            System.out.print(mot+" "); // affiche le mot suivi d'un espace.
        }
        System.out.println(); // passe à la ligne.
        afficheTableau(mots); // la même chose mais dans une méthode.
    }

    static void afficheTableau(String[] mots) {
        for (String mot : mots) {
            System.out.print(mot+" ");
        }
        System.out.println();
    }
}
```

TableauChaines.java

Exercice 3 **Les tailles**

Complètez l'exemple ci-dessus en y ajoutant une méthode `void afficherTailles(String[] tab)` qui affiche la taille de chaque `String` d'un tableau. Par exemple si le tableau est celui de l'exemple la méthode affiche 3 5 5 3 5 4 3 4 3.

Exercice 4 **Le plus long mot**

Toujours dans l'exemple, ajoutez une méthode `String plusLongMot(String[] tab)` qui reçoit un tableau de `String` et retourne le plus long mot. S'il y a plusieurs mots les plus longs, la méthode retournera n'importe lequel parmi ceux-ci. Dans l'exemple ci-dessus la méthode retournera `quick`, `brown` ou `jumps`.

3 Retourner un tableau

Il est bien sûr possible qu'une méthode retourne un tableau. Dans l'exemple suivant la méthode `copie` effectue une copie d'un tableau reçu en paramètre. La méthode `pasUneCopie` illustre une erreur fréquente lorsque l'on débute dans l'apprentissage des tableaux en Java.

```java
package esi.dev1.td10;

/**
 * Exemples de méthodes retournant un tableau.
 * @author fservais
 */
public class TableauCopie {

   public static void main(String[] args) {
      int[] tableau = {1, 2, 3, 4, 5};
      afficherTab(tableau);
      int[] copie1 = copie(tableau);
      int[] copie2 = pasUneCopie(tableau);
      tableau[0] = 9;
      System.out.print("tableau: " );
      afficherTab(tableau);
      System.out.print("copie1: " );
      afficherTab(copie1);
      System.out.print("copie2: " );
      afficherTab(copie2);
   }

   static int[] copie(int[] tab) {
      int[] copie = new int[tab.length];
      for (int i = 0; i<tab.length; i++) {
         copie[i] = tab[i];
      }
      return copie;
   }
   static int[] pasUneCopie(int[] tab) {
      int[] copie = tab;
      return copie;
   }
   static void afficherTab(int[] tab) {
      for (int element : tab) {
          System.out.print(element+" ");
      }
      System.out.println();
   }
}
```

TableauCopie.java

Exercice 5 **Miroir**

Complètez l'exemple ci-dessus en y ajoutant une méthode `int miroir(int[] tab)` qui retourne un nouveau tableau dont les éléments sont les éléments du tableau reçu en paramèter mais en ordre inverse. Par exemple si le tableau est celui de l'exemple la méthode retourne un tableau contenant les entiers 5, 4, 3, 2, 1.

4 Tableaux et tests

La manipulation des tableaux n'est pas aisée, les tests JUnit sont un outil essentiel afin de s'assurer de la correction de nos méthodes.

La méthode `assertArrayEquals` de la classe `org.junit.Assert` permet de s'assurer que deux tableaux sont strictement égaux : de même taille, de même type et que tous leurs éléments sont égaux et apparaissent dans le même ordre.

```java
package esi.dev1.td10;

import org.junit.Test;
```

```java
import static org.junit.Assert.*;

public class TableauTest {

   @Test
   public void testIncrémenter() {
      int[] tab = {1, 2, 3};
      Tableau.incrémenter(tab);
      int[] tabAttendu = {2, 3, 4};
      assertArrayEquals(tabAttendu, tab);
   }

   @Test
   public void testIncrémenterTableauVide() {
      int[] tab = {};
      Tableau.incrémenter(tab);
      int[] tabAttendu = {};
      assertArrayEquals(tabAttendu, tab);
   }

