

Exercice 1 :

1) rapport de "checkstyle"

Détails

Exemple.java

Sévérité	Catégorie	Règle	Message	Ligne
✖ Erreur	misc	NewlineAtEndOfFile	Il manque un caractère NewLine à la fin du fichier	
✖ Erreur	javadoc	JavadocPackage	Manquant fichier package-info.java.	
✖ Erreur	javadoc	JavadocVariable	Commentaire javadoc manquant.	5
✖ Erreur	javadoc	JavadocVariable	Commentaire javadoc manquant.	6
✖ Erreur	javadoc	JavadocMethod	Commentaire javadoc manquant.	7
✖ Erreur	misc	FinalParameters	Le paramètre t devrait être final.	7
✖ Erreur	coding	HiddenField	't' masque un attribut.	7
✖ Erreur	design	DesignForExtension	La méthode 'getV' n'est pas conçue pour être dérivée - il faut la déclarer abstraite, finale ou la laisser vide.	13
✖ Erreur	misc	FinalParameters	Le paramètre v devrait être final.	16
✖ Erreur	coding	HiddenField	'v' masque un attribut.	16
✖ Erreur	blocks	LeftCurly	'{' à la colonne 30 devrait avoir saut de ligne après.	16
✖ Erreur	whitespace	WhitespaceAround	Il manque une espace avant '{'.	16
✖ Erreur	whitespace	WhitespaceAround	Il manque une espace après '{'.	16
✖ Erreur	whitespace	WhitespaceAround	Il manque une espace après '='.	16
✖ Erreur	whitespace	WhitespaceAfter	Il manque une espace après ';'.	16
✖ Erreur	whitespace	WhitespaceAround	Il manque une espace avant '}'.	16
✖ Erreur	blocks	NeedBraces	L'instruction 'if' devrait utiliser des accolades ('{' et '}').	21
✖ Erreur	whitespace	ParenPad	Il y a une espace de trop après '('.	21

2) réécriture du code

```
/**
 * Ma class d'exemple.
 */
public class Exemple2 {
    /**
     * @param t
     */
    private final String t;
    /**
     * @param v
     */
    private int v;
    /**
     * @param arg initialise la valeur de v au constructeur
     */
    public Exemple2(final String arg) {
        this.t = arg;
    }
    /**
     * @return la valeur de v
     */
    public final int getV() {
        return this.v;
    }
    /**
     * @param arg edite la valeur v de la classe
     */
    public final void setV(final int arg) {
        this.v = arg; }
    /**
     * @return t si v est positif
     */
    public final String getT() {
        if (v > 0) {
            return t;
        }
        return null;
    }
}
```

3-rapport de "checkstyle" après la réécriture du code

Résultats Checkstyle

Le document suivant contient les résultats de [Checkstyle](#) 6.18 avec les règles [sun_checks.xml](#). [XML](#)

Résumé

Fichiers	 Infos	 Avertissements	 Erreurs
2	0	0	0

Fichiers

Fichier	 I	 A	 E
---------	---	---	---

Règles

Catégorie	Règle	Violations	Sévérité
-----------	-------	------------	----------

Détails

Remarque : plus aucune anomalie n'est détecté dans le code

Exercice 2

1- tests unitaires.

```
class TabAlgosTest {  
    /**  
     * Teste la fonction Plusgrand  
     */  
    @Test  
    void testPlusGrand() {  
        assertEquals(44, TabAlgos.plusGrand(new int[]{ 1,20,3,7,13,44,5,9,10,2 }));  
    }  
  
    /**  
     * Teste Si le tableau est null ou vide dans la fonction Plusgrand  
     */  
    @Test  
    void testPlusGrandNullVide() {  
        try {  
            TabAlgos.plusGrand(null);  
            TabAlgos.plusGrand(new int[] {});  
            fail("l'exception pour le tableau a valeur null ou vide aurait dû être levée");  
        }  
        catch (IllegalArgumentException e) {  
            // Si tout fonctionne bien  
        }  
    }  
  
    /**  
     * Teste la fonction moyenne  
     */  
    @Test  
    void testMoyenne() {  
        assertEquals(18, TabAlgos.moyenne(new int[]{4,70,13,5,11,6}));  
    }  
  
    /**  
     * Teste Si le tableau est null ou vide dans la fonction moyenne  
     */  
}
```

```
@Test
void testMoyenneNullVide() {
    try {
        TabAlgos.moyenne(null);
        TabAlgos.moyenne(new int[] {});
        fail("l'exception pour le tableau a valeur null ou vide aurait dû être levée");
    }
    catch(IllegalArgumentException e) {
        // Si tout fonctionne bien
    }
}

/**
 * Teste la fonction egaux
 */

@Test
void testEgaux() {

    assertFalse(TabAlgos.egaux(new int[]{4,70,13,5,6,11},new int[]{4,70,13,5,11,6}));

}

/**
 * Teste Si le tableau 1 est null dans la fonction Egaux
 */

@Test
void testEgauxNullTab1() {

    try {
        TabAlgos.egaux(null,new int[]{});
        fail("l'exception pour le tableau 1 a valeur null aurait dû être levée");
    }
    catch(IllegalArgumentException e) {
        // Si tout fonctionne bien
    }
}

/**
 * Teste Si le tableau 2 est null dans la fonction Egaux
 */
```

```
@Test
void testEgauxNullTab2() {

    try {
        TabAlgos.egaux(new int[] {}, null);
        fail("l'exception pour le tableau 2 a valeur null aurait dû être levée");
    }
    catch (IllegalArgumentException e) {
        // Si tout fonctionne bien
    }
}

/**
 * Teste la fonction Similaire
 */

@Test
void testSimilaires() {
    assertFalse(TabAlgos.similaires(new int[] {6,11,5,13,70,14}, new int[] {4,70,13,5,11,6}));
}

/**
 * Teste la fonction trierTableau
 */

@Test
void testTrierTableau() {

    try {
        TabAlgos.trierTableau(null);
        fail("l'exception pour le tableau a valeur null aurait dû être levée");
    }
    catch (IllegalArgumentException e) {
        // Si tout fonctionne bien
    }
}

/**
```

```
    * Teste Si le tableau a trier null dans la fonction trierTableau
    */
    @Test
    void testTrierTableauNull() {

        try {
            TabAlgos.trierTableau(null);
            fail("l'exception pour le tableau a valeur null aurait dû être levée");
        }
        catch (IllegalArgumentException e) {
            // Si tout fonctionne bien
        }
    }
}
```

