Exercice 1 : Ecriture de code

1. Ce code ne compile pas parce que :

* Le code essai de modifier des valeurs de constantes (le prix et la refernce)
* Des attributs reference et prix ne sont pas initialisés;
* Les valeurs de ces variables non initialisé sont modifiées au cours de l’exécution ;
* La méthode getReference() ne retourne pas de valeur si le prix est inférieur à 0

1. Code corrigé :

**package** com.activite.un;

/\*\* Classe Produit représentant un produit avec un prix et une référence \*/

**public** **class** Produit {

**private** String reference;

**private** **double** prix;

**double** tva = 0.20;

**public** Produit(String reference) {**this**.reference = reference;}

/\*\* **@return** le prix \*/

**public** **double** getPrix() {

**return** **this**.prix;

}

/\*\* modifie le prix.

\* **@param** prix

\*\*/

**public** **final** **void** setPrix(Double prix){**this**.prix = prix;}

/\*\*

\* **@return** la reference si le prix est positif, null sinon

\*/

**public** **final** String getReference() {

**return** reference;

}

@Override

**public** **boolean** equals(Object o) {

**return** reference == ((Produit)o).reference;

}

}

1. **Rapport de checkstyle, spotbugs et PMD appliqués à votre code.**

* **CheckStyle**

C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un>java -jar checkstyle-8.45.1-all.jar -c sun\_checks.xml Produit.

java

D├®but de la v├®rification...

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:1: Il manque un caract├¿re NewLine ├á la

fin du fichier. [NewlineAtEndOfFile]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:3: La premi├¿re ligne de la Javadoc doit

se terminer avec un point. [JavadocStyle]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:5:2: Commentaire Javadoc manquant. [Javad

ocVariable]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:6:2: Commentaire Javadoc manquant. [Javad

ocVariable]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:7:2: Commentaire Javadoc manquant. [Javad

ocVariable]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:7:9: La variable 'tva' devrait ├¬tre priv

├®e et avoir des accesseurs. [VisibilityModifier]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:7:15: '0.20' devrait ├¬tre d├®fini comme

une constante. [MagicNumber]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:9:16: Le param├¿tre reference devrait ├¬t

re final. [FinalParameters]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:9:23: 'reference' masque un attribut. [Hi

ddenField]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:9:34: '{' ├á la colonne 34 devrait avoir

un saut de ligne apr├¿s. [LeftCurly]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:9:34: Il manque une espace apr├¿s '{'. [W

hitespaceAround]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:9:61: Il manque une espace apr├¿s ';'. [W

hitespaceAfter]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:9:62: Il manque une espace avant '}'. [Wh

itespaceAround]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:19:27: Le param├¿tre prix devrait ├¬tre f

inal. [FinalParameters]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:19:34: 'prix' masque un attribut. [Hidden

Field]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:19:39: '{' ├á la colonne 39 devrait avoir

un saut de ligne apr├¿s. [LeftCurly]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:19:39: Il manque une espace apr├¿s '{'. [

WhitespaceAround]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:19:39: Il manque une espace avant '{'. [W

hitespaceAround]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:19:56: Il manque une espace apr├¿s ';'. [

WhitespaceAfter]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:19:57: Il manque une espace avant '}'. [W

hitespaceAround]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:28:1: La classe 'Produit' semble ├¬tre co

n├ºue pour extension (peut ├¬tre h├®rit├®e), mais la m├®thode'equals' n'a pas de Javadoc qui explique comment le faire en toute s├®curit├®. S

i la classe n'est pas con├ºue pour extension, envisagez de rendre la classe 'Produit' finale ou de rendre la m├®thode 'equals' static/final/a

bstract/empty, d'ajouter les annotations permises pour la m├®thode. [DesignForExtension]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:28:1: La d├®finition de la m├®thode 'equa

ls()' doit toujours ├¬tre accompagn├®e de la d├®finition de la m├®thode 'hashCode()'. [EqualsHashCode]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:29:23: Le param├¿tre o devrait ├¬tre fina

l. [FinalParameters]

[ERROR] C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un\Produit.java:30:30: Il manque une espace apr├¿s 'conve

rsion de type'. [WhitespaceAfter]

V├®rification termin├®e.

