

Exercice 01

1. Rapport checkstyle appliqué au code de base :

```
C:\Windows\System32\cmd.exe
Microsoft Windows [version 10.0.18363.720]
(c) 2019 Microsoft Corporation. Tous droits réservés.

G:\Activité 01\Exercice 01>java -jar checkstyle-8.31-all.jar -c sun_checks.xml Exemple.java
Démarrage de la vérification...
[ERROR] G:\Activité 01\Exercice 01\Exemple.java:1: Il manque un caractère NewLine à la fin du fichier. [NewlineAtEndOfFile]
[ERROR] G:\Activité 01\Exercice 01\Exemple.java:1: Le fichier package-info.java est manquant. [JavadocPackage]
[ERROR] G:\Activité 01\Exercice 01\Exemple.java:5:2: Commentaire Javadoc manquant. [JavadocVariable]
[ERROR] G:\Activité 01\Exercice 01\Exemple.java:6:2: Commentaire Javadoc manquant. [JavadocVariable]
[ERROR] G:\Activité 01\Exercice 01\Exemple.java:7:1: Commentaire Javadoc manquant. [MissingJavadocMethod]
[ERROR] G:\Activité 01\Exercice 01\Exemple.java:7:16: Le paramètre t devrait être final. [FinalParameters]
[ERROR] G:\Activité 01\Exercice 01\Exemple.java:7:23: 't' masque un attribut. [HiddenField]
[ERROR] G:\Activité 01\Exercice 01\Exemple.java:16:24: Le paramètre v devrait être final. [FinalParameters]
[ERROR] G:\Activité 01\Exercice 01\Exemple.java:16:28: 'v' masque un attribut. [HiddenField]
[ERROR] G:\Activité 01\Exercice 01\Exemple.java:16:30: '{' à la colonne 30 devrait avoir un saut de ligne après. [LeftCurly]
[ERROR] G:\Activité 01\Exercice 01\Exemple.java:16:30: Il manque une espace après '{'. [WhitespaceAround]
[ERROR] G:\Activité 01\Exercice 01\Exemple.java:16:30: Il manque une espace avant '{'. [WhitespaceAround]
[ERROR] G:\Activité 01\Exercice 01\Exemple.java:16:38: Il manque une espace après '='. [WhitespaceAround]
[ERROR] G:\Activité 01\Exercice 01\Exemple.java:16:40: Il manque une espace après ';'. [WhitespaceAfter]
[ERROR] G:\Activité 01\Exercice 01\Exemple.java:16:41: Il manque une espace avant '}'. [WhitespaceAround]
[ERROR] G:\Activité 01\Exercice 01\Exemple.java:21:1: L'instruction 'if' devrait utiliser des accolades ('{' et '}'). [NeedBraces]
[ERROR] G:\Activité 01\Exercice 01\Exemple.java:21:4: Il y a une espace de trop après '(''. [ParenPad]
Vérification terminée.
Checkstyle se termine par 17 erreurs.

G:\Activité 01\Exercice 01>
```

2. Réécrivons le code

```
/**
 * Ma class d'exemple.
 */

package activite.exercice01;

public class Exemple {
    /**
     * Variable t qui est un attribut privé
     * de type chaîne de caractères.
     */
    private final String t;

    /**
     * Variable v qui est un attribut private de type nombre entier.
     */
    private final int v;

    /**
     * Contructeur de la classe Exemple.
     * @param paramT chaîne de caractères.
     */
    public Exemple(final String paramT) {
        t = paramT;
    }

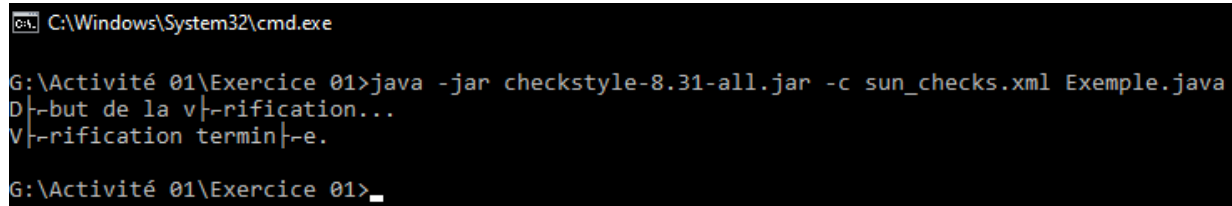
    /**
     * Accesseur de l'attribut "v".
     * @return la valeur de v
     */
    public int getV() {
        return this.v;
    }

    /**
     * Mutateur de l'attribut "v".
     * @param paramV de type int
     */
    public final void setV(final int paramV) {
        v = paramV;
    }

    /**
     * Accesseur de l'attribut "t".
     * @return t si v est positif
     */
    public final String getT() {
        if (v > 0) {
            return t;
        }
    }
}
```

