Ewen CLÉMENT – MASTER MIAGE

Activté 1 C306 - Ingénierie du logiciel

Exercice 1 : Écriture de code

1. Pourquoi ce code ne compile-t-il pas ?

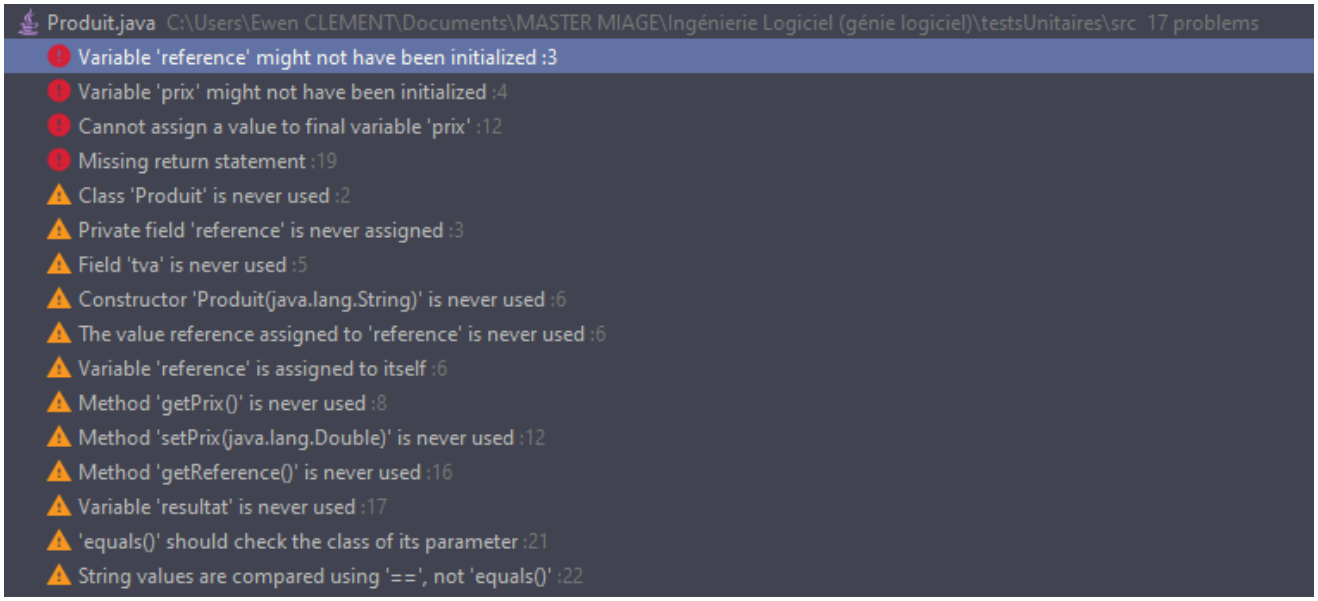
Le code ne compile pas pour plusieurs raisons (je ne cite ici que les erreurs, pas les avertissements) :

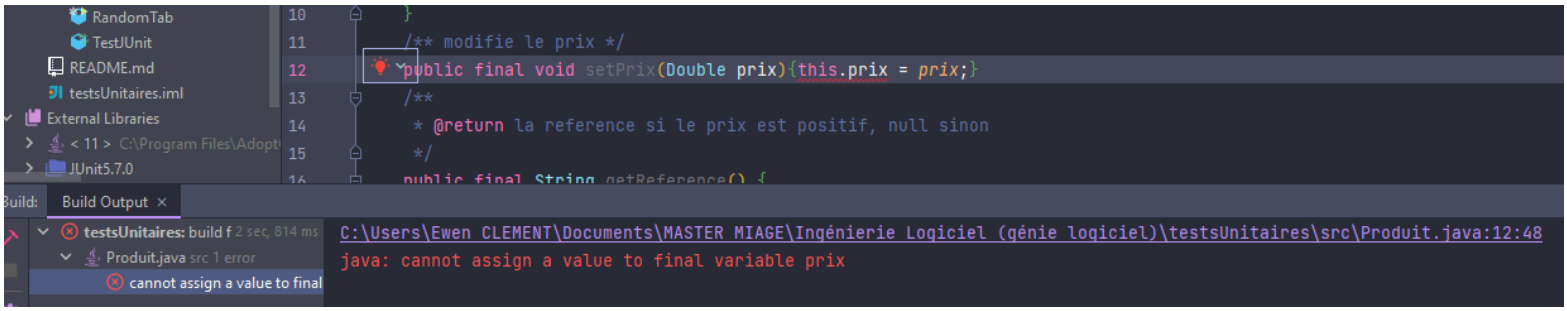
- l'attribut prix est désigné comme constante mais n'est pas initialisé, on ne lui affecte pas de valeur, et on tente de le modifier dans la fonction setPrix.

