TD N°05 JAVA

Exercise 1 :

```
public class MaximumTableau {

  public static int maxTableau(int[] tableau) {
    if (tableau.length == 0) {
      throw new IllegalArgumentException("Un tableau vide n'a pas de
maximum");
    }
    int max = tableau[0];
    for (int i = 1; i < tableau.length; i++) {
      if (tableau[i] > max) {
        max = tableau[i];
      }
    }
    return max;
  }

  public static void main(String[] args) {
    int[] t = //{1, 2, 5, 89, 75, 10};
    // {};
      {8};
    System.out.println(maxTableau(t));
  }
}
```

Exercise 2 :

```
public class Tableau {
  private static final int VIDE = -1;

  /**
   * Afficher les éléments d'un tableau d'entiers
   * @param tableau le tableau dont on affiche les éléments
   */
  public static void afficheTableau(int[] tableau) {
    for (int i = 0; i < tableau.length; i++) {
      System.out.print(tableau[i] + " ; ");
    }
  }

  /**
   * Afficher les éléments d'un tableau d'entiers
   * @param tableau le tableau dont on affiche les éléments
   */
  public static void afficheTableau2(int[] tableau) {
    for (int val : tableau) {
      System.out.print(val + " ; ");
    }
  }

  /**
   * Afficher les éléments "non vides" d'un tableau d'entiers
   * (-1 correspond à un élément vide).
   * @param tableau le tableau dont on affiche les éléments
   */
  public static void afficheTableau3(int[] tableau) {
    int i = 0;
    while (i < tableau.length && tableau[i] != VIDE) {
      System.out.print(tableau[i] + " ; ");
      i++;
    }
  }

  public static void main(String[] args) {
    int[] t = {1, 2, 3, 4, -1, -1};
    afficheTableau(t);
    System.out.println();
    afficheTableau2(t);
    System.out.println();
    afficheTableau3(t);
  }

}
```

Exercise 3 :

```
public class Tableau {
  private static final int VIDE = -1;

  /**
   * Afficher les éléments d'un tableau d'entiers
   * @param tableau le tableau dont on affiche les éléments
   */
  public static void afficheTableau(int[] tableau) {
    for (int i = 0; i < tableau.length; i++) {
      System.out.print(tableau[i] + " ; ");
    }
  }

  /**
   * Afficher les éléments d'un tableau d'entiers
   * @param tableau le tableau dont on affiche les éléments
   */
  public static void afficheTableau2(int[] tableau) {
    for (int val : tableau) {
      System.out.print(val + " ; ");
    }
  }

  /**
   * Afficher les éléments "non vides" d'un tableau d'entiers
   * (-1 correspond à un élément vide).
   * @param tableau le tableau dont on affiche les éléments
   */
  public static void afficheTableau3(int[] tableau) {
    int i = 0;
    while (i < tableau.length && tableau[i] != VIDE) {
      System.out.print(tableau[i] + " ; ");
      i++;
    }
  }

  public static void initialise(int[] tableau) {
    for (int i = 0; i < tableau.length; i++) {
      tableau[i] = VIDE;
    }
  }

  public static boolean ajouterElement(int[] tableau, int element) {
    // Trouver la 1ère place vide pour ajouter l'élément
    int i = 0;
    // Remarquez le raccourci avec &&.
    // Essayez par exemple d'intervertir les conditions.
    while (i < tableau.length && tableau[i] != VIDE) {
      i++;
    }
    // Attention, l'indice le plus élevé d'un tableau à n élement est n - 1
    if (i == tableau.length) {
      return false;
    }
    tableau[i] = element;
    return true;
  }

  public static void remplir(int[] tableau, int element) {
```

```
    initialise(tableau);
    // Le corps du while est vide (";" suit la condition).
    while (ajouterElement(tableau, element));
  }

  /**
   * Teste la méthode remplir.