}

3. Rapport checkstyle appliqué au code réécrit.



```
C:\Windows\System32\cmd.exe

G:\Activité 01\Exercice 01>java -jar checkstyle-8.31-all.jar -c sun_checks.xml Exemple.java
Début de la vérification...
Vérification terminée.

G:\Activité 01\Exercice 01>
```

Exercice 02

1. Ecrivons des tests unitaires permettant de tester les méthodes à implémenter.

```
/**
 * Dans cet exercice, nous allons implémenter des
 * tests unitaires pour la classe TabAlgos.
 * Le but étant de fournir un code respectant les
 * règles d'écriture permettant de valider que les
 * méthodes testées fourniront bien le résultat attendu.
 */

package activite.exercice02;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.fail;

import org.junit.Test;

/**
 * Dans cette classe, nous allons implémenter
 * des tests unitaires.
 * Ils nous permettront de tester les méthodes
 * écrites dans la classe TabAlgos.
 * @author LEUMASSI FANSI Jean-Léopold
 */
public class TabAlgosTest {

    /**
     * Constante ayant pour valeur 99.
     */
    private final int val1 = 99;

    /**
     * Constante ayant pour valeur 45.
     */
    private final int val2 = 45;

    /**
     * Constante ayant pour valeur 68.
     */
    private final int val3 = 68;

    /**
     * Constante ayant pour valeur 18.
     */
    private final int val4 = 18;

    /**
     * Constante ayant pour valeur 34.
     */
}
```

```
*/
private final int val5 = 34;

/**
 * Constante ayant pour valeur 50.
 */
private final int val6 = 50;

/**
 * Constante ayant pour valeur 50.
 */
private final int val7 = 37;

/**
 * Tableau N° 01.
 */
private final int[] tab1 = {val1, val2, val3, val4, val5, val6};

/**
 * Tableau N° 02.
 */
private final int[] tab2 = {val1, val2, val3, val4, val5, val6};

/**
 * Tableau N° 03.
 */
private final int[] tab3 = {val3, val2, val1, val4, val6, val5};

/**
 * Tableau N° 04.
 */
private final int[] tab4 = {val3, val4, val6, val5};

/**
 * Tableau N° 05.
 */
private final int[] tab5 = {val3, val2, val7, val4, val6, val5};

/**
 * Tableau nul l.
 */
private final int[] tabNull = null;

/**
 * Tableau vide.
 */
private final int[] tabVide = new int[0];

/**
 * Constante ayant pour valeur 0,1.
 */
private final double delta = 0.1;

/**
 * Constante ayant pour valeur 41,3.
 */
```

```
*/
private final double moyenne = 52.33;

/**
 * Test pour la méthode plusGrand. Cas nominal
 */
@Test
public void plusGrandTest() {
    assertEquals(val 1, TabAlgos.plusGrand(tab1));
}

/**
 * Test de la méthode plusGrand avec tableau vide.
 */
@Test
public void plusGrandTestAvecParamVide() {
    try {
        TabAlgos.plusGrand(tabVide);