- l'attribut reference est lui aussi désigné comme une constante mais n'est pas initialisé.

- dans la fonction getReference, si le prix est de 0 ou moins, aucun return ne sera fait.

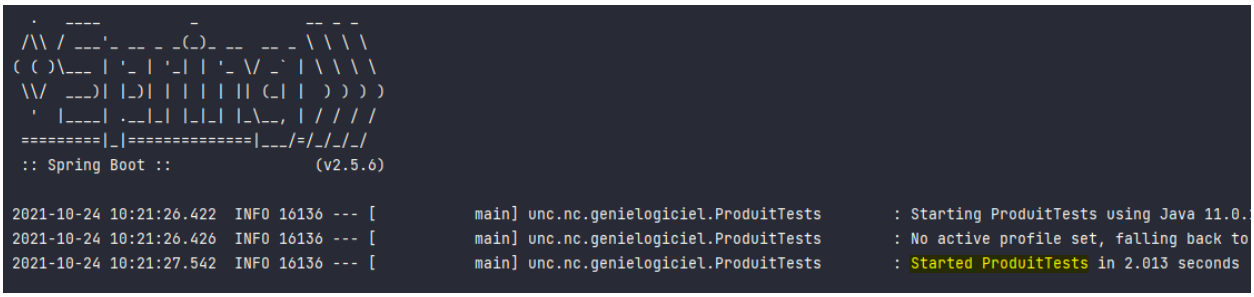
Voici ce que mon IDE retourne (Intellij Idea) :





2. Corrigez le code pour qu'il compile (en respectant les consignes de la documentation).

J'ai modifié le code de façon à ne plus avoir d'erreurs qui empêchaient la compilation. J'ai mis cette classe produit dans un projet Spring Boot afin d'avoir toutes les dépendances de tests sans que j'aie besoin de les importer moi-même et j'ai créé une classe de test pour tester la classe produit qui se lance correctement.



3. Donnez le rapport de checkstyle, spotbugs et PMD appliqués à votre code.

J'ai pris des captures d'écran de ces rapports avant la réécriture du code. Lorsque j'en suis venu à écrire le code, je le modifiais au fur et à mesure de ce que disait les rapports pour l'améliorer. Donc ici je ne vais mettre que les captures d'écran des rapports avant réécriture du code.

<u>Checkstyle :</u> (avec les règles Google Checks)

Mon screen ne montre pas toutes les erreurs (il y en a 40 en tout) mais c'est suffisant pour se rendre compte du nombre de problèmes dans le code initial.

Spotbugs :

Ne se lance pas à cause des erreurs de compilation

PMD :

21 problèmes sont reportés.

PMD report

Problems found

#	File	Line	Problem
1	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	2	All classes, interfaces, enums and annotations must belong to a named package
2	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	3	Field comments are required
3	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	4	Field comments are required
4	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	5	Field comments are required
5	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	5	To avoid mistakes add a comment at the beginning of the tva field if you want a default access modifier
6	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	5	Use explicit scoping instead of the default package private level
7	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	6	Avoid idempotent operations (like assigning a variable to itself).
8	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	6	Avoid reassigning parameters such as 'reference'
9	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	6	Public method and constructor comments are required
10	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	6	The value assigned to variable 'reference' is never used
11	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	12	Parameter 'prix' is not assigned and could be declared final
12	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	17	Avoid declaring a variable if it is unreferenced before a possible exit point.
13	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	17	Avoid unused local variables such as 'resultat'.
14	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	17	Found 'DU'-anomaly for variable 'resultat' (lines '17'-'19').
15	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	17	Local variable 'resultat' could be declared final
16	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	18	Avoid using if statements without curly braces
17	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	18	This statement should have braces
18	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	21	Avoid variables with short names like o
19	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	21	Ensure you override both equals() and hashCode()
20	C:\Users\Ewen CLEMENT\Documents\MASTER MIAGE\Ingénierie Logiciel (génie logiciel)\testsUnitaires\src\Produit.java	21	Parameter 'o' is not assigned and could be declared final

