Rapport de Test et Revue de Code

Projet - TP6 : Giovanni's Touch

December 18, 2024

Introduction

Dans ce rapport, je vais analyser l'implémentation de la classe `RocketPokemonFactory`, fournie par la Team Rocket, en fonction des critères de qualité du code, des tests automatisés et des bonnes pratiques. Mon objectif est d'intégrer cette classe dans notre projet, d'exécuter notre suite de tests existante, de repérer les défauts et de proposer des améliorations concrètes.

Résultats des Tests

Après avoir intégré `RocketPokemonFactory`, j'ai observé plusieurs anomalies lors de l'exécution des tests. Voici les erreurs que j'ai relevées :

Erreurs observées

- `testCreatePokemonInvalidIndex :` Ce test s'attendait à ce que l'exception `PokedexException` soit levée lorsqu'un indice invalide est fourni. Pourtant, aucune exception n'a été levée, ce qui indique une gestion insuffisante des cas d'indices incorrects.
- `testCreatePokemonMetadataException :` En cas d'erreur de récupération des métadonnées, je m'attendais à ce qu'une exception `PokedexException` soit levée, mais ce n'est pas le cas.
- `testCreatePokemonValid :` Lorsque j'ai testé la création d'un `Pokemon` avec un index valide, le test s'attendait au nom `Bulbasaur`, mais j'ai constaté que la valeur obtenue était `MISSINGNO`. Cela révèle un problème dans la logique de mappage des noms de Pokémon.

Défauts Identifiés

En plus des erreurs relevées par les tests, j'ai procédé à une analyse approfondie du code pour repérer d'autres problèmes :

- **Utilisation d'un `UnmodifiableMap` incomplet :** Le `Map index2name` ne contient que trois entrées, ce qui limite la gestion des indices valides.
- **Boucle inefficace dans `generateRandomStat()` :** J'ai noté que cette méthode effectue **1 000 000** itérations inutiles, réduisant considérablement les performances.

- **Gestion incorrecte des indices négatifs :** Pour des indices négatifs, des valeurs arbitraires (`attack`, `defense`, `stamina` fixées à 1000) sont utilisées sans justification claire.
- **Clés absentes dans `index2name` :** J'ai observé que lorsqu'un indice est absent, la valeur par défaut `"MISSINGNO"` est retournée, ce qui n'est pas une solution optimale.
- **Calcul incorrect dans `generateRandomStat()` :** J'ai constaté que la division par `10 000` produit des résultats incohérents.

Revue de Code

Points Positifs

- La classe respecte la signature de l'interface `IPokemonFactory`.
- L'utilisation de `UnmodifiableMap` garantit l'immuabilité des données.

Points Négatifs et Suggestions

- **Initialisation du `Map` :** J'ai identifié le besoin d'étendre cette carte pour inclure plus d'indices ou de gérer dynamiquement les entrées inconnues.
- **Efficacité de `generateRandomStat()` :** Remplacer cette méthode par un appel direct à `Random.nextInt()` améliorerait significativement les performances.
- **Indices négatifs :** J'estime qu'il serait préférable de lever une exception ou de définir une logique métier claire pour ces cas.
- **Clés absentes :** Je recommande de retourner une valeur plus explicite que `"MISSINGNO"`, ou de lever une exception pour signaler ces cas.

Conclusion

J'ai pu mettre en évidence plusieurs défauts dans l'implémentation de la Team Rocket qui impactent les performances et la logique métier. Pour corriger ces problèmes, voici les recommandations que je propose :

- Optimiser la méthode `generateRandomStat()` pour réduire la complexité.
- Étendre ou gérer dynamiquement le `Map index2name`.
- Documenter ou revoir la logique pour les indices négatifs.

Après avoir apporté ces modifications, je suis convaincu que cette implémentation pourrait s'intégrer de manière efficace dans notre projet.