        fail("l'exception pour les tableaux vides aurait dû être levée.");
    } catch (IllegalArgumentException e) {
        // rien à faire il s'agit d'un comportement normal
    }
}

/**
 * Test de la méthode plusGrand avec tableau vide.
 */
@Test
public void plusGrandTestAvecParamNull() {
    try {
        TabAlgos.plusGrand(tabNull);
        fail("l'exception pour les tableaux nuls aurait dû être levée.");
    } catch (IllegalArgumentException e) {
        // rien à faire il s'agit d'un comportement normal
    }
}

/**
 * Test de la méthode moyenne cas nominal.
 */
@Test
public void moyenneTest() {
    assertEquals(moyenne, TabAlgos.moyenne(tab1), delta);
}

/**
 * Test de la méthode moyenne avec tableau vide.
 */
@Test
public void moyenneTestAvecParamVide() {
    try {
        TabAlgos.moyenne(tabVide);
        fail("l'exception pour les tableaux vides "
            + "et nuls aurait dû être levée.");
    }
}
```

```
    } catch (IllegalArgumentException e) {  
        // rien à faire il s'agit d'un comportement normal  
    }  
}  
  
/**  
 * Test de la méthode moyenne avec tableau nul ou vide.  
 */  
@Test  
public void moyenneTestAvecParamNul() {  
    try {  
        TabAlgos.moyenne(tabNul);  
        fail("l'exception pour les tableaux vides "  
            + "et nuls aurait dû être levée.");  
    } catch (IllegalArgumentException e) {  
        // rien à faire il s'agit d'un comportement normal  
    }  
}  
  
/**  
 * Test pour la méthode egaux. Cas nominal  
 */  
@Test  
public void egauxTest() {  
    assertEquals(true, TabAlgos.egaux(tab1, tab2));  
}  
  
/**  
 * Test pour la méthode egaux: tableaux de même  
 * taille avec des différences de valeurs.  
 */  
@Test  
public void egauxTestDifferenceValeurs() {  
    assertEquals(false, TabAlgos.egaux(tab1, tab3));  
}  
  
/**  
 * Test pour la méthode egaux: tableaux de  
 * taille différentes.  
 */  
@Test  
public void egauxTestTaillesDifferentes() {  
    assertEquals(false, TabAlgos.egaux(tab1, tab4));  
}  
  
/**  
 * Test pour la méthode similaires. Cas nominal  
 */  
@Test  
public void similairesTest() {  
    assertEquals(true, TabAlgos.similaires(tab1, tab3));  
}  
  
/**  
 * Test pour la méthode similaires:
```

```
* tableaux de même taille avec différences de valeurs.
*/
@Test
public void similairesTestDifferenceValeurs() {
    assertEquals(false, TabAlgos.similaires(tab1, tab5));
}

/**
 * Test pour la méthode similaires: tableaux de
 * taille différentes.
 */
@Test
public void similairesTestTaillesDifferente() {
    assertEquals(false, TabAlgos.similaires(tab1, tab4));
}
}
```