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

Après avoir tenu compte des problèmes soulevés par les trois outils, je n'ai plus eu aucun avertissement pas le checkstyle, spotbugs et PMD. Voici les principales actions de réécritures effectuées :

- Retrait du mot-clé final sur prix et référence
- Ajout du mot-clé final sur TVA (mis en majuscule car constante)
- Ajout d'un constructeur par défaut si pas de paramètres passés lors de l'instanciation
- Correction globale de l'indentation
- Ajout des commentaires pour chaque attributs et méthodes en suivant les bonnes pratiques
- Ajout de getter pour le prix (sans ça, un des rapports me proposait de mettre l'attribut en static)
- Ajout d'un setter pour la référence (sans quoi il proposait de mettre l'attribut en final, mais comme chaque référence est différente, on suppose qu'on puisse venir à les modifier)
- Modification de la méthode equals et ajout d'un hashCode sur le conseil du SpotBugs afin d'être sûr que la comparaison de 2 références se fasse correctement. J'ai utilisé le générateur de code d'Intellij pour cela et adapté en retirant la comparaison faite sur le prix pour ne garder que celle sur la référence.
- Rédaction de tests.

Ci-dessous, le code réécrit pour la classe Produit :

```java
package unc.nc.genielogiciel.model;

import java.util.Objects;

/**
 * Classe Produit représentant un produit avec un prix et une référence.
 */
public class Produit {

  /** Référence du produit. */
  private String reference;

  /** Prix du produit. */
  private double prix;

  /** Représente le taux de la TVA. */
  private static final double TVA = 0.20;

  /** Constructeur par défaut si pas de paramètres passés. */
  public Produit() {
    this.reference = "non référencé";
    this.prix = 0;
  }

  /**
   * Constructeur.
   *
   * @param reference la référence du produit.
   * @param prix le prix du produit.
   */
  public Produit(final String reference, final double prix) {
    this.reference = reference;
    this.prix = prix;
  }

  /**
   * Retourne le prix du produit.
   *
   * @return prix retourné.
   */
  public double getPrix() {
    return prix;
  }

  /**
   * Permet de fixer un nouveau prix au produit.
   *
   * @param prix nouveau prix.
   */
  public void setPrix(final double prix) {
    this.prix = prix;
  }

  /**
   * Permet d'obtenir la référence d'un produit
```

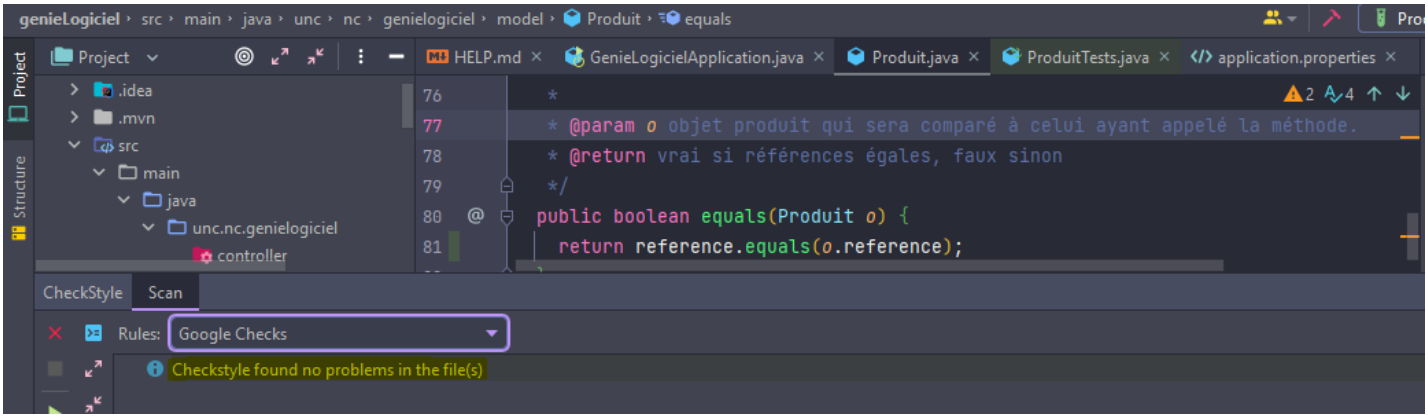
```
   *
   * @return la reference si le prix est strict positif, null sinon.
   */
  public final String getReference() {
    if (this.prix > 0) {
      return reference;
    } else {
      return null;
    }
  }

  /**
   * Permet de changer la référence d'un produit.
   *
   * @param reference la nouvelle référence.
   */
  public void setReference(final String reference) {
    this.reference = reference;
  }

  public double getTva() {
    return TVA;
  }

  /**
   * Compare la référence de deux produits.
   *
   * @param objet objet produit qui sera comparé.
   * @return vrai si références égales, faux sinon.
   */
  @Override
  public boolean equals(final Object objet) {
    if (this == objet) {
      return true;
    }
    if (objet == null || getClass() != objet.getClass()) {
      return false;
    }
    final Produit produit = (Produit) objet; // On fait un cast objet en
Produit
    return Objects.equals(reference, produit.reference); // Compare référence
des deux produits.
  }

  @Override
  public int hashCode() {
    return Objects.hash(reference);
  }
}
```

