AVIGNON UNIVERSITÉ

TP 6

IA:ALT

IDRISSOU FATAI

31 mars 2024

Master Informatique
INTELLIGENCE ARTIFICIELL
UE Technique de Test

Responsables
BONNEFOY Ludovic

UFR
SCIENCES
TECHNOLOGIES
SANTÉ

AVIGNON UNIVERSITÉ
CERI

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

Sommaire

1 Introduction

Dans le cadre de ce projet, nous avons intégré l'implémentation de IPokemonFactory fournie par la Team Rocket dans notre code Java. Nous avons ensuite évalué cette implémentation à travers nos tests existants et une revue de code approfondie afin de déterminer sa qualité et sa fiabilité.

2 RocketPokemonFactory

```java
@Override
public Pokemon createPokemon(int index, int cp, int hp, int dust, int candy) {
    String name;
    if (!index2name.containsKey(index)) {
        name = index2name.get(0);
    } else {
        name = index2name.get(index);
    }
    int attack;
    int defense;
    int stamina;
    double iv;
    if (index < 0) {
        attack = 1000;
        defense = 1000;
        stamina = 1000;
        iv = 0;
    } else {
        attack = RocketPokemonFactory.generateRandomStat();
        defense = RocketPokemonFactory.generateRandomStat();
        stamina = RocketPokemonFactory.generateRandomStat();
        iv = 1;
    }
    return new Pokemon(index, name, attack, defense, stamina, cp, hp, dust, candy, iv);
}
```

2.1 Description génerale :

La classe RocketPokemonFactory implémente l'interface IPokemonFactory et fournit une méthode createPokemon pour créer des instances de la classe Pokemon. Elle utilise une table de hachage pour mapper les indices des Pokémon à leurs noms. De plus, elle génère aléatoirement les statistiques d'attaque, de défense et de points de vie des Pokémon, avec une méthode generateRandomStat, lors de la création de nouveaux Pokémon, garantissant ainsi une variabilité dans les caractéristiques des Pokémon créés

2.2 CreatePkemon

La fonction commence par récupérer le nom du Pokémon correspondant à l'indice donné. Si l'indice n'est pas trouvé dans la table de hachage index2name, le nom par défaut est attribué (ici, le nom associé à l'indice 0).

Ensuite, elle détermine aléatoirement les statistiques d'attaque, de défense et de points de vie du Pokémon. Si l'indice est inférieur à zéro, les statistiques sont fixées à des valeurs arbitrairement élevées (1000) et l'indice de valeur individuelle (iv) est fixé à 0. Sinon, les statistiques sont générées aléatoirement à l'aide de la méthode generateRandomStat() de la classe RocketPokemonFactory et l'indice de valeur individuelle est fixé à 1.

Finalement, la fonction crée une nouvelle instance de la classe Pokemon avec les paramètres fournis et les statistiques calculées, puis la retourne.

2.3 Détails clés

- Cette implémentation crée des Pokémon avec des attributs (index, CP, HP, dust, candy) conformes aux paramètres passés à la méthode de création.
- Les statistiques d'attaque, de défense et de stamina ainsi que la valeur individuelle (IV) des Pokémon sont générées de manière aléatoire, ce qui ne répond pas entièrement aux critères de conformité aux tests.
- L'implémentation autorise la création de Pokémon avec des index dont la valeur est en dehors de la plage valide, ce qui pourrait entraîner des comportements indésirables.

3 Conclusion

Dans le cadre de ce projet, nous avons intégré l'implémentation de IPokemonFactory fournie par la Team Rocket dans notre code Java. Après évaluation, bien que la classe RocketPokemonFactory présente des fonctionnalités prometteuses telles que la création de Pokémon conformes aux paramètres spécifiés, des lacunes subsistent notamment en ce qui concerne la génération aléatoire des statistiques et l'autorisation de créer des Pokémon avec des index en dehors de la plage valide. Ainsi, des améliorations sont nécessaires pour garantir une conformité totale aux critères de qualité et de fiabilité dans notre projet Java.

4 References