Checkstyle se termine par 24 erreurs.

* **Spotbugs**
* **PMD**

src/main/java/com/activite/un/Produit.java:5: BeanMembersShouldSerialize: BeanMembersShouldSerialize: Found non-transient, non-static member. Please mark as transient or provide accessors.

src/main/java/com/activite/un/Produit.java:5: BeanMembersShouldSerialize: BeanMembersShouldSerialize: Found non-transient, non-static member. Please mark as transient or provide accessors.

src/main/java/com/activite/un/Produit.java:5: CommentRequired: CommentRequired: Field comments are required

src/main/java/com/activite/un/Produit.java:5: CommentRequired: CommentRequired: Field comments are required

src/main/java/com/activite/un/Produit.java:5: ImmutableField: ImmutableField: Private field 'reference' could be made final; it is only initialized in the declaration or constructor.

src/main/java/com/activite/un/Produit.java:5: ImmutableField: ImmutableField: Private field 'reference' could be made final; it is only initialized in the declaration or constructor.

src/main/java/com/activite/un/Produit.java:6: CommentRequired: CommentRequired: Field comments are required

src/main/java/com/activite/un/Produit.java:6: CommentRequired: CommentRequired: Field comments are required

src/main/java/com/activite/un/Produit.java:7: CommentDefaultAccessModifier: CommentDefaultAccessModifier: To avoid mistakes add a comment at the beginning of the tva field if you want a default access modifier

src/main/java/com/activite/un/Produit.java:7: CommentDefaultAccessModifier: CommentDefaultAccessModifier: To avoid mistakes add a comment at the beginning of the tva field if you want a default access modifier

src/main/java/com/activite/un/Produit.java:7: CommentRequired: CommentRequired: Field comments are required

src/main/java/com/activite/un/Produit.java:7: CommentRequired: CommentRequired: Field comments are required

src/main/java/com/activite/un/Produit.java:7: DefaultPackage: DefaultPackage: Use explicit scoping instead of the default package private level

src/main/java/com/activite/un/Produit.java:7: DefaultPackage: DefaultPackage: Use explicit scoping instead of the default package private level

src/main/java/com/activite/un/Produit.java:9: CommentRequired: CommentRequired: Public method and constructor comments are required

src/main/java/com/activite/un/Produit.java:9: CommentRequired: CommentRequired: Public method and constructor comments are required

src/main/java/com/activite/un/Produit.java:9: MethodArgumentCouldBeFinal: MethodArgumentCouldBeFinal: Parameter 'reference' is not assigned and could be declared final

src/main/java/com/activite/un/Produit.java:9: MethodArgumentCouldBeFinal: MethodArgumentCouldBeFinal: Parameter 'reference' is not assigned and could be declared final

src/main/java/com/activite/un/Produit.java:19: MethodArgumentCouldBeFinal: MethodArgumentCouldBeFinal: Parameter 'prix' is not assigned and could be declared final

src/main/java/com/activite/un/Produit.java:19: MethodArgumentCouldBeFinal: MethodArgumentCouldBeFinal: Parameter 'prix' is not assigned and could be declared final

src/main/java/com/activite/un/Produit.java:29: MethodArgumentCouldBeFinal: MethodArgumentCouldBeFinal: Parameter 'o' is not assigned and could be declared final

src/main/java/com/activite/un/Produit.java:29: MethodArgumentCouldBeFinal: MethodArgumentCouldBeFinal: Parameter 'o' is not assigned and could be declared final

src/main/java/com/activite/un/Produit.java:29: OverrideBothEqualsAndHashcode: OverrideBothEqualsAndHashcode: Ensure you override both equals() and hashCode()