Ci-dessous, les tests écrits pour la classe Produit :

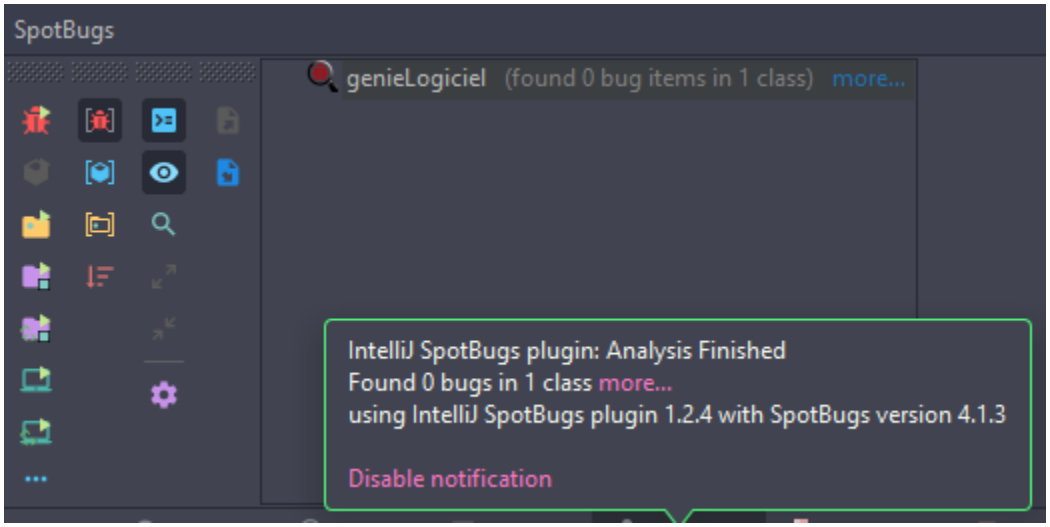
```java
package unc.nc.genielogiciel;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;
import unc.nc.genielogiciel.model.Produit;

import static org.junit.jupiter.api.Assertions.*;

@SpringBootTest
public class ProduitTests {

  Produit produit = new Produit("R2D2", 5000.0);
  Produit produit2 = new Produit();

  @Test
  public void produit() {
    produit.setPrix(10000);
    assertEquals(10000, produit.getPrix());
  }

  @Test
  public void produitRef() {
    assertEquals("R2D2", produit.getReference());
  }

  @Test
  public void produit2() {
    assertNull(produit2.getReference());
  }

  @Test
  public void sameProduct() {
    assertNotEquals(produit, produit2);
  }
}
```

5. Donnez les rapports appliqués au code réécrit.

<u>Checkstyle :</u>



Spotbugs :



PMD :

Seul PMD garde quelques problèmes :

- 3 problèmes qui indiquent qu'une classe ne devrait utiliser qu'un seul return. Pour contourner cela j'ai tente d'utiliser une variable booléenne locale qui serait retournée en fin de méthode mais cela remontait de nouveaux problèmes qui indiquaient que l'assignation de la valeur de cette variable n'était pas pertinente. Je suis donc retourné aux return.

