AVIGNON UNIVERSITÉ

Rapport TP6

Maryem Bouziane

09/04/2024

Master 1 en Informatique
intelligence artificielle

UE Techniques de tests

Responsable
Ludovic Bonnefoy

UFR
SCIENCES
TECHNOLOGIES
SANTÉ

AVIGNON UNIVERSITÉ
CERI

CENTRE
D'**ENSEIGNEMENT**
ET DE **RECHERCHE**
EN **INFORMATIQUE**
ceri.univ-avignon.fr

Sommaire

1 Résumé

L'objectif de TP6 est d'intégrer et évaluer une implémentation de `IPokemonFactory` fournie par une autre équipe (Team Rocket) à travers des tests et une revue de code, afin d'identifier et corriger les éventuels défauts et améliorer la qualité du projet.

2 Intégration de l'Implémentation Fournie par la Team Rocket

J'ai copié-collé la classe RocketPokemonFactory dans le package contenant les autres classes, donc la première phase consistait à adapter le package :

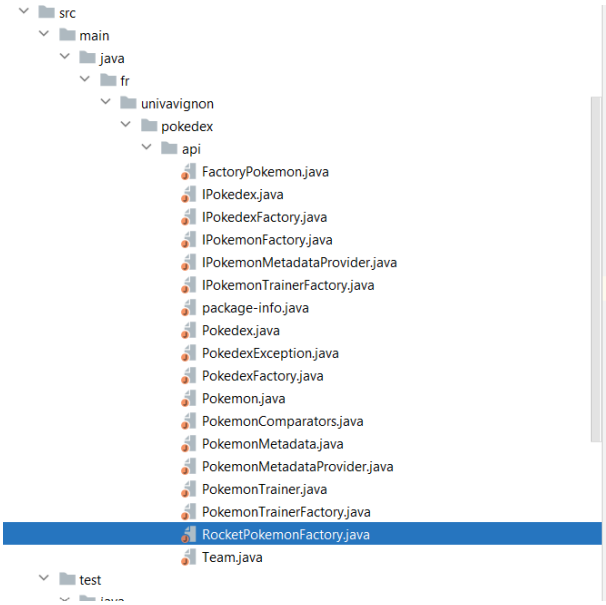

Figure 1. ajout de la classe dans le projet

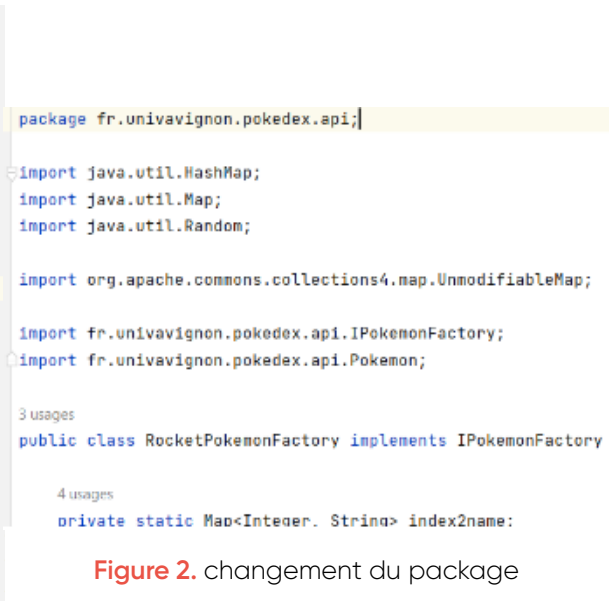

Figure 2. changement du package

et aprés, j'ai testé la compatibilité avec mvn verify :

Figure 3. verification des dependences

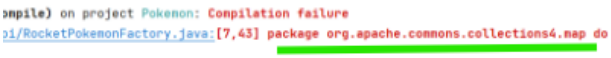

Figure 4. dependence manquante

nous avons touvé qu'il manque la dependence suivante pour que l'integration marche :

```xml
<!-- dependency pour RocketPokemonFactory  -->
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-collections4</artifactId>
    <version>4.0</version>
</dependency>
```

Figure 5. dependence ajoutée

3 utilisation de ma suite de tests

3.1 Test CircleCI

J'ai dupliqué ma classe **IpokemonFactoryTest** en une classe **RocketPokemonFactoryTest**. J'ai modifié un seul élément : mon setup. Au lieu d'utiliser mon implémentation **FactoryPokemon**, cette fois-ci, j'ai utilisé l'implémentation **RocketFactoryPokemon** pour tester le comportement de cette dernière :

