#### Kokou Wama DJIMISSA

Activté 1 C306 - Ingénierie du logiciel

# E-Miage Session 2024 - Semestre 2

#### Exercice 1 : Ecriture de code

- 1. Raisons pour lesquelles le code ne compile pas :
  - Les variables d'instances **reference** et **prix** sont **final** et on ne peut leur affecter des valeurs.
  - Il manque l'instruction return dans la méthode getReference()
- 2. Correction du code pour qu'il compile : (Voir image ci-dessus)

```
1- /** Classe Produit représentant un produit avec un prix et une référence */
2- public class Produit {
3-
       private String reference;
4-
       private double prix;
5-
       double tva = 0.20;
6-
7-
       public Produit(String reference) {
8-
           reference = reference;
9-
10-
11-
       /** @return le prix */
12-
       public double getPrix() {
13-
           return this.prix;
14-
15-
16-
       /** modifie le prix */
17-
       public final void setPrix(Double prix) {
18-
           this.prix = prix;
19-
20-
21-
22-
        * @return la reference si le prix est positif, null sinon
23-
24-
       public final String getReference() {
25-
           String resultat = reference;
26-
           if (prix > 0)
27-
               return reference;
28-
           return null;
29-
30-
       @Override
31-
       public boolean equals(Object o) {
32-
           return reference == ((Produit) o).reference;
33-
34-}
```

- 3. Les rapports checkstyle, spotbugs et PMD sont rassemblés dans le répertoire Exercice\_1/rapport-avant et nommés chacun respectivement :
  - Exercice\_1/rapport-avant/checkstyle\_KokouWamaDJIMISSA.txt
  - Exercice\_1/rapport-avant/spotbugs\_kokouWamaDJIMISSA.html
  - Exercice\_1/rapport-avant/pmd 7.5.0\_KokouWamaDJIMISSA.html
- 4. Explication des problèmes soulevés par rapport :

Rapport	Explications
Checkstyle checkstyle_KokouWamaDJIMISSA.txt	[ERROR] C:\Users\kokou.djimissa\Documents\codesource\Produit.java :1: Le fichier package-info.java est manquant. [JavadocPackage]: indique que le répertoire où se trouve le code source manque le fichier package-info.java qui permet de documenter les packages existant dans le répertoire principal du code source et de le spécifier.
	[ERROR] C:\\Produit.java:1: La première ligne de la Javadoc doit se terminer avec un point. [JavadocStyle]: indique qu'il faut obligatoirement mettre un point à la fin de la première phrase du commentaire d'un code source.
	[ERROR] C:\\Produit.java: Produit.java:3:5: Commentaire Javadoc manquant: ce message d'erreur indique qu'il manque des commentaires sur certaines fonctions.
	[ERROR] C:\\Produit.java : 5:12: La variable d'instance 'tva' devrait être de type privée (pivate) et avoir des getters et setters. [VisibilityModifier]
	[ERROR] C:\\Produit.java:5:18: '0.20' devrait être défini comme une constante: la valeur d'affectation de tva doit provenir d'une constance et non être en dure dans le code comme ça.
	[ERROR] C:\\Produit.java:7:20: Le paramètre reference devrait être final. [FinalParameters]: indique que la paramètre reference du constructeur doit être final pour éviter que la valeur ne soit changée dans cette méthode.
	[ERROR] C:\\Produit.java:7:27: 'reference' masque un attribut. [HiddenField]: dans le constructeur il y a le paramètre reference et reference la variable d'instance. Mais l'expression écrite masque la variable d'instance (attribut).
	[ERROR] C:\\Produit.java:17:38: Balise Javadoc @param manquante pour 'prix'. [JavadocMethod]: JavaDoc exige la balise @Param si une méthode à un paramètre.

[ERROR] C:\ ...\Produit.java:26:9: L'instruction 'if' devrait utiliser des accolades ('{' et '}'). [NeedBraces]: if dans notre code n'utilise pas les accolades pourtant checkstyle le recommande.

[ERROR] C:\ ...\Produit.java :32:5: La définition de la méthode 'equals()' doit toujours être accompagnée de la définition de la méthode 'hashCode()'.

[Equals HashCode]: il manque la méthode hashCode(), recommandé quand equals() est définie.