- Le fait que la classe produit est une Data Class. D'après la documentation cela désigne une classe contenant peu de fonctionnalités et qui est donc suspecté d'être finalement peu utile et potentiellement supprimée. Étant donné que dans ce type d'exercice il est normal que la classe n'ait pas de fonctionnalités poussées, on ne peut pas faire grand-chose pour ne plus avoir ce problème.

PMD report

Problems found

#	File	Line	Problem
1	C:\Users\Ewen CLEMENT\Documents\LP MIAW\Spring Boot\genieLogiciel\src\main\java\unc\nc\genielogiciel\model\Produit.java	8	The class 'Produit' is suspected to be a Data Class (WOC=22.222%, NOPA=0, NOAM=5, WMC=13)
2	C:\Users\Ewen CLEMENT\Documents\LP MIAW\Spring Boot\genieLogiciel\src\main\java\unc\nc\genielogiciel\model\Produit.java	61	A method should have only one exit point, and that should be the last statement in the method
3	C:\Users\Ewen CLEMENT\Documents\LP MIAW\Spring Boot\genieLogiciel\src\main\java\unc\nc\genielogiciel\model\Produit.java	89	A method should have only one exit point, and that should be the last statement in the method
4	C:\Users\Ewen CLEMENT\Documents\LP MIAW\Spring Boot\genieLogiciel\src\main\java\unc\nc\genielogiciel\model\Produit.java	92	A method should have only one exit point, and that should be the last statement in the method

Exercice 2 : Tests unitaires

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

J'ai rédigé des tests unitaires, avec deux issues à chaque fois : celle où on obtient le résultat attendu (assertEquals ou assertTrue) ou bien le contraire où l'on obtient un résultat différent (assertNotEquals ou assertFalse). J'en ai aussi rédigé un pour le throw pour la méthode de la moyenne (étant la seule avec un throw dans le commentaire de la méthode).

Ci-dessous le code des tests unitaires :

```java
package unc.nc.genielogiciel;

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

import org.springframework.boot.test.context.SpringBootTest;
import unc.nc.genielogiciel.model.TabAlgosUtils;

import static org.junit.jupiter.api.Assertions.*;

@SpringBootTest
public class TabAlgosTests {

  @Test
  public void testMax() {
    int[] tab = new int[]{1, 24, 5, 30, 6, 18};
    assertEquals(30, TabAlgosUtils.plusGrand(tab));
  }

  @Test
  public void testMaxNot() {
    int[] tab = new int[]{1, 24, 5, 30, 60, 18};
    assertNotEquals(30, TabAlgosUtils.plusGrand(tab));
  }

  @Test
  public void testMoyenne() {
    int[] tab = new int[]{1, 24, 5, 30, 6, 18};
    assertEquals(14.0, TabAlgosUtils.moyenne(tab));
  }

  @Test
  public void testMoyenneNot() {
    int[] tab = new int[]{1, 24, 5, 30, 6, 18};
    assertNotEquals(11.0, TabAlgosUtils.moyenne(tab));
  }
```

```
  @Test
  public void testMoyenneThrow() {
    int[] tab = null;
    Exception exception = assertThrows(IllegalArgumentException.class, () ->
TabAlgosUtils.moyenne(tab));
    assertEquals("Le tableau fourni ne doit pas être null ou vide.",
exception.getMessage());
  }

  @Test
  public void testMoyenneThrowNot() {
    int[] tab = null;
    Exception exception = assertThrows(IllegalArgumentException.class, () ->
TabAlgosUtils.moyenne(tab));
    assertNotEquals("C'est un mauvais tableau", exception.getMessage());
  }

  @Test
  public void testEgaux() {
    int[] tab1 = new int[]{1, 24, 5, 30, 6, 18};
    int[] tab2 = new int[]{1, 24, 5, 30, 6, 18};
    assertTrue(TabAlgosUtils.egaux(tab1, tab2));
  }

  @Test
  public void testEgauxNot() {
    int[] tab1 = new int[]{1, 24, 5, 30, 6, 18};
    int[] tab2 = new int[]{1, 24, 5, 30, 6, 19};
    assertFalse(TabAlgosUtils.egaux(tab1, tab2));
  }

  @Test
  public void testSimilaires() {
    int[] tab1 = new int[]{2,2,1,0,4};
    int[] tab2 = new int[]{2,0,4,2,1};
    assertTrue(TabAlgosUtils.similaires(tab1, tab2));
  }

  @Test
  public void testSimilairesFalse() {
    int[] tab1 = new int[]{2,2,1,0,4};
    int[] tab2 = new int[]{0,1,4,2,3};
    assertFalse(TabAlgosUtils.similaires(tab1, tab2));
  }
}
```

