TD n°7 – Héritage, polymorphisme, collections
Le projet Pokemons

À rendre au plus tard le 25 juin minuit. Projet à envoyer à remi.watrigant@inria.fr avec comme objet "Projet POO"

N'hésitez pas à poser des questions en TP ou par mail entre temps…

Consignes

- **Ce projet se réalise sur plusieurs séances, par groupe de 2 étudiants.**
- **Le code doit être correctement commenté (pas trop, ni trop peu), robuste (gestion des exceptions), et associé à des TestCase JUnit.**
- **Un travail minimum est demandé lors de chaque séance de TD. La réalisation de ce travail est obligatoire et sa bonne réalisation assure la moyenne au groupe.**
- **Des extensions seront suggérées ou laissées au libre choix des étudiants. Toute extension correctement implémentée améliore la note du groupe.**
- **Au contraire, tout code similaire entre groupes diminue la note de tous les groupes concernés.**

Les Pokemons

➢ 1ère partie obligatoire

Les classes des Pokemons

Les Pokemons sont de gentilles créatures passionnées par la programmation objet en général et par le polymorphisme en particulier.

Nous commençons par créer une classe `Pokemon`. Un Pokemon se définit par (entre autres) :

- Un nom, une taille, et un poids ;
- Un attribut pv correspondant à ses points de vie ;
- Un attribut pc correspondant à sa puissance de combat ;
- Un type :

En Java, le type d'un Pokemon sera représenté par un **type énuméré** qui permet de définir et d'utiliser un ensemble fixe de constantes comme des variables objets. **Lire la documentation sur les types énumérés (http://docs.oracle.com/javase/tutorial/java/javaOO/enum.html) et créer un type énuméré pour les types des Pokemons décrits plus bas;**

- Une méthode `estVivant()` indiquant si un Pokemon est toujours en vie (pv > 0) ou non ;
- Une méthode `attaquer(Pokemon p)` permettant au Pokémon courant d'attaquer le Pokemon passé en paramètre. Cette méthode inflige des dégâts aux points de vie des Pokemons. **Elle sera implémentée plus spécifiquement plus tard,** et dépendra du type de chaque Pokemon.
- Une redéfinition de la méthode `toString()` héritée de la classe `Object` qui affiche les informations du Pokémon (choix d'affichage libre).

Les Pokemons décrits à travers la classe `Pokemon` sert de base à la modélisation de ces petites créatures. Le travail consiste maintenant à décrire les Pokemons en fonction de leur type. Il existe plus d'une quinzaine de types différents[1]. Nous nous concentrons ici sur quatre catégories :

1

1. Les Pokemons de type EAU : ils possèdent en plus un nombre de nageoires. Ces Pokemons ne se déplacent que dans la mer à une vitesse = (poids * nombre de nageoires) / 25 ;
2. Les Pokemons de type FEU : ils possèdent en plus un nombre de pattes. Ces Pokemons se déplacent sur la terre à une vitesse = nombre de pattes * poids * 0,03 ;
3. Les Pokemons de type PLANTE : ils se déplacent sur la terre très lentement à une vitesse = 10 / (poids * taille) ;
4. Les Pokemons de type ELECTRIK : ils possèdent un nombre de pattes, un nombre d'ailes, ainsi qu'une intensité. Au contraire des Pokemons de type PLANTE, ils se déplacent sur la terre ou dans les airs très vite à une vitesse = (nombre de pattes + nombre d'ailes) * intensité * 0,05.

Les Pokemons de type EAU sont super efficaces contre les Pokemons de type FEU et leur infligent deux fois plus de dégâts (2*pc). Par contre, ils sont très peu efficaces contre les Pokémon de type PLANTE ou de type EAU et ne leur infligent que la moitié des dégâts (0.5*pc). Ils infligent des dégâts normaux aux Pokémon de type ELECTRIK.

Les Pokemons de type FEU sont super efficaces contre les Pokemons de type PLANTE et leur infligent deux fois plus de dégâts (2*pc). Par contre, ils sont très peu efficaces contre les Pokemons de type EAU ou de type ELECTRIK et ne leur infligent que la moitié des dégâts (0.5*pc). Ils infligent des dégâts normaux aux Pokémon de type FEU.

Les Pokemons de type ELECTRIK sont super efficaces contre les Pokemons de type EAU et leur infligent deux fois plus de dégâts (2*pc). Par contre, ils sont très peu efficaces contre les Pokemons de type ELECTRIK ou de type PLANTE et ne leur infligent que la moitié des dégâts (0.5*pc). Ils infligent des dégâts normaux aux Pokémon de type FEU.

Enfin, les Pokemons de type PLANTE sont super efficaces contre les Pokemons de type ELECTRIK et leur infligent deux fois plus de dégâts (2*pc). Par contre, ils sont très peu efficaces contre les Pokémon de type PLANTE ou de type FEU et ne leur infligent que la moitié des dégâts (0.5*pc). Ils infligent des dégâts normaux aux Pokémon de type EAU.