}
```

TableauTest.java

Exercice 6 Tests

Créer des tests JUnit pour les méthodes de les exercices 1, 4 et 5. Pour chacune de ces méthodes testez un cas général ainsi que les cas limites (tableau vide, maximum en début/fin de tableau, valeurs négatives, etc).

5 Exercices récapitulatifs

Exercice 7 `TableauUtil`

Dans une classe `TableauUtil` écrivez les méthodes suivantes, leur javadoc ainsi que les tests JUnit correspondant :

- `static double min(double[] tab)` qui retourne le minimum du tableau passé en paramètre ;
- `static double max(double[] tab)` qui retourne le maximum du tableau ;
- `static double somme(double[] tab)` qui retourne la somme des éléments du tableau ;
- `static double moyenne(double[] tab)` qui retourne la moyenne des éléments du tableau ;
- `static double[] copie(double[] tab)` qui retourne une copie du la tableau passé en paramètre ;

Exercice 8 `TableauUtil` (suite)

Dans la classe `TableauUtil` ajoutez les méthodes suivantes, leur javadoc ainsi que les tests JUnit correspondant :

- `int indiceMax(double[] tab)` : retourne l'indice du maximum du talbeau passé en paramètre ;
- `boolean estTrié(double[] tab)` : retourne vrai si le tableau est trié par ordre croissant et faux sinon ;
- `int indice(int[] tab, int valeur)` : retourne l'indice de cette valeur dans le tableau. Que retournez-vous si la valeur n'apparait pas ?

- ▷ `boolean contient(String[] tab, String mot)` : retourne vrai si un tableau de `String` contient un mot passé en paramètre ;

  Astuce : utiliser la méthode `equals` afin de tester l'égalité entre deux `String` et non pas l'opérateur '=='.
- ▷ `boolean contient(double[] tab, double valeur, double erreur)` : retourne vrai si un élément du tableau est proche de la `valeur` à l'`erreur` près, c'est-à-dire si la distance entre l'élément et la valeur est inférieure à `erreur`.

Exercice 9 `TableauUtil` (suite)

Dans la classe `TableauUtil` ajoutez les méthodes suivantes, leur javadoc ainsi que les tests JUnit correspondant :

- ▷ `static void inverser(double[] tab)` qui inverse l'ordre des éléments du tableau passé en paramètre ;
- ▷ `static boolean doublons(int[] tab)` qui retourne vrai si le tableau possède des doublons (2 fois la même valeur) et faux sinon ;
- ▷ `static int nbNégatifs(double[] tab)` qui retourne le nombre d'éléments négatifs du tableau ;
- ▷ `static void échange(double[] tab, indice1, indice2)` qui échange la valeura se trouvant à l'indice `indice1` avec la valeur se trouvant à l'indice `indice2` ;
- ▷ `static int[] indicesMin(double[] tab)` qui retourne un tableau conteant les indices des minimums.

Exercice 10 Initialisation par défaut

Dans la méthode principale d'une classe `TestInit` :

- ▷ créer un tableau d'entier de taille 10 et afficher-le (comme à la ligne 18 de la classe `Tableau` ci-dessus).

  Quelle valeur par défaut initialise chacune des cases du tableau ?
- ▷ créer un tableau de double de taille 10 et afficher-le ;
- ▷ créer un tableau de 10 booléens et afficher-le ;
- ▷ créer un tableau de 10 `String` et afficher-le.

Exercice 11 `Tri`

Dans la classe `Tri` ajoutez les méthodes suivantes, leur javadoc ainsi que les tests JUnit correspondant :

- ▷ `static void triBulle(int[] tab)` qui trie le tableau en utilisant le tri bulle vu au cours ;
- ▷ `static void triSélection(int[] tab)` qui trie le tableau en utilisant le tri par sélection vu au cours ;
- ▷ `static void triInsertion(int[] tab)` qui trie le tableau en utilisant le tri par insertion vu au cours ;
- ▷ `static boolean rechercheDichotomique(int[] tab, int valeur )` qui retourne vrai si l'élément se trouve dans le tableau trié et faux sinon. On suppose que le tableau reçu est trié. Vous utiliserez l'algorithme de recherche dichotomique vu au cours.