2. Implémentons les méthodes en respectant les règles d'écriture contrôlées par l'outil "checkstyle"

```
/**
 * Implémenter de méthodes un tableau d'entiers.
 * Le but étant de fournir un code bien fait,
 * respectant les règles d'écriture et accompagné,
 * suivi d'un jeu de tests unitaires permettant de
 * valider qu'il fourni bien le résultat attendu.
 */

package acti vi te. exerci ce02;

/**
 * Dans cette classe, nous allons implémenter
 * des algorithmes sur un tableau d'entiers.
 * @author LEUMASSI FANSI Jean-Léopold
 */
public final class TabAl gos {
    /**
     * Constructeur protected avec exception
     * pour empêcher l'instantiation de la classe.
     * @throws Exception cette classe ne peut pas être instanciée.
     */
    protected TabAl gos() throws Exception {
        throw new Exception("cette classe ne peut pas être instanciée");
    }

    /**
     * Trouver valeur Max d'un tableau.
     * @param tab est un tableau d'entier.
     * @return valeur la plus grande d'un tableau.
     * @throw IllegalArgumentException si tab est null ou vide.
     */
    public static int plusGrand(final int[] tab) {
        //Cas d'un tableau null
        if (tab == null) {
            throw new IllegalArgumentException("le tableau ne doit pas être null.");
        }

        //Cas d'un tableau vide. La taille du tableau est égale à zero.
        if (tab.length == 0) {
            throw new IllegalArgumentException("le tableau ne doit pas être null.");
        }

        //on affecte la plus petite valeur entière possible à notre variable.
        int valeurMax = Integer.MIN_VALUE;

        //En parcourant le tableau, si une valeur supérieur à valeurMax est trouvée,
        //alors valeurMax prend cette valeur.
        for (int i = 0; i < tab.length; i++) {
            if (tab[i] > valeurMax) {
                valeurMax = tab[i];
            }
        }
    }
}
```

```
    }
}
return valeurMax;
}

/**
 * Retourne la moyenne du tableau.
 * @param tab est un tableau d'entier.
 * @return moyenne des valeurs du tableau.
 * @throw IllegalArgumentException si tab est null ou vide.
 */
public static double moyenne(final int[] tab) {
    double somme = 0.0;
    //Cas d'un tableau null
    if (tab == null) {
        throw new IllegalArgumentException("Le tableau ne doit pas être null.");
    }

    //Cas d'un tableau vide. La taille du tableau est égale à zero.
    if (tab.length == 0) {
        throw new IllegalArgumentException("Le tableau ne doit pas être null.");
    }

    for (int i = 0; i < tab.length; i++) {
        somme += tab[i]; //on procède à la somme des éléments du tableau.
    }
    return (somme / tab.length);
}

/**
 * Compare le contenu de 2 tableaux en tenant compte de l'ordre.
 * @param tab1 est un tableau d'entiers.
 * @param tab2 est un tableau d'entiers.
 * @return true si les 2 tableaux contiennent les mêmes éléments
 * avec les mêmes nombres d'occurrences
 * (avec les éléments dans le même ordre).
 */
public static boolean egaux(final int[] tab1, final int[] tab2) {
    //Si les deux tableaux sont de même taille
    if (tab1.length == tab2.length) {
        for (int i = 0; i < tab1.length; i++) {
            if (tab1[i] != tab2[i]) {
                //Si deux éléments de même index sont différents
                //alors les tableaux ne sont pas égaux.
                return false;
            }
        }
        return true;
    } else {
        //si les deux tableaux ne sont pas de même taille.
        return false;
    }
}

/**
```

```
* Compare le contenu de 2 tableaux sans tenir compte de l'ordre.
* @param tab1 est un tableau d'entiers.
* @param tab2 est un tableau d'entiers.
* @return true si les 2 tableaux contiennent les mêmes éléments
*         avec les mêmes nombres d'occurrence
*         (pas forcément dans le même ordre).
**/
public static boolean similaires(final int[] tab1, final int[] tab2) {
    //si les tableaux sont de la même taille
    if (tab1.length == tab2.length) {
        int[] temp1 = triABulles(tab1);
        int[] temp2 = triABulles(tab2);
        return egaux(temp1, temp2);
    } else {
        //si les deux tableaux ne sont pas de même taille.
        return false;
    }
}

/**
 * Compare le contenu de 2 tableaux sans tenir compte de l'ordre.
 * @param tab est un tableau d'entier.
 * @return un tableau trié par ordre croissant.
 **/
private static int[] triABulles(final int[] tab) {
    int temp;
    for (int i = tab.length - 1; i >= 1; i--) {
        for (int j = 0; j < i; j++) {
            if (tab[j] > tab[j + 1]) {
                temp = tab[j + 1];
                tab[j + 1] = tab[j];
                tab[j] = temp;
            }
        }
    }
    return tab;
}
```