src/main/java/com/activite/un/Produit.java:29: OverrideBothEqualsAndHashcode: OverrideBothEqualsAndHashcode: Ensure you override both equals() and hashCode()

src/main/java/com/activite/un/Produit.java:29: ShortVariable: ShortVariable: Avoid variables with short names like o

src/main/java/com/activite/un/Produit.java:29: ShortVariable: ShortVariable: Avoid variables with short names like o

src/main/java/com/activite/un/Produit.java:30: CompareObjectsWithEquals: CompareObjectsWithEquals: Use equals() to compare object references.

src/main/java/com/activite/un/Produit.java:30: CompareObjectsWithEquals: CompareObjectsWithEquals: Use equals() to compare object references.

src/main/java/com/activite/un/Produit.java:30: UseEqualsToCompareStrings: UseEqualsToCompareStrings: Use equals() to compare strings instead of '==' or '!='

src/main/java/com/activite/un/Produit.java:30: UseEqualsToCompareStrings: UseEqualsToCompareStrings: Use equals() to compare strings instead of '==' or '!='

1. **Proposez une réécriture en tenant compte des problèmes soulevés par checkstyle, spotbugs et PMD.**

**package** com.activite.un;

/\*\* Classe Produit représentant un.

\* produit avec un prix et une référence. \*/

**public** **class** Produit {

/\*\* String reference. \*\*/

**private** String reference;

/\*\* double prix. \*\*/

**private** **double** prix;

//double tva = 0.20;

/\*\*

\* **@author** MARCO

\* **@param** lReference

\*/

**public** Produit(**final** String lReference) {

**this**.reference = lReference;

}

/\*\* **@param** lReference

\* \*/

**public** **final** **void** setReference(**final** String lReference) {

**this**.reference = lReference;

}

/\*\* **@return** le prix. \*/

**public** **double** getPrix() {

**return** **this**.prix;

}

/\*\* modifie le prix.

\* **@param** lPrix

\*\*/

**public** **final** **void** setPrix(**final** Double lPrix) {

**this**.prix = lPrix;

}

/\*\*

\* **@return** la reference si le prix est positif, null sinon.

\*/

**public** **final** String getReference() {

**return** reference;

}

/\*\*

\* **@return** reference si le produit.

\* correspond à l'objet passé en paramètre

\*/

@Override

**public** **boolean** equals(**final** Object object) {

**boolean** bool;

**if** (object == **null**) {

bool = **false**;

} **else** {

bool = **this**.reference.equals(((Produit) object).reference);

}

**return** bool;

}

/\*\*

\* **@return** reference hashée

\*/

@Override

**public** **int** hashCode() {

**return** **this**.getReference().hashCode();

}

}

1. **Donnez les rapports appliqués au code réécrit.**

* Checkstyle

C:\Users\MARCO\eclipse-workspace\com.activite.un\src\main\java\com\activite\un>java -jar checkstyle-8.45.1-all.jar -c sun\_checks.xml Produit.java

D├®but de la v├®rification...

V├®rification termin├®e.

* PMD

src/main/java/com/activite/un/Produit.java:5: DataClass: DataClass: The class 'Produit' is suspected to be a Data Class (WOC=28.571%, NOPA=0, NOAM=4, WMC=8)

src/main/java/com/activite/un/Produit.java:5: DataClass: DataClass: The class 'Produit' is suspected to be a Data Class (WOC=28.571%, NOPA=0, NOAM=4, WMC=8)

* SpotBugs

**Exercice 2** : Tests unitaires

1. Ecriture des tests unitaires (en utilisant JUnit 5) permettant de tester les méthodes à implémenter.

**public** **final** **class** TabAlgos {

/\*\* **@return** valeur la plus grande d'un tableau. \*/

**public** **static** **int** plusGrand(**final** **int**[] tab) {

//Si le tableau null

**if** (tab == **null**) {

**throw** **new** IllegalArgumentException("le tableau ne doit pas être null.");

}

//Cas d'un tableau vide.