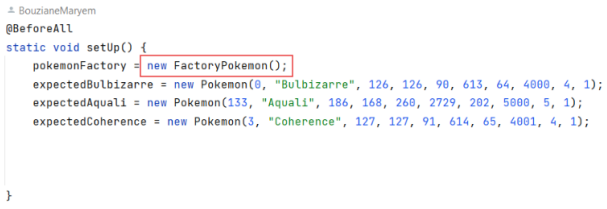

Figure 6. test avec mon implementation

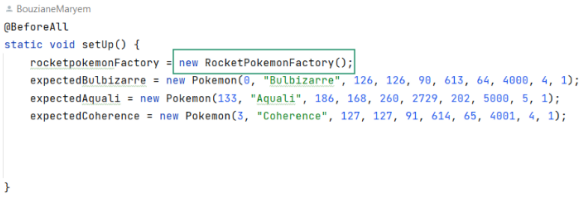

Figure 7. test avec l'équipe rocket

J'ai conservé les mêmes tests pour **FactoryPokemon** et j'ai lancé CircleCI pour examiner les résultats des tests :

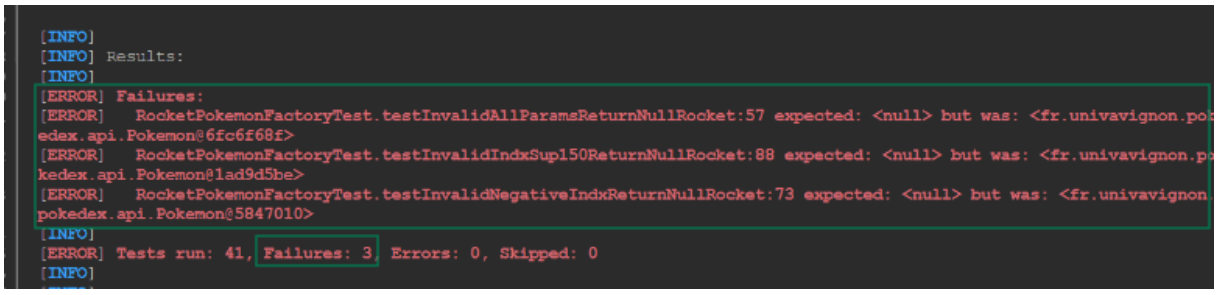

Figure 8. Resultats de tests de circle ci

Un total de 41 tests ont été réalisés, dont 3 ont échoué, comme illustré dans les résultats de CircleCI ci-joints. Aucun test n'a été ignoré ou n'a abouti à des erreurs autres que des échecs de test. Cela indique une bonne intégrité de la suite de tests mais révèle des lacunes dans l'implémentation. Les tests échoués sont les suivants :

- **Test de Paramètres Totalement Invalides** : Le test `testInvalidAllParamsReturnNullRocket` a échoué, révélant que la méthode retourne une instance de `Pokemon` au lieu de `null` lorsque tous les paramètres sont invalides.
- **Test d'Indice Supérieur à la Plage Acceptée** : Le test `testInvalidIndexSup150ReturnNullRocket` a échoué car la méthode produit une instance de `Pokemon` pour un indice supérieur à 150, ce qui est contraire à l'attente de retourner `null` .
- **Test d'Indice Négatif** : Le test `testInvalidNegativeIndexReturnNullRocket` a montré que la méthode ne gère pas correctement les indices négatifs, retournant une instance de `Pokemon` au lieu de `null` .

Ces trois cas de test étaient censés valider la robustesse de la `RocketPokemonFactory` en vérifiant la gestion des paramètres invalides. L'attente était que la fabrique retourne `null` dans les cas où les paramètres d'entrée sont soient tous invalides, soient hors des bornes acceptées (index négatif ou supérieur à la plage prévue).