3. Donnons les rapports des tests et de checkstyle.

a. Rapports checkstyle



Rapport checkstyle du fichier TabAlgosTest.java

```
C:\Windows\System32\cmd.exe
Microsoft Windows [version 10.0.18363.720]
(c) 2019 Microsoft Corporation. Tous droits réservés.

G:\Activité 01\Exercice 02>java -jar checkstyle-8.31-all.jar -c sun_checks.xml TabAlgosTest.java
Début de la vérification...
Vérification terminée.

G:\Activité 01\Exercice 02>
```

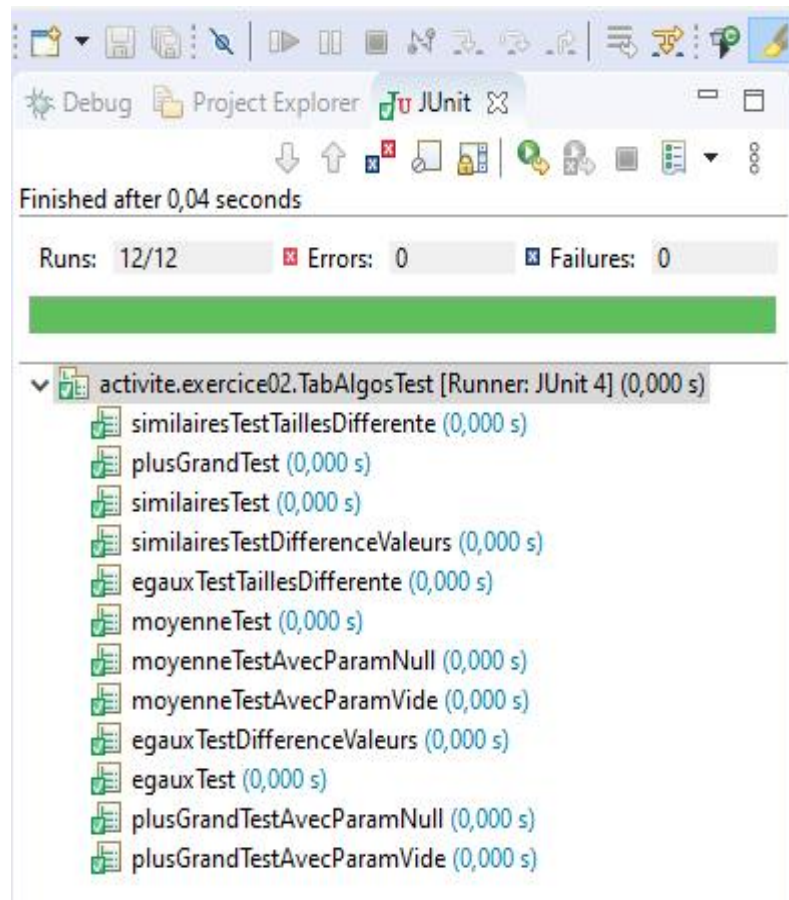


Rapport checkstyle du fichier TabAlgos.java

```
C:\Windows\System32\cmd.exe
G:\Activité 01\Exercice 02>java -jar checkstyle-8.31-all.jar -c sun_checks.xml TabAlgos.java
Début de la vérification...
Vérification terminée.

G:\Activité 01\Exercice 02>
```

b. Rapports de tests



```
C:\Windows\System32\cmd.exe

G:\Activité 01\Exercice 02>javac -d out -cp junit-platform-console-standalone-1.6.2.jar;out TabAlgos.java
G:\Activité 01\Exercice 02>javac -d out -cp junit-platform-console-standalone-1.6.2.jar;out TabAlgosTest.java
G:\Activité 01\Exercice 02>java -jar junit-platform-console-standalone-1.6.2.jar -cp out --scan-classpath
Thanks for using JUnit! Support its development at https://junit.org/sponsoring

[36m.[0m
[36m+--[0m [36mJUnit Jupiter[0m [32m[OK][0m
[36m'--[0m [36mJUnit Vintage[0m [32m[OK][0m
[36m '-[0m [36mTabAlgosTest[0m [32m[OK][0m
[36m   +--[0m [34msimilairesTestTaillesDifferente[0m [32m[OK][0m
[36m   +--[0m [34mplusGrandTest[0m [32m[OK][0m
[36m   +--[0m [34msimilairesTest[0m [32m[OK][0m
[36m   +--[0m [34msimilairesTestDifferenceValeurs[0m [32m[OK][0m
[36m   +--[0m [34megauxTestTaillesDifferente[0m [32m[OK][0m
[36m   +--[0m [34moyenneTest[0m [32m[OK][0m
[36m   +--[0m [34moyenneTestAvecParamNull[0m [32m[OK][0m
[36m   +--[0m [34moyenneTestAvecParamVide[0m [32m[OK][0m
[36m   +--[0m [34megauxTestDifferenceValeurs[0m [32m[OK][0m
[36m   +--[0m [34megauxTest[0m [32m[OK][0m
[36m   +--[0m [34mplusGrandTestAvecParamNull[0m [32m[OK][0m
[36m   '-[0m [34mplusGrandTestAvecParamVide[0m [32m[OK][0m

Test run finished after 111 ms
[      3 containers found      ]
[      0 containers skipped    ]
[      3 containers started    ]
[      0 containers aborted    ]
[      3 containers successful ]
[      0 containers failed     ]
[     12 tests found           ]
[      0 tests skipped         ]
[     12 tests started         ]
[      0 tests aborted         ]
[     12 tests successful      ]
[      0 tests failed          ]

G:\Activité 01\Exercice 02>_
```

Exercice 03

Lien du dépôt Git :

https://github.com/leolelover93/M1-MIAGE-C306-INGENIERIE_DU_LOGICIEL