**if** (tab.length == 0) {

**throw** **new** IllegalArgumentException("le tableau ne comporte pas de données.");

}

//Affectation de la plus petite valeur entière possible à notre variable.

**int** vMax = Integer.***MIN\_VALUE***;

//Parcourir le tableau à la recherche de la valeur max

**for** (**int** i = 0; i < tab.length; i++) {

**if** (tab[i] > vMax) {

vMax = tab[i];

}

}

**return** vMax;

}

/\*\*

\* **@return** moyenne des valeurs du tableau.

\* **@throw** IllegalArgumentException si tab et 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.");

}

//Si le tableau est vide.

**if** (tab.length == 0) {

**throw** **new** IllegalArgumentException("le tableau ne comporte pas de données.");

}

**for** (**int** i = 0; i < tab.length; i++) {

somme += tab[i]; //On fait la somme des éléments du tableau.

}

**return** (somme / 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'occurences

\* (avec les elements dans le meme ordre).

\*\*/

**public** **static** **boolean** egaux(**final** **int**[] tab1,**final** **int**[] tab2) {

//Vérifier 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 tabeaux ne sont pas de même taille.

**return** **false**;

}

}

/\*\*

\* Trie le tableau passé en paramètre.

\* **@param** tab est un tableau d'entier.

\* **@return** un tableau trié par ordre croissant.

\*\*/

**private** **static** **int**[] tri(**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;

}

/\*\*

\* 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

\*\*/

**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 = *tri*(tab1);

**int**[] temp2 = *tri*(tab2);

**return** *egaux*(temp1, temp2);

} **else** {

//si les deux tabeaux ne sont pas de même taille.

**return** **false**;

}

}

}

* Classe de test

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

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

**import** org.junit.jupiter.api.Test;

**public** **class** TabAlgosTest {

/\*\*

\* Constante ayant pour valeur 70.

\*/

**private** **final** **int** p1 = 70;

/\*\*

\* Constante ayant pour valeur 30.

\*/

**private** **final** **int** p2 = 30;

/\*\*

\* Constante ayant pour valeur 48.

\*/

**private** **final** **int** p3 = 48;

/\*\*

\* Constante ayant pour valeur 22.

\*/

**private** **final** **int** p4 = 22;

/\*\*

\* Constante ayant pour valeur 43.

\*/

**private** **final** **int** p5 = 43;

/\*\*

\* Constante ayant pour valeur 10.

\*/

**private** **final** **int** p6 = 10;

/\*\*

\* Constante ayant pour valeur 9.

\*/

**private** **final** **int** p7 = 9;

/\*\*

\* Tableau N° 01.

\*/

**private** **final** **int**[] tab1 = {p1, p2, p3, p4, p5, p6};

/\*\*

\* Tableau N° 02.

\*/

**private** **final** **int**[] tab2 = {p1, p2, p3, p4, p5, p6};

/\*\*

\* Tableau N° 03.

\*/

**private** **final** **int**[] tab3 = {p3, p2, p1, p4, p6, p5};

/\*\*

\* Tableau N° 04.

\*/

**private** **final** **int**[] tab4 = {p3, p4, p6, p5};

/\*\*

\* Tableau N° 05.

\*/

**private** **final** **int**[] tab5 = {p3, p2, p7, p4, p6, p5};

/\*\*

\* Tableau null.

\*/

**private** **final** **int**[] tableauDeNull = **null**;

/\*\*

\* Tableau vide.

\*/

**private** **final** **int**[] tableauDeVide = **new** **int**[0];

/\*\*

\* Constante ayant pour valeur 33,14.

\*/

**private** **final** **double** constanteC = 33.14;

/\*\*

\* Constante ayant pour valeur 33,14.