2-Implementation des méthodes

```
/**
 * Classe TabAlgos.
 */
public final class TabAlgos {
    /**
     * Constructeur privé non accessible.
     */
    private TabAlgos() {
    }
    /**
     * @return valeur la plus grande d'un tableau.
     * @param tab le tableau a traiter.
     */
    public static int plusGrand(final int[] tab) {
        if (tab == null || tab.length == 0) {
            throw new IllegalArgumentException("Tableau vide ou null");
        }
        int val = tab[0];
        for (int i = 0; i < tab.length; i++) {
            if (tab[i] >= val) {
                val = tab[i];
            }
        }
    }
}
```

```
}
    return val;
}
/**
 * @return moyenne des valeurs du tableau.
 * @param tab le tableau a traiter.
 * @throw IllegalArgumentException si tab et null ou vide.
 */
public static double moyenne(final int[] tab) {
    if (tab == null || tab.length == 0) {
        throw new IllegalArgumentException("Tableau vide ou null");
    }
    int val = 0;
    for (int i = 0; i < tab.length; i++) {
        val += tab[i];
    }
    return val / tab.length;
}
/**
 * Compare le contenu de 2 tableaux en tenant compte de l'ordre.
 * @return true si les 2 tableaux contiennent les mêmes éléments.
 * avec les mêmes nombres d'occurrences.
 * (avec les elements dans le meme ordre).
 * @param tab1 le tableau 1 a comparer.
 * @param tab2 le tableau 2 a comparer.
 */
public static boolean egaux(final int[] tab1, final int[] tab2) {
    if (tab1 == null) {
        throw new IllegalArgumentException("Le tableau 1 est null");
    }
    if (tab2 == null) {
        throw new IllegalArgumentException("Le tableau 2 est null");
    }
    if (tab1.length != tab2.length) {
        return false;
    }
    for (int i = 0; i < tab1.length; i++) {
        if (tab1[i] != tab2[i]) {
            return false;
        }
    }
}
```



```
}
return true;
}
/**
 *Compare le contenu de 2 tableaux sans tenir compte de l'ordre.
 *@return true si les 2 tableaux contiennent les mêmes éléments
 * avec les mêmes nombres d'occurrence.
 *(pas forcément dans le meme ordre).
 * @param tab1 le tableau 1 a comparer.
 * @param tab2 le tableau 2 a comparer.
 */
public static boolean similaires(final int[] tab1, final int[] tab2) {
return egaux(trierTableau(tab1), trierTableau(tab2));
}
/**
 *@return trie le tableau passer en parametre
 *@param tab le tableau a trier.
 */
protected static int[] trierTableau(final int[] tab) {
if (tab == null) {
throw new IllegalArgumentException("Le tableau est null");
}
boolean permuter = true;
int j = 0;
int tamp;
while (permuter) {
permuter = false;
j++;
for (int i = 0; i < tab.length - j; i++) {
if (tab[i] > tab[i + 1]) {
tamp = tab[i];
tab[i] = tab[i + 1];
tab[i + 1] = tamp;
permuter = true;
}
}
}
return tab;
}
}
```

3- validité des tests

a) Rapport checkstyle

Résultats Checkstyle

Le document suivant contient les résultats de Checkstyle  6.18 avec les règles sun_checks.xml. 

Résumé

Fichiers	 Infos	 Avertissements	 Erreurs
1	0	0	0

Fichiers

Fichier	 I	 A	 E
---------	---	---	---

Règles

Catégorie	Règle	Violations	Sévérité
-----------	-------	------------	----------

Détails

b) Rapport des test unitaires(Test Cases):

Test Cases[\[Summary\]](#) [\[Package List\]](#) [\[Test Cases\]](#)**TabAlgosTest**

	testTrierTableauNull	0,001
	testEgaux	0
	testEgauxNullTab1	0
	testEgauxNullTab2	0
	testPlusGrand	0
	testPlusGrandNullVide	0
	testMoyenneNullVide	0,006
	testTrierTableau	0
	testSimilaires	0
	testMoyenne	0,01

Exercice 3 : à ce jours je n'est pas reçus de réponse concernant le choix du serveur du versioning (alors le travail est temporairement téléchargé sur GitHub à l'adresse : <https://github.com/irtrade/devoir1.git>)