2. Implémentez les méthodes en respectant les règles d'écriture contrôlées par les outils Checkstyle, Spotbugs et PMD.

Ci-dessous, le code avec les méthodes complétés et en respectant les règles d'écriture.

```
package unc.nc.genielogiciel.model;

import java.util.Arrays;

/**
 * Classe effectuant diverses opérations arithmétiques.
 */
public final class TabAlgosUtils {
  /**
   * Renvoie le plus grand entier d'un tableau.
   *
   * @return valeur la plus grande d'un tableau.
   */
  public static int plusGrand(final int... tab) {
    int maximum = Integer.MIN_VALUE;
    for (final int element : tab) {
      if (element > maximum) {
        maximum = element;
      }
    }
    return maximum;
  }

  /**
   * Calcul de la moyenne des entiers d'un tableau.
   *
   * @return moyenne des valeurs du tableau.
   * @throw IllegalArgumentException si tab et null ou vide.
   **/
  public static double moyenne(final int... tab) throws
IllegalArgumentException {
    if (tab == null || tab.length == 0) {
      throw new IllegalArgumentException("Le tableau fourni ne doit pas être
null ou vide.");
    }
    int somme = 0;
    for (final int element : tab) {
      somme += element;
    }
    return somme / (double) 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).
   **/
```

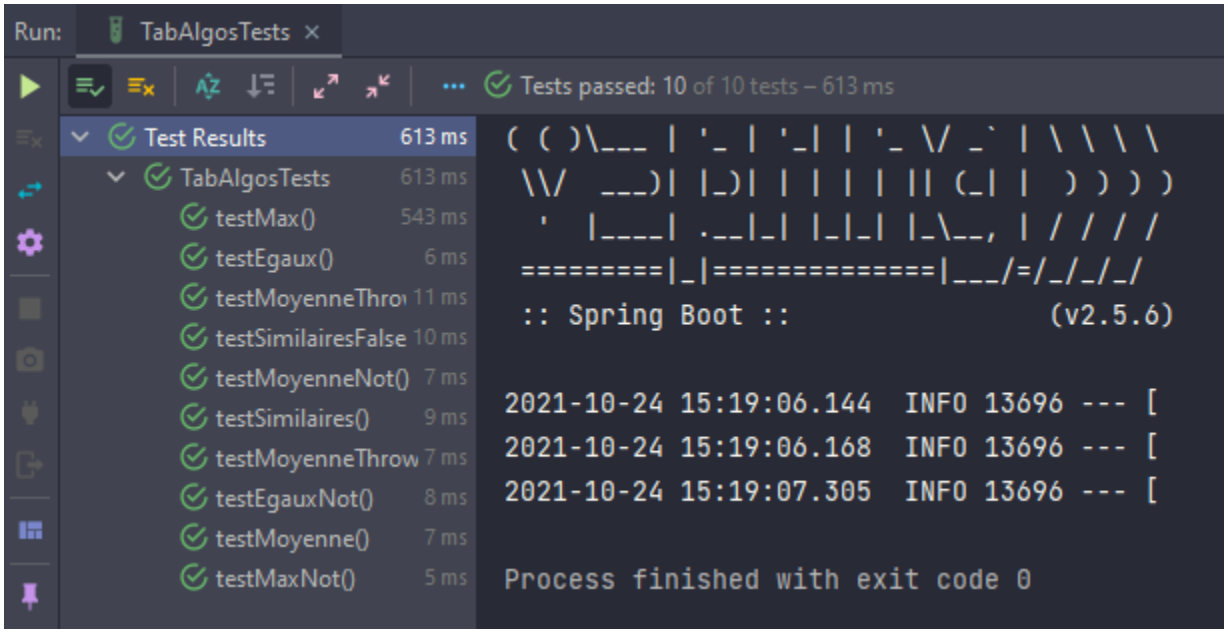
```java
  public static boolean egaux(final int[] tab1, final int... tab2) {
    for (int i = 0; i < tab1.length; i++) {
      if (tab1[i] != tab2[i]) {
        return false;
      }
    }
    return true;
  }

  /**
   * 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).
   **/
  public static boolean similaires(final int[] tab1, final int... tab2) {
    // Tableau qui contiendra les index des éléments du tab2 déjà trouvés.
    int [] tab2FindedIndex = new int[tab1.length];
    // On remplit ce tableau de -1, afin de ne pas être gêné pour l'index 0.
    Arrays.fill(tab2FindedIndex, -1);

    for (int i = 0; i < tab1.length; i++) {
      final int element = tab1[i];
      for (int j = 0; j < tab2.length; j++) {
        final int finalJ = j;
        // Si l'index du tableau 2 est présent dans notre tableau d'index
trouvés,
        // On fait un continue car on ne veut pas recomparer un élément déjà
trouvé.
        if (Arrays.stream(tab2FindedIndex).anyMatch(k -> k == finalJ)) {
          continue;
        }
        // Si élément du tab1 égal du tab2, on ajoute l'index du tab2 dans
        // tableau d'index trouvé, et on fait un break.
        if (element == tab2[j]) {
          tab2FindedIndex[i] = j;
          break;
        }
        // Si tab2 parcouru sans trouver de correspondance,
        // élément de tab1 n'est pas présent dans tab2, et on retourne false.
        if (j == tab2.length - 1) {
          return false;
        }
      }
    }
    // Si toutes les correspondances sont trouvées, on retourne true.
    return true;
  }

  private TabAlgosUtils() {}
}
```