   * @param tableau
   * @param element
   * @return
   */
  public static boolean testRemplir(int[] tableau, int element) {
    remplir(tableau, element);
    for (int valeur : tableau) {
      if (valeur != element) {
        return false;
      }
    }
    return true;
  }

  /**
   * @param args
   */
  public static void main(String[] args) {
    int[] t = {1, 2, 3, 4, -1, -1};
    ajouterElement(t, 8);
    afficheTableau(t);
    System.out.println();
    ajouterElement(t, 15);
    afficheTableau(t);
    System.out.println();
    if (! ajouterElement(t, 1500)) {
      System.out.println("Tableau plein !");
    }
    afficheTableau(t);
    System.out.println();
    remplir(t, 250);
    afficheTableau(t);
    System.out.println();
    if (testRemplir(t, 1000)) {
      System.out.println("OK");
    }
    else {
      System.out.println("Pas OK !");
    }
  }
}
```

Exercise 4 :

```
import java.util.Arrays;

public class Tableau {
  private static final int VIDE = -1;

  /**
   * Afficher les éléments d'un tableau d'entiers
   * @param tableau le tableau dont on affiche les éléments
   */
  public static void afficheTableau(int[] tableau) {
    for (int i = 0; i < tableau.length; i++) {
      System.out.print(tableau[i] + " ; ");
    }
  }

  /**
   * Afficher les éléments d'un tableau d'entiers
   * @param tableau le tableau dont on affiche les éléments
   */
  public static void afficheTableau2(int[] tableau) {
    for (int val : tableau) {
      System.out.print(val + " ; ");
    }
  }

  /**
   * Afficher les éléments "non vides" d'un tableau d'entiers
   * (-1 correspond à un élément vide).
   * @param tableau le tableau dont on affiche les éléments
   */
  public static void afficheTableau3(int[] tableau) {
    int i = 0;
    while (i < tableau.length && tableau[i] != VIDE) {
      System.out.print(tableau[i] + " ; ");
      i++;
    }
  }

  public static void initialise(int[] tableau) {
    for (int i = 0; i < tableau.length; i++) {
      tableau[i] = VIDE;
    }
  }

  public static boolean ajouterElement(int[] tableau, int element) {
    // Trouver la 1ère place vide pour ajouter l'élément
    int i = 0;
    // Remarquez le raccourci avec &&.
    // Essayez par exemple d'intervertir les conditions.
    while (i < tableau.length && tableau[i] != VIDE) {
      i++;
    }
    // Attention, l'indice le plus élevé d'un tableau à n élement est n - 1
    if (i == tableau.length) {
      return false;
    }
    tableau[i] = element;
    return true;
  }
```

```
public static void remplir(int[] tableau, int element) {
  initialise(tableau);
  // Le corps du while est vide (";" suit la condition).
  while (ajouterElement(tableau, element));
}

/**
 * Teste la méthode remplir.
 * @param tableau
 * @param element
 * @return
 */
public static boolean testRemplir(int[] tableau, int element) {
  remplir(tableau, element);
  for (int valeur : tableau) {
    if (valeur != element) {
      return false;
    }
  }
  return true;
}

public static int rechercher(int[] tableau, int element) {
  int i = 0;
  while (i < tableau.length && tableau[i] != VIDE) {
    if (tableau[i] == element) {
      return i;
    }
    else {
      i++;
    }
  }
  return -1;
}

/**
 * Recherche toutes les occurrences d'une valeur dans un tableau
 * @param tableau le tableau dans lequel on cherche
 * @param valeurCherchee la valeur cherchée
 * @return un tableau de la longueur du tableau passé en paramètre
 * qui contient les indices où valeur a été trouvée
 * (les cases vides de la fin contiennent -1).
 */
public static int[] rechercherTous(int[] tableau, int element) {
  // Le tableau qui contient les positions de element dans tableau
  int [] positionsTrouvees = new int[tableau.length];
  initialise(positionsTrouvees);
  int i = 0;
  while (i < tableau.length && tableau[i] != VIDE) {
    if (tableau[i] == element) {
      ajouterElement(positionsTrouvees, i);
    }
    i++;
  }
  return positionsTrouvees;
}

/**
 * Recherche toutes les occurrences d'une valeur dans un tableau.
```

```
   * Manière classique de faire (ne tient pas compte d'une valeur spéciale
"vide").
   * @param tableau le tableau dans lequel on cherche
   * @param valeurCherchee la valeur cherchée
   * @return un tableau complètement rempli
   * qui contient les indices où valeur a été trouvée.
   */
  public static int[] rechercherTousBis(int[] tableau, int valeurCherchee)
{
    // Le tableau qui contient les positions de element dans tableau
    int [] positionsTrouvees = new int[tableau.length];
    int i = 0;
    // Nombre d'éléments du tableau égaux à valeurCherchee
    int nbPositionsTrouvees = 0;
    while (i < tableau.length && tableau[i] != VIDE) {
      if (tableau[i] == valeurCherchee) {
        positionsTrouvees[nbPositionsTrouvees++] = i;
      }
      i++;
    }
    // Il faut maintenant retourner un tableau complètement rempli
    int[] tableauIndices = new int[nbPositionsTrouvees];
    // Recopie les positions trouvées dans le tableau que l'on va retourner
//    for (int j = 0; j < nbPositionsTrouvees; j++) {
//      tableauIndices[j] = positionsTrouvees[j];
//    }
    // Autre façon de faire qui utilise Arrays :
    return Arrays.copyOf(positionsTrouvees, nbPositionsTrouvees);
  }

  /**
   *
   * @param tableau tableau trié dans lequel element est cherché.