3.2 Couverture Codecov

j'ai verifié aussi la couverture des tests dans codecov pour l'implementation de l'équipe Rocket

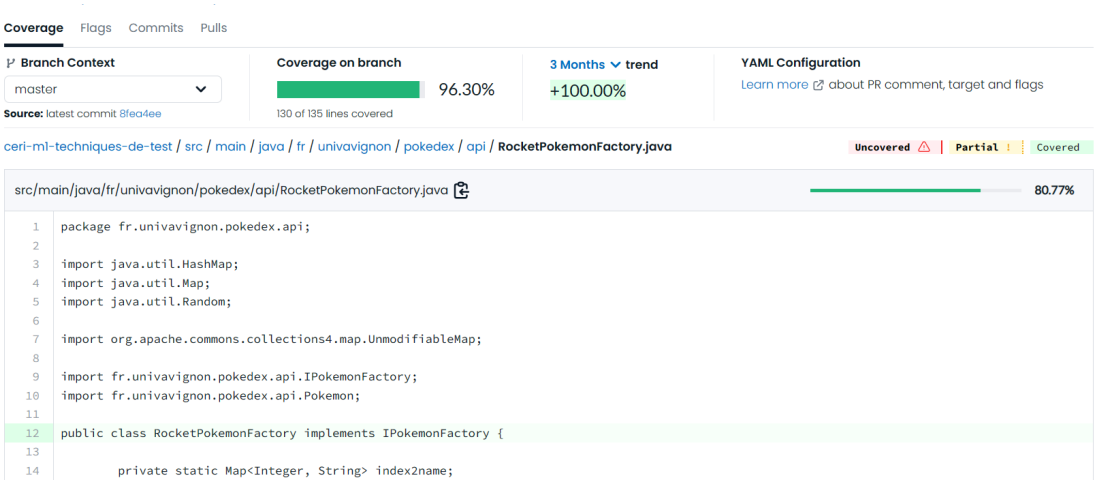

Figure 9. couverture codecove

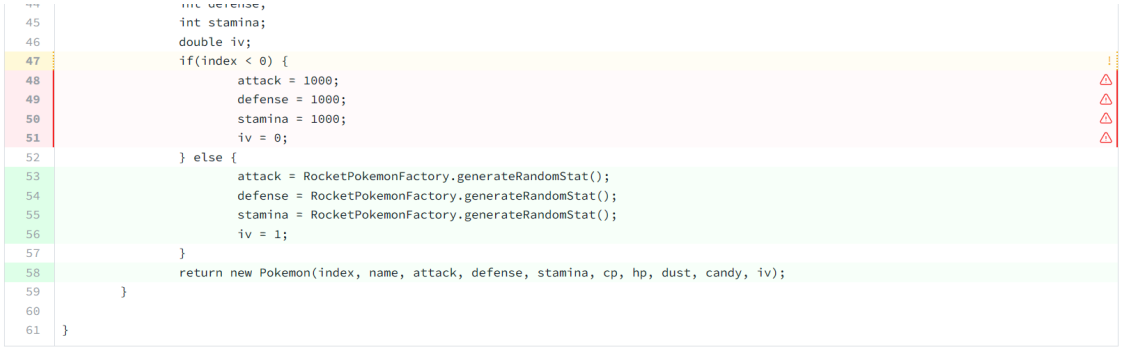

Figure 10. couverture codecov

La couverture de Codecov montre que $80{,}77\%$ de `RocketPokemonFactory` est testée. Cependant, les tests ne couvrent pas les situations où l'indice est négatif. Mon code original gérait cela en appelant `getPokemonMetadata` de `PokemonMetaProvider`, qui, en cas d'erreur, levait une exception et `RocketPokemonFactory` retournait `null` :

```
try {
    PokemonMetadataProvider provider = new PokemonMetadataProvider();
    pokemonMetadata = provider.getPokemonMetadata(index);
} catch (PokedexException pe) {
    System.err.println("Error PokemonMetadataProvider !!!");
    return null;
}
```

Le code de la Team Rocket attribue des valeurs très élevées égales à 1000 aux statistiques en cas d'indice négatif :

```
if(index < 0) {
    attack = 1000;
```

```
    defense = 1000;
    stamina = 1000;
    iv = 0;
}
```

Cette manière de procéder n'est pas considérée comme une bonne pratique, car elle crée une instance de Pokemon même avec des indices invalides, ce qui ne devrait pas arriver. Il est important de corriger cela pour que le code réponde bien quand il rencontre des indices qui ne sont pas valides.