[ERROR] C:\ ...\Produit.java :32:5: La classe 'Produit' semble être conçue pour extension (peut être héritée), mais la méthode 'equals' n'a pas de Javadoc qui explique comment le faire en toute sécurité. Si la classe n'est pas conçue pour extension, envisagez de rendre la classe 'Produit' finale ou de rendre la méthode 'equals' static/final/abstract/empty, d'ajouter les annotations permises pour la méthode. [DesignForExtension]: indique que la méthode "equals" n'a pas de commentaire. A cause de l'annotation @Override, la classe est prise comme une classe qui hérite d'une autre or ce n'est pas le cas. Pour corriger cela deux

### Spotbugs

## spotbugs\_kokouWamaDJIMISSA.html

Ces deux rapports présentent plusieurs types d'erreurs à savoir : erreur de mauvaise pratique en programmation, des avertissements d'exactitude, de performance et de code douteux.

solutions sont proposées (1) rendre la classe Produit final ou la

#### PMD

### pmd7.5.0\_KokouWamaDJIMISSA.html

Le paramètre de la méthode "equals" de type Produit n'est pas une bonne pratique.

L'opérateur == n'est pas conseiller pour comparer deux objets String.

Il faut nécessairement définir la méthode hashCode() si on définit equals(). Puisque ''equals'' utilise hashCode() pour comparer les objets.

Des champs inutilisés qui agiront sur la performance, mauvaise utilisation des variables (auto-alimentation).

Accolade manquant de l'instruction if ...

5. Le code corrigé est la version de la classe *codesource/Produit.java* qui se trouve dans le travail rendu.

méthode "equals" static.

## Différentes Commande exécutées :

\$> java -jar checkstyle-10.18.1-all.jar -c sun\_checks.xml
codesource\Produit.java

\$> java -jar lib\spotbugs.jar -html -output rapportspotbugs\_KokouWamaDJIMISSA.html ..\codesource\Produit.class

\$> pmd.bat check -f html -R ..\rulesets\java\quickstart.xml -d
..\..\codesource\

- 6. Les rapports **checkstyle**, **spotbugs** et **PMD** appliqués au code réécrit sont rassemblés dans le répertoire **Exercice\_1/rapport-apres** et nommés chacun respectivement :
  - Exercice\_1/rapport-apres/checkstyle\_KokouWamaDJIMISSA.txt
  - Exercice\_1/rapport-apres/spotbugs\_kokouWamaDJIMISSA.html
  - Exercice\_1/rapport-apres/pmd 7.5.0\_KokouWamaDJIMISSA.html

#### **Exercice 2:** Tests Unitaires

1. Classification des tests

Méthode	Classes d'équivalence	Instances de tests au bord	Tests au bord
plusGrand (tab)	Tableau avec des	{3, 4,5, 6, 7}	Tableau avec deux
	entiers positives		entiers identiques
	Tableau avec des	{-6, -5, -4, -3, -2, -1}	Tableau avec un seul
	entiers négatives		entier.
	Tableau avec un entier	{15}	
	Tableau vide ou null	{}ou null	
moyenne(tab)	Tableau avec des	{3, 4,5, 6, 7}	Tableau vide : provoque
	entiers positives		une exception
	Tableau avec des	{-6, -5, -4, -3, -2, -1}	
	entiers négatives		
	Tableau vide ou null	{} ou null	
	Tableau avec un entier	{15}	
egaux()	Deux tableaux égaux	{7, 8, 9} et {7, 8, 9}	
	avec les mêmes		
	données		
	Deux tableaux différents	{1, 2, 3} et {4, 5, 6}	
	avec même nombre des		
	données différentes		
	Deux tableaux de	{1, 2, 3} et {3,4,5,7}	
	différent nombre de		
	données		
	Tableau vide ou null	{} ou null	
similaires()	Deux tableaux de même	{7, 8, 9} et {9, 8, 7}	
	taille avec les mêmes		
	données dans un ordre		
	différent.		

Deux tableaux différents	{1, 2, 3} et {4, 5, 6}	
Deux tableaux de	{2,3,4} et {2,3}	
différente taille		
Tableau vide ou null	{} ou null	

- 2. Les tests unitaires sont dans la classe Exercice\_2/codesource/TabAlgoTest.java
- 3. Les méthodes implémentées sont dans la classe

# Exercice\_2/codesource/TabAlgos.java

5. Les rapports des test unitaires, de checkstyle, spotbugs et PMD :

## Exercice\_2/rapport et nommés chacun respectivement :

- Exercice\_2/rapport/rapport-testunitaires.txt
- Exercice\_2/rapport/tabAlgo-checkstyle-KokouWamaDJIMISSA.html
- Exercice\_2/rapport/tabAlgo-pmd-KokouWamaDJIMISSA.html
- Exercice\_2/rapport/tabAlgos-checkstyle\_KokouWamaDJIMISSA.txt