AVIGNON UNIVERSITÉ

Rapport

Guillaume Bonenfant

19 mai 2022

M1 Informatique
ILSEN
UE Techniques de test
UCE Génie logiciel avancé

Responsables
Prénom5 Nom5
Emmanuel Ferreira

UFR
SCIENCES
TECHNOLOGIES
SANTÉ

AVIGNON UNIVERSITÉ
CERI

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

Sommaire

1 Implémentation de `RocketPokemonFactory`

1.1 Implémentation

Pour l'implémentation des tests de la classe `RocketPokemonFactory` (qui remplace `PokemonFactory`), une première erreur au niveau du code est apparue : le constructeur ne demandait aucun argument alors que celui de `PokemonFactory` demandait une variable de type `IPokemonMetadataProvider`.

1.2 tests

En ce qui concerne les tests, j'ai 6 erreurs qui sont apparus en remplaçant `PokemonFactory` par `RocketPokemonFactory`.

La première est due à la création d'un pokemon avec l'id 0 qui ne correspond pas au bon pokemon. En effet, Bulbizarre est censé être le pokemon avec cet id mais dans la classe `RocketPokemonFactory`, le pokemon correspondant est "MISSINGNO"

Puis, avec le test sur une création de pokemon avec un id au-delà de ce qui est autorisé, le pokemon devrait correspondre à `null` mais est encore une fois "MISSINGNO".

J'ai également des erreurs dues à des variables de pokemons incorrectes, par exemple `getAttack()` qui ne renvoie pas une attaque de pokemon attendue.

2 Conclusion

Grâce à ces tests, il est possible de rapidement voir que la classe `RocketPokemonFactory` a des problèmes et n'est pas bien codée. En effet, lorsque l'on implémente cette classe, il est possible de voir que les tests ne passent plus et il est donc possible d'en conclure que des modifications dans la classe sont nécéssaires. En effet, il ne faut pas modifier les tests pour qu'ils passent, mais modifier la classe pour qu'elle fasse passer les tests car cela veut dire que nous n'obtenons actuellement pas les résultats attendus !

Enfin, grâce à ces tests, en plus de rapidement remarquer qu'il y a des erreurs dans la classe, il est également possible de rapidement cibler les erreurs et savoir où elles se situent, par exemple, il est facile de remarquer qu'il y a des erreurs dans la méthode `createPokemon()` de `RocketPokemonFactory` au niveau de l'initialisation de l'attaque, défense et stamina où les valeurs sont fixées à 1000 si jamais l'id est inférieur à 0 et sont choisies aléatoirement sinon alors qu'elles devraient avoir des valeurs précises à plus ou moins 15 de différence. Il est donc facile de maintenant modifier ces méthodes à l'aide de l'analyse des tests.