\*/

**private** **final** **double** moyenne = 33.14;

/\*\*

\* Test pour la méthode plusGrand. Cas nominal

\*/

@Test

**public** **void** plusGrandTest() {

*assertEquals*(p1, TabAlgos.*plusGrand*(tab1));

}

/\*\*

\* Test de la méthode plusGrand avec tableau vide.

\*/

@Test

**public** **void** plusGrandTestAvecParamVide() {

**try** {

TabAlgos.*plusGrand*(tableauDeVide);

*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*(tableauDeNull);

*fail*("l'exception pour les tableaux nulls 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), constanteC);

}

/\*\*

\* Test de la méthode moyenne avec tableau vide.

\*/

@Test

**public** **void** moyenneTestAvecParamVide() {

**try** {

TabAlgos.*moyenne*(tableauDeVide);

*fail*("l'exception pour les tableaux vides "

+ "et nulls aurait dû être levée.");

} **catch** (IllegalArgumentException e) {

// rien à faire il s'agit d'un comportement normal

}

}

/\*\*

\* Test de la méthode moyenne avec tableau null ou vide.

\*/

@Test

**public** **void** moyenneTestAvecParamNull() {

**try** {

TabAlgos.*moyenne*(tableauDeNull);

*fail*("l'exception pour les tableaux vides "

+ "et nulls 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** egauxTestTaillesDifferente() {

*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));

}

}

1. Implémentons les méthodes en respectant les règles d'écriture contrôlées par les outils checkstyle, spotbugs et PMD.

**package** com.exercice2;

/\*\*

\* **@author** MARCO

\* Classe TabAlgos

\*/

**public** **final** **class** TabAlgosClass {

/\*\*

\* **@author** MARCO

\* Boolean bool

\*/

**private** **static** **boolean** *bool*;

/\*\*

\* Constructeur protected qui lève une exception

\* pour empecher l'instantiation de la classe.

\* **@throws** Exception cette classe ne peut pas être instanciée.

\*/

**private** TabAlgosClass() { }

/\*\* **@return** valeur la plus grande d'un tableau.

\* **@param** tab

\* **@param** args

\*/

**public** **static** **int** plusGrand(**final** **int**[] tab, **final** Object... args) {

//Si le tableau null

**if** (tab == **null**) {

**throw** **new** IllegalArgumentException("le tableau ne doit pas être null.");

}

//Cas d'un tableau vide.

**if** (tab.length == 0) {

**throw** **new** IllegalArgumentException("le tableau ne comporte pas de données.");

}

//Affectation de la plus petite valeur entière possible à notre variable.

**int** vMax = Integer.***MIN\_VALUE***;

//Parcourir le tableau à la recherche de la valeur max

**for** (**final** **int** j : tab) {

**if** (j > vMax) {

vMax = j;

}

}

**return** vMax;

}

/\*\*

\* **@return** moyenne des valeurs du tableau.

\* **@throw** IllegalArgumentException si tab et null ou vide.

\* **@param** tab

\* **@param** args

\*\*/

**public** **static** **double** moyenne(**final** **int**[] tab, **final** Object... args) {

//Cas d'un tableau null

**if** (tab == **null**) {

**throw** **new** IllegalArgumentException("le tableau ne doit pas être null.");

}

//Si le tableau est vide.

**if** (tab.length == 0) {

**throw** **new** IllegalArgumentException("le tableau ne comporte pas de données.");

}

**double** somme = 0.0;

**for** (**final** **int** i : tab) {

somme += i; //On fait la somme des éléments du tableau.

}

**return** somme / tab.length;

}

/\*\*

\* Compare le contenu de 2 tableaux en tenant compte de l'ordre.