- **Créez quatre classes `PokemonFeu`, `PokemonEau`, `PokemonPlante`, et `PokemonElectrik` qui héritent de la classe `Pokemon` et qui représentent les quatre types de Pokemons susmentionnés.**
- **Ajouter les constructeurs, accesseurs, ainsi que les méthodes `attaquer` et `calculerVitesse`. Factoriser la déclaration de la fonction `calculerVitesse`.**
- **Mettre à jour la méthode `toString()` affichant par exemple pour un Pokemon de type ELECTRIK :**

```
« Je suis le Pokemon Pikachu (type ELECTRIK). Mon poids est de 6 kg, ma vitesse est de 2,5 km/h. J'ai 2 pattes, ma taille est de 0,4m et mon intensité est de 50 mA. »
```

Ou encore :

```
« Je suis le Pokemon Carapuce (type EAU). Mon poids est de 12 kg, ma vitesse est de 1,9 km/h. J'ai 4 nageoires, ma taille est de 0,81m. »
```

Les tests unitaires

Définir des *TestCase* sur les Pokemons permettant de tester les différentes classes et les méthodes `estVivant`, `calculerVitesse` et `attaquer`.

---

http://www.pokepedia.fr/Catégorie:Pokémon_par_type

Les joueurs

➢ 2ème partie obligatoire

Collection de Pokemons

Nous allons maintenant considérer la classe `Joueur` qui contient (au moins) :

1. Un nom ;
2. Un niveau ;
3. Un nombre de points ;
4. Une collection de Pokemons ;

Quel est le type de collections à utiliser pour déclarer la collection du joueur ? (« Généraliser » le plus possible le type déclaré)

La méthode `vitesseMoyenne()` retournant la vitesse moyenne des Pokemons de la collection ;
Des méthodes `vitesseMoyenneTYPE(String t)` calculant la vitesse moyenne des Pokemons de type `t` de la collection.
N'oubliez pas de traiter les éventuels mauvais usages de la méthode en levant des exceptions !
La méthode `toString()` affichant l'état d'un joueur (affichage au choix).

En plus de ces fonctionnalités de base, un joueur peut :

- Attraper un Pokemon en l'ajoutant à sa collection ; Ceci est réalisé par la méthode `attraper`
- Transférer un Pokemon en l'enlevant de sa collection ; Ceci est réalisé par la méthode `transférer`

Combat de Pokemons entre Joueurs : méthode aléatoire

Enfin, un joueur peut attaquer avec l'un des Pokemons de sa collection un autre joueur. Le joueur attaqué choisit alors un Pokemon de sa collection qui combattra contre le Pokemon de l'attaquant.

Nous souhaitons définir une méthode `void attaquer(Joueur adversaire)`.

L'attaque la plus simple est de choisir aléatoirement les Pokemons qui se combattent dans les collections respectives des 2 joueurs.

La méthode `attaquer` de la classe `Joueur` doit donc :

- Choisir aléatoirement un Pokemon dans la collection de chacun des joueurs ;
- Appeler la méthode `attaquer` de la classe `Pokemon` afin de modifier les points de vie des Pokemons qui combattent.

Les tests unitaires

Ne pas oublier de tester la classe `Joueur` ! Et agrémenter les tests des Pokemons avec les nouvelles fonctions développées.

Les parties

➢ 3ème partie obligatoire

Le fichier des Pokemons

Un fichier texte nommé *ListePokemon.txt* est disponible en annexe de ce sujet. Il contient sur chaque ligne la description d'un Pokemon :

nom type taille poids pv pc

En fonction du type, les paramètres additionnels sont ajoutés à la suite de la puissance de combat sur la ligne. Les données sont séparées entre elles per un espace.

Dans une classe `Partie`, définir une méthode qui va se charger d'initialiser une liste de Pokemons instanciés à partir du fichier texte, en objets référençant le bon type avec tous ses attributs bien remplis.

Jeu à 2 joueurs

Modifier la classe `Partie` afin de lancer une partie « aléatoire » entre 2 joueurs dans laquelle :

5. Chaque joueur démarre avec 10 Pokemons dans sa collection. Ces Pokemons sont choisis de manière aléatoire dans la liste initiale des Pokemons enregistrée à partir du fichier texte.
6. Les deux joueurs s'attaquent séquentiellement (peu importe qui attaque qui) et choisissent aléatoirement un Pokemon dans leur collection respective pour combattre.
7. La partie s'arrête alors lorsqu'un joueur n'a plus de Pokemon vivant dans sa collection.
8. **Afficher le résultat de la partie.**

Extensions possibles

Libre à vous d'imaginer des extensions à votre programme. Par exemple :

- générer une Javadoc
- utiliser les points et niveaux des joueurs
- permettre à un Pokemon d'évoluer
- sauvegarder une partie
- …

Dans ce cas, merci d'expliquer vos extensions dans un fichier annexe au projet…