4 Revue de Code

Maintenant , j'ai analyser le code manuellement , et j'ai le code de l'implémentation de IPokemonFactory

4.1 Qualité de code- checkstyle

en utilisant les regles google de check style , nous avons identifié 24 violations :

```
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] There are 24 errors reported by Checkstyle 10.15.0 with sun_checks.xml ruleset.
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[1] (misc) NewlineAtEndOfFile: Il manque un caract
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[9,1] (imports) RedundantImport: Import redondant
 fr.univavignon.pokedex.api.IPokemonFactory.
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[10,1] (imports) RedundantImport: Import redondant
- fr.univavignon.pokedex.api.Pokemon.
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[13] (regexp) RegexpSingleline: Line has trailing
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[13,1] (whitespace) FileTabCharacter: Le fichier c
i├¿re occurence).
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[14,9] (javadoc) JavadocVariable: Commentaire Javac
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[23] (regexp) RegexpSingleline: Line has trailing
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[26,17] (whitespace) WhitespaceAfter: Il manque une
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[26,17] (whitespace) WhitespaceAround: Il manque un
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[26,26] (whitespace) WhitespaceAround: Il manque un
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[26,26] (whitespace) WhitespaceAround: Il manque un
t ├¬tre h├®rit├®e), mais la m├®thode'createPokemon' n'a pas de Javadoc qui explique comment le faire en toute s├®curit├®. Si
ndre la classe 'RocketPokemonFactory' finale ou de rendre la m├®thode 'createPokemon' static/final/abstract/empty, d'ajouter
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[36] (sizes) LineLength: La ligne exc├¿de 80 caract
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[36,38] (misc) FinalParameters: Le param├¿tre inde
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[36,49] (misc) FinalParameters: Le param├¿tre cp de
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[36,57] (misc) FinalParameters: Le param├¿tre hp de
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[36,65] (misc) FinalParameters: Le param├¿tre dust
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[36,75] (misc) FinalParameters: Le param├¿tre candy
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[38,17] (whitespace) WhitespaceAfter: Il manque une
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[38,17] (whitespace) WhitespaceAround: Il manque un
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[47,17] (whitespace) WhitespaceAfter: Il manque une
[ERROR] src\main\java\fr\univavignon\pokedex\api\RocketPokemonFactory.java:[47,17] (whitespace) WhitespaceAround: Il manque un
```

Figure 11. resultat de checkstyle

le fichier checkstyle-rocket contient le resultat de checkstyle que vous pouvez examinés , en cliquant sur le lien suivant : https://github.com/BouzianeMaryem/ceri-m1-technique

s-de-test/blob/master/target/RocketPokemonFactory-checkstyle.xml.

Suite à l'application des règles de style de Google via `Checkstyle`, la classe `RocketPokemonFactory` montre plusieurs points à corriger :

- Des espaces manquants apparaissent souvent, notamment après `'for'`, `'if'`, ou autour des signes `'='`.
- Il existe des imports superflus puisque les classes importées sont déjà dans le même package.
- Les espaces en fin de ligne et l'utilisation de tabulations au lieu d'espaces pour l'indentation nécessitent une correction.
- Des commentaires explicatifs sont absents pour certaines variables et méthodes.
- Certaines lignes dépassent la longueur recommandée, ce qui peut gêner la lecture du code.
- Plusieurs paramètres de méthodes devraient être déclarés comme `final` pour éviter des modifications non prévues.

Il est essentiel de résoudre ces problèmes pour que le code soit propre, conforme aux lignes directrices de Google, et facile à comprendre pour tous les développeurs.

4.2 mauvaises pratiques de codage

1. **Génération aléatoire inefficace :** Créer un nouvel objet `Random` à chaque itération de la boucle (1 000 000 de fois) est inefficace.
2. **Calcul incorrect de la statistique :** La méthode actuelle aboutit à des statistiques de Pokémon irréalistes en divisant par 10 000 au lieu du nombre total d'itérations (1 000 000).
3. **Initialisation complexe de la carte :** L'utilisation de `UnmodifiableMap` pour initialiser `index2name` est plus complexe que nécessaire.
4. **Indices manquants :** Utiliser `MISSINGNO` pour tout indice non défini, simplifie trop la gestion des absences.

5 Amelioration possibles

1. **Optimisation de Random :** Créer un seul objet `Random` avant la boucle et générer des nombres aléatoires entre 0 et 150 pour réduire la plage de valeurs et éviter de recréer

l'objet à chaque itération.

2. **Ajustement des statistiques :** Pour des résultats réalistes, diviser la somme totale par 1 000 000, correspondant au nombre réel d'itérations effectuées.
3. **Simplification de l'initialisation :** Remplacer `UnmodifiableMap` par une `List` ou `ArrayList`, qui sont plus simples à manipuler pour initialiser `index2name`.
4. **Indices manquants et invalides :** Utiliser des exceptions pour gérer les cas où un indice n'est pas trouvé ou hors limites (supérieur à 150 ou inférieur à 0), plutôt que d'attribuer systématiquement 'MISSINGNO'.