\* **@return** true si les 2 tableaux sont egaux

\* **@param** tab1

\* **@param** tab2

\* **@param** args

\*\*/

**public** **static** **boolean** egaux(**final** **int**[] tab1, **final** **int**[] tab2,

**final** Object... args) {

*bool* = **false**;

//Vérifier si les deux tableaux sont de même taille

**if** (tab1.length == tab2.length) {

**int** cpt = 0;

**if** (tab1[cpt] == tab2[cpt]) {

**do** {

cpt++;

} **while** (tab1[cpt] == tab2[cpt] && cpt < tab1.length - 1);

}

**if** (cpt == tab1.length - 1) {

*bool* = **true**;

}

}

**return** *bool*;

}

/\*\*

\* Trie le tableau passé en paramètre.

\* **@param** tab est un tableau d'entier.

\* **@return** un tableau trié par ordre croissant.

\* **@param** args

\*\*/

**public** **static** **int**[] tri(**final** **int**[] tab, **final** Object... args) {

**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;

}

/\*\*

\* Compare le contenu de 2 tableaux sans tenir compte de l'ordre.

\* **@return** true si les 2 tableaux sont egaux

\* **@param** tab1

\* **@param** tab2

\* **@param** args

\*\*/

**public** **static** **boolean** similaires(**final** **int**[] tab1, **final** **int**[] tab2,

**final** Object... args) {

*bool* = **false**;

//si les tableaux sont de la même taille

**if** (tab1.length == tab2.length) {

**final** **int**[] temp1 = *tri*(tab1);

**final** **int**[] temp2 = *tri*(tab2);

**if** (*egaux*(temp1, temp2)) {

*bool* = **true**;

}

}

//si les deux tabeaux ne sont pas de même taille.

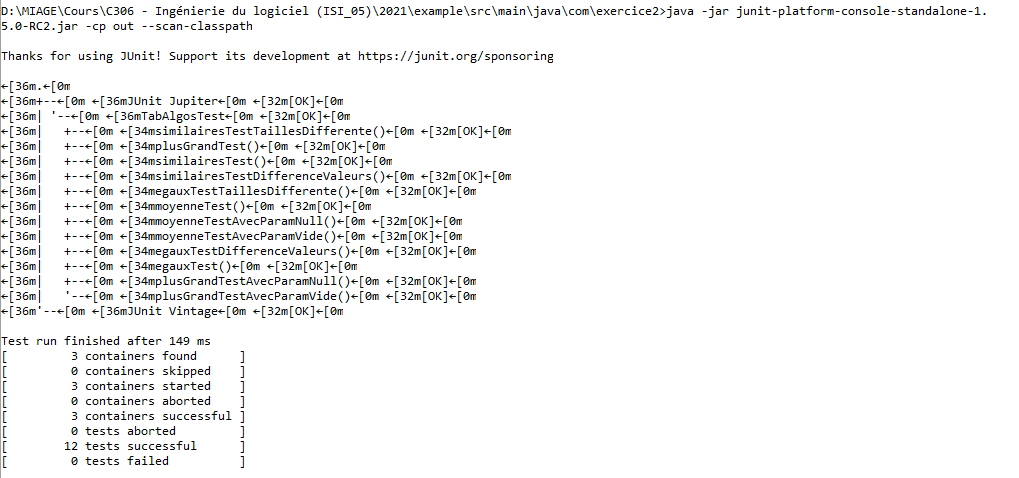
**return** *bool*;

}

}

1. Vérifiez la validité des tests avec le code implémenté.
2. Fournir les rapports des tests unitaires, de checkstyle, spotbugs et PMD. Il ne doit rester aucune anomalie.

* Rapport de test



* Checkstyle

D:\MIAGE\Cours\C306 - Ingénierie du logiciel (ISI\_05)\2021\example\src\main\java\com\exercice2>java -jar checkstyle-8.45.1-all.jar -c sun\_checks.xml TabAl

gosClass.java

D├®but de la v├®rification...

V├®rification termin├®e.

**Exercice 3 : Dépôt sur serveur de Versionning**

Lien vers le repository : <https://github.com/marckodjo/Activite1>

Clé rsa :

ssh-rsa  [marc.amedonou@gmail.com]