Voici les principaux points qui avaient été relevés par les outils de vérification et que j'ai retravaillé :

- Transformer la classe TabAlgos en classe utilitaire, en modifiant son nom en TabAlgosUtils, en lui ajoutant le mot-clé final et en lui passant un constructeur vide.
- Ajout du mot-clé final sur les arguments des fonctions.
- Ajout d'espaces à certains endroit (dans les conditions des 'if' par exemples).
- Utilisation de varArgs pour les paramètres finaux des fonctions, qui est ici du sucre syntaxique.
- Ajout de commentaires de description de la classe et des méthodes.

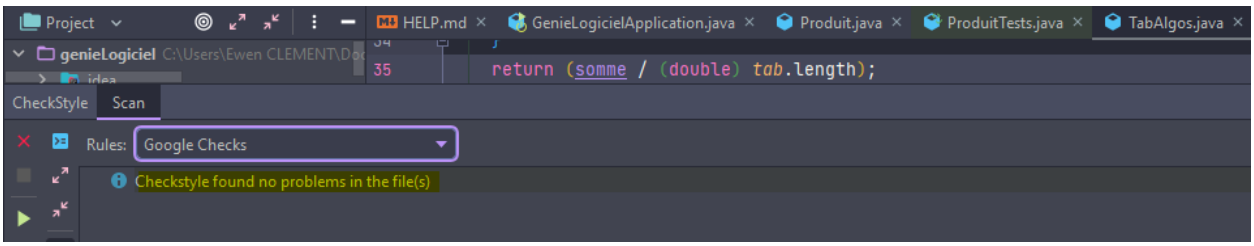
3. Vérifiez la validité des tests avec le code implémenté.

Tous les tests sont passés :

