Développement orientée objet - Java
TD-TP 05

Lionnel Conoir, Isabelle Delignières, Wajdi Elleuch, Rémi Synave

Lors de ce TD, vous allez apprendre :
— À écrire vos propres classes.
— À écrire toutes les méthodes nécessaires.
— À écrire un programme principal utilisant vos propres classes.

Le contexte

Les vacances... Quel enfer ! On s'ennuie. En plus, les serveurs de Pokemon GO sont tombés. Impossible de se connecter. Le parc Saint Pierre de Calais s'est vidé. Et pourtant, vous avez une furieuse envie de faire quelques combats. Qu'à cela ne tienne, vous allez coder vos propres combats Pokemon.
Let's go ! ! ! !

Analyse et modélisation du problème

Vous allez devoir créer et manipuler des pokemons. Avant de commencer à programmer, vous devez analyser le problème et recenser toutes les caractéristiques de ces monstres dont vous allez avoir besoin.
Avant de décrire les pokemons, il faut savoir que la liste des pokemons avec toutes leurs caractéristiques chiffrées sont réunis dans une liste appelée pokedex. Le pokedex est fourni à titre d'information sous forme d'un fichier CSV contenant les informations nécessaires pour la création des pokemons pour ce projet (pas d'info sur les évolutions par exemple).

Les pokemons

Un pokemon, c'est un monstre qui possède :
— Une race, une espèce : comme dit précédemment, l'ensemble des espèces de pokemon est défini dans le pokedex qui associe un numero à chaque espèce. Cet ensemble est fini et nous ne nous intéresserons ici qu'à la première génération, c'est à dire aux pokemons ayant pour numéro 1 à 151. Dans ce projet, nous stockerons le numéro du pokemon du pokedex.

— Un nom : il est possible de donner un nom aux pokemons. Habituellement, le nom d'un pokemon est le nom de son espèce.
— Un ou deux types : il existe 15 types différents dont `normal`, `feu`, `eau`, `plante`, etc. Un pokemon possède au minimum un type mais peut en avoir deux. La liste des types peut être retrouvée sur l'image contenant le tableau d'efficacité de différents types au combat. Cette particularité sera décrite plus tard dans le sujet.
— Des caractéristiques de vie, attaque, défense et vitesse représentées par des nombres. On appellera `PV` le nombre de points de vie du pokémon, `att` son nombre de points d'attaque, `def` son nombre de points de défense et `vit` sa vitesse. Évidemment, plus le nombre est élevé, meilleure est la caractéristique. Ces nombres seront utiles lors des combats. Les pokemons se combattant verront leurs points de vie diminuer lors des attaques et les dégâts infligés seront fonction des caractéristiques d'attaque et de défense.

Ci-dessous deux exemples de pokémon :

Numéro : 57
Espèce : Colossinge
Nom : Robert
Type 1 : Combat
Type 2 : ∅
PV : 65
Att : 105
Def : 60
Vit : 95

Numéro : 94
Espèce : Ectoplasma
Nom : Gustave
Type 1 : Spectre
Type 2 : Poison
PV : 60
Att : 65
Def : 60
Vit : 110

Les combats de pokemon

Dans un premier temps, les combats seront gérés de manière assez simpliste. Lorsqu'un pokemon attaque un autre pokemon, il va lui retirer des PV. Le nombre de PV retiré (les dégâts) va suivre la formule suivante :
dégats = Att du pokemon qui attaque - défense du pokemon attaqué.
Si la valeur est inférieure à 0 alors on mettra la valeur à 0.
De plus, lorsqu'un combat est lancé, c'est le pokemon ayant la statistique de vitesse la plus élevée qui attaque en premier. Toutefois, il faut savoir que les types des pokemons influent également sur la force des attaques par rapport à celui qui défend. Cette information pourra être prise en compte lors des affrontements.
Par exemple, un pokemon de type eau sera plus efficace contre un pokemon de type feu que contre un pokémon de type plante. Chaque type possède une efficacité contre tous les autres types. Cette information peut être retrouvée sur l'image fournie avec ce projet et dans la classe `Type`. Il n'y a pas de consigne donnée pour son utilisation.

La classe `Type`

La classe `Type` sera une classe outil permettant de retrouver facilement des informations à propos des pokemons et types. Elle regroupe un ensemble de données et méthodes statiques pour la gestion des noms de pokemons, des types et de leur efficacité lors des combats. Cette classe est donnée en partie. Vous devrez vous y conformer et respecter l'existant.
Cette classe contient :

- un tableau constant de chaines de caractères contenant le nom de tous les pokemons dans l'ordre du pokedex,
- des constantes pour chaque type,
- un tableau constant de chaînes de caractères contenant les noms des types (leur indice correspond aux constantes des types),
- un tableau constant à double entrée contenant les efficacités de type ([typeAtt][typeDef]),
- des méthodes statiques à compléter.

La classe `Pokemon`

En accord avec la description ci-dessus, la classe Pokemon sera composée de :

- un attribut entier contenant le numero du pokemon dans le pokedex,
- un attribut chaine de caractère contenant le petit nom du pokemon,
- deux attributs dont le type est contraint (à vous de deviner) pour les deux types du pokemon,
- quatre attributs entier pour les PV, l'attaque, la défense et la vitesse du pokemon.

Le nom de l'espèce du pokemon n'a pas besoin d'être stocké. En effet, avec le numéro du pokemon dans le pokedex et la classe `Type`, il est possible de retrouver le nom de l'espèce du pokemon.

Comme il a été vu en cours, ces attributs doivent être déclarés **privés**. Cette classe doit donc également contenir toutes les méthodes permettant l'accès en lecture et écriture aux attributs (seulement pour les attributs pour lesquels c'est nécessaire).

La classe `Pokemon` devra également être dotée de constructeurs permettant la création d'un pokemon :

— un constructeur par défaut qui créera un pokemon que vous choisissez (missingno ?),
— un constructeur prenant le numéro du pokemon dans le pokedex, un nom ainsi que les deux types et les quatres statistiques de PV, attaque, défense et vitesse.

La classe devra également contenir les méthodes `equals` et `toString`.

Finalement, il faut créer une méthode `attaque` prenant un objet de type `Pokemon` en paramètre. Cette méthode devra lancer un tour d'attaque : le pokemon le plus rapide attaque en premier puis le moins rapide contre attaque s'il est encore vivant.

TODO

Le moteur du jeu

Terminez le développement de la classe `Type` puis développez la classe `Pokemon` en suivant les instructions ci-dessus.

La classe `Main`

Écrivez ensuite une classe `Main` contenant un programme principal dans lequel est créé deux pokemons et les faire combattre jusqu'à ~~la mort de l'un deux~~ ce que l'un d'entre eux retourne dans sa pokeball.

Simplification d'instanciation

Sans programmer, proposez une stratégie de simplification de création des pokemons. Actuellement, lorsque vous créez N fois le même pokemon, vous devez rechercher ses statistiques. Proposez une méthode permettant de simplifier la création. Attention, cette simplification ne doit pas s'appuyer sur le mécanisme d'héritage.

Si vous avez tout terminé

Vous pouvez prendre en compte les types pour les combats. A vous de voir comment inclure l'efficacité dans le combat.

— En mulitipliant les dégâts par l'efficacité ?
— En multipliant la statistique d'attaque par l'efficacité ?
— Comment gérer les doubles types ?
— ...