TP2 : Pokeunit

Episode 1 - Tests unitaires

Place aux tests unitaires ! On voudra une couverture maximale (mais pertinente) de notre API. Il faut donc des tests pour toutes les interfaces de notre application, les tests de nos interfaces pourrons ensuite être réutilisés sur les implémentations pour s'assurer de leurs bon fonctionnement.

Tout d'abord il vous faut créer un nouveau repertoire `src/test/java` et un package `fr.univavignon.pokedex.api` pour vos tests, et mettre à jour votre *POM* afin d'associer le repertoire crée comme un repertoire contenant des tests unitaires. Voici la liste des *test cases* attendus :

- **`IPokemonMetadataProviderTest`**
- **`IPokemonFactoryTest`**
- **`IPokedexFactoryTest`**
- **`IPokedexTest`**
- **`IPokemonTrainerFactoryTest`**

Il est fortement recommandé de mettre pour chaque classe de test une couche d'abstraction pour récupérer des instances d'objets cible, qui seront dans le cadre des tests de l'API des *mocks* crées avec **Mockito**, et de vraies instances pour les tests d'implémentation.

Episode 2 - Catch the API

L'idée est de stocker des informations sur des *Pokemons*, dans un conteneur qu'est le *Pokedex*. Dans le cadre de ce TP nous nous contenterons de la 1ere génération[1], soit 151 espèces. Une espèce de *Pokemon* est décrite par des métadonnées communes à chaque individu de l'espèce à savoir :

---

[1] #LameilleureDesGénérationsPokemon.

- Un index numérique (allant de 0 à 150)
- Un nom
- Un niveau d'attaque[2]
- Un niveau de defense
- Un niveau d'endurance ou stamina

Ces informations sont représentées par la classe `PokemonMetadata`. Les métadonnées décrivent une espèce, alors qu'un individu est défini par la classe `Pokemon`, défini par les attributs suivant :

- Un niveau de combat ou CP
- Un niveau de vie ou HP
- Un niveau de poussière d'étoile ou dust
- Un nombre de bonbon ou candy
- Un pourcentage de perfection
- Des métadonnées

Les métadonnées d'un individu ne doivent pas être confondues avec les métadonnées de l'espèce. En effet, les valeurs des niveaux d'attaque, de defense, ou d'endurance sont bornées entre 0 et 15 dans le cadre d'un individu. Le niveau pour une statistique donnée se calcule ainsi :

```
Niveau de base de l'espèce + Niveau de l'individu
```

La description de la perfection ainsi que les autres attributs d'un individu fera l'objet d'un autre TP, lors de l'implémentation. Revenons à notre API, et passons à la description des interfaces de notre système :

- `IPokemonMetadataProvider` est en charge pour un index donné de retourner les métadonnées d'une espèce.
- `IPokemonFactory` permet de créer un individu.
- `IPokedex` est notre conteneur, qui étend les deux interfaces précédentes, qu'il fourni à travers le pattern *Décorateur*.
- `IPokedexFactory` permet de créer une instance de *Pokedex*.
- Un *Pokedex* appartient à un `PokemonTrainer`, défini par un nom et une équipe.
- Les `PokemonTrainer` sont crées à travers l'interface `IPokemonTrainerFactory`.

Avec toutes ces informations vous pouvez maintenant écrire des tests pour toutes les interfaces et créer les *mocks* nécessaires.

---

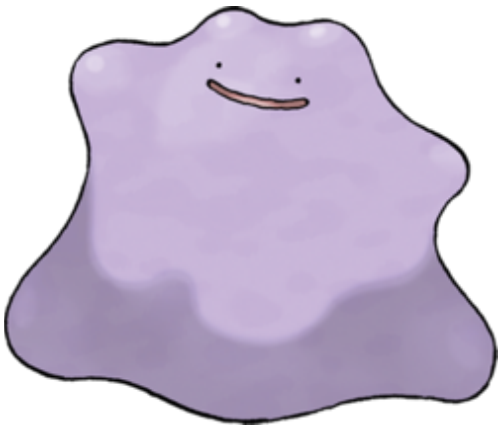
[2] Les statistiques tels que l'attaque, la défense ou l'endurance sont des valeurs entières

Episode 3 - Use cases

Afin d'avoir des *mocks* réalistes, voici deux instances de *Pokemon* ainsi que les métadonnées de leur espèce que vous pouvez utiliser :

Index	0	133
Nom	Bulbizarre	Aquali
CP	613	2729
HP	64	202
Dust	4000	5000
Candy	4	4
Attaque	126	186
Défense	126	168
Stamina	90	260
IV	56 %	100 %

Félicitation tu as gagné le badge suivant :



Metamocker