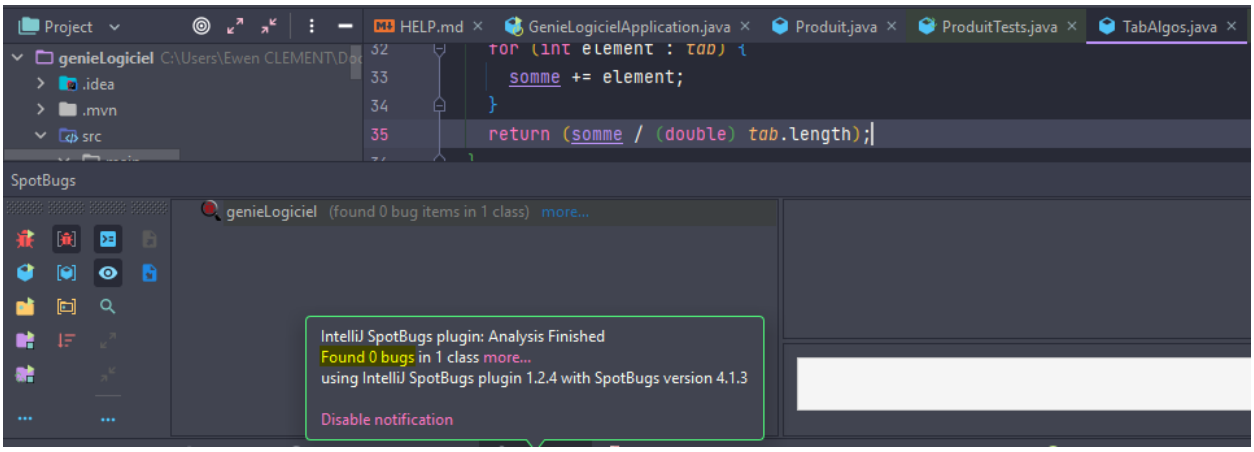


4. Fournir les rapports des tests unitaires, de checkstyle, spotbugs et PMD. Il ne doit rester aucune anomalie.

<u>Checkstyle :</u>



Spotbugs :



PMD :

Le seul où il reste encore des problèmes (3 dont 2 qui se répètent) :

- unique return : même problème que dans le premier exercice avec les mêmes raisons qui me poussent à ne pas le régler.

- DD et DU anomalie : ce problème est remonté lorsqu'une variable récente n'est pas définie ou redéfinie peu de temps après sa déclaration. Mais comme j'initialise directement les variables en question, je vois mal pourquoi ces erreurs apparaissent.

- Potentielle violation de la loi de Demeter qui consiste à ne pas modifier un élément par le biais d'un autre (exemple : définir un objet c qui a pour valeur le getter d'un objet b puis appeler une méthode de b par le biais de c). Dans mon cas, je fais simplement une recherche dans un tableau qui a été défini directement dans la méthode, donc aucune violation de ce genre.

PMD report

Problems found

#	File	Line	Problem
1	C:\Users\Ewen CLEMENT\Documents\LP MIAW\Spring Boot\genieLogiciel\src\main\java\unc\nc\genielogiciel\model\TabAlgosUtils.java	48	A method should have only one exit point, and that should be the last statement in the method
2	C:\Users\Ewen CLEMENT\Documents\LP MIAW\Spring Boot\genieLogiciel\src\main\java\unc\nc\genielogiciel\model\TabAlgosUtils.java	68	Found 'DD'-anomaly for variable 'element' (lines '68'-'68').
3	C:\Users\Ewen CLEMENT\Documents\LP MIAW\Spring Boot\genieLogiciel\src\main\java\unc\nc\genielogiciel\model\TabAlgosUtils.java	68	Found 'DU'-anomaly for variable 'element' (lines '68'-'91').
4	C:\Users\Ewen CLEMENT\Documents\LP MIAW\Spring Boot\genieLogiciel\src\main\java\unc\nc\genielogiciel\model\TabAlgosUtils.java	70	Found 'DD'-anomaly for variable 'finalJ' (lines '70'-'70').
5	C:\Users\Ewen CLEMENT\Documents\LP MIAW\Spring Boot\genieLogiciel\src\main\java\unc\nc\genielogiciel\model\TabAlgosUtils.java	70	Found 'DU'-anomaly for variable 'finalJ' (lines '70'-'91').
6	C:\Users\Ewen CLEMENT\Documents\LP MIAW\Spring Boot\genieLogiciel\src\main\java\unc\nc\genielogiciel\model\TabAlgosUtils.java	73	Potential violation of Law of Demeter (method chain calls)
7	C:\Users\Ewen CLEMENT\Documents\LP MIAW\Spring Boot\genieLogiciel\src\main\java\unc\nc\genielogiciel\model\TabAlgosUtils.java	79	Found 'DU'-anomaly for variable 'tab2FindedIndex' (lines '79'-'91').
8	C:\Users\Ewen CLEMENT\Documents\LP MIAW\Spring Boot\genieLogiciel\src\main\java\unc\nc\genielogiciel\model\TabAlgosUtils.java	85	A method should have only one exit point, and that should be the last statement in the method

Exercice 3 : Dépôt sur serveur de Versionning

Personnellement, j'utilise git et gitHub.

Lien du dépôt : https://github.com/EWEN14/Activites-Genie-Logiciel