   * @param element élément cherché.
   * @return la 1ère position du tableau qui contient element.
   * -1 si element n'est pas trouvé dans le tableau.
   */
  public static int rechercherTableauTrie(int[] tableau, int element) {
    int i = 0;
    while (i < tableau.length && tableau[i] != VIDE) {
      if (tableau[i] == element) {
        return i;
      }
      // Si on est ici c'est que t[i] est différent de element
      if (tableau[i] > element) {
            // On a dépassé la valeur de element ; comme le tableau est
trié,
      // inutile d'aller plus loin
      return -1;
      }
      // On n'a pas dépassé la valeur de element ; on va plus loin
      i++;
    }
    // Pour le cas où le dernier élément du tableau est plus petit que
element
    return -1;
  }

  public static boolean croissant(int[] tableau) {
    int valeurPrecedente = -1;
    for (int valeur : tableau) {
```

```
      if (valeur == VIDE) {
        return true;
      }
      if (valeur < valeurPrecedente) {
        return false;
      }
      valeurPrecedente = valeur;
    }
    return true;
  }

  public static int rechercher2(int[] tableau, int element) {
    if (croissant(tableau)) {
      return rechercherTableauTrie(tableau, element);
    }
    else {
      return rechercher(tableau, element);
    }
  }

  /**
   * @param args
   */
  public static void main(String[] args) {
    int[] t = {1, 2, 3, 4, 2, -1, -1};
    int valeurCherche = 3;
    int i = rechercher(t, valeurCherche);
    System.out.println(valeurCherche + " est en position " + i);
    valeurCherche = 1;
    i = rechercher(t, valeurCherche);
    System.out.println(valeurCherche + " est en position " + i);
    valeurCherche = 99;
    i = rechercher(t, valeurCherche);
    System.out.println(valeurCherche + " est en position " + i);
    int element = 2;
    System.out.println("Résultat de la recherche :");
    afficheTableau3(rechercherTous(t, element));

    System.out.println("Recherche dans tableau trié");
    t = new int[] {1, 2, 3, 4, 12, -1, -1};
    valeurCherche = 3;
    i = rechercherTableauTrie(t, valeurCherche);
    System.out.println(valeurCherche + " est en position " + i);
    valeurCherche = 1;
    i = rechercherTableauTrie(t, valeurCherche);
    System.out.println(valeurCherche + " est en position " + i);
    valeurCherche = 8;
    i = rechercherTableauTrie(t, valeurCherche);
    System.out.println(valeurCherche + " est en position " + i);
    valeurCherche = 99;
    i = rechercherTableauTrie(t, valeurCherche);
    System.out.println(valeurCherche + " est en position " + i);

    if (croissant(t)) {
          // La classe java.util.Arrays peut être utile... Voir javadoc.
      System.out.println(Arrays.toString(t) + " est croissant");
    }
    else {
      System.out.println(Arrays.toString(t) + " est décroissant");
    }
    t = new int[] {-1};
```

```
    if (croissant(t)) {
      System.out.println(Arrays.toString(t) + " est croissant");
    }
    else {
      System.out.println(Arrays.toString(t) + " est décroissant");
    }
    t = new int[] {};
    if (croissant(t)) {
      System.out.println(Arrays.toString(t) + " est croissant");
    }
    else {
      System.out.println(Arrays.toString(t) + " est décroissant");
    }

    t = new int[] {1, 2, 3, 4, 12, -1, -1};
    valeurCherche = 3;
    i = rechercher2(t, valeurCherche);
    System.out.println(valeurCherche + " est en position " + i);
    t = new int[] {1, 25, 2, 3, 4, 12, -1, -1};
    valeurCherche = 3;
    i = rechercher2(t, valeurCherche);
    System.out.println(valeurCherche + " est en position " + i);

    t = new int[] {1, 2, 3, 4, 12, 3, 50, 3, -1, -1};
    afficheTableau3(t);
    System.out.println();
    int[] indices = rechercherTous(t, 3);
    afficheTableau(indices);

  }

}
```