Cahier des charges

Module d’entraînement des Pokémons

Description

Votre entreprise achète et vend des pokémon.

Vous devez développer un module pour entraîner ces pokémons, utilisant un autre module qui est le référentiel des pokémons.

Deux « modules » sont à développer :

- le référentiel des pokémons « bouchonné »,
- le module d'entraînement.

Le référentiel de pokémons

Dans la version finale, nous nous brancherons au référentiel de pokémons officiel de l'entreprise. Pour des raisons de prototypage, vous devrez réaliser un référentiel de pokémons qui fera office de « bouchon ». Le référentiel de pokémons doit contenir des pokémons. Il doit être initialisé avec quelques pokémons (au moins 6) de différents types. Ce référentiel, offre, via une API Java les fonctionnalités suivantes :

- Récupérer tous les pokémons
- Récupérer un pokémon par son identifiant
- Mettre à jour un pokémon

Pour des raisons de simplicité, ce référentiel n'est pas sauvegardé de façon pérenne. Il est réinitialisé à chaque démarrage de l'application : ses données restent dans la mémoire de la JVM.

Le module d'entraînement

Le module d'entraînement est dépendant du référentiel de pokémons. Il offre, via la console, les fonctionnalités suivantes :

- Afficher dans la console une liste de tous les identifiants et noms d'un pokémon
- Afficher dans la console une liste triée de tous les pokémons par expérience (croissante ou décroissante)
- Afficher dans la console le détail d'un pokémon (choisi par son id)
- Afficher dans le console une liste de toutes les arènes
- Faire combattre via la console deux pokémons (choisis par leur id) dans une arène (choisie par son nom), ceci mettra à jour le pokémon vainqueur dans le référentiel

Description des entités

- Toutes les entités ont un identifiant numérique unique et non nul pour son type.
- Un pokémon appartient forcèment à une espèce (ex : bulbizarre). Il a un nom, un nombre de points de vie et un nombre de points d'expérience.
- Une espèce de pokémon détermine son type, le nombre de points de vie maximum et l'attaque du pokemon. Modéliser au moins les espèces du tableau des annexes.
- L'attaque d'un pokémon a un nom et un type. Elle inflige des dégâts différents selon le type de l'attaque et le type du pokémon attaqué (Cf. tableau infra).
- Une arène a un nom et deux effets :

    - l’effet unique s’applique une seule fois par combat, au début de celui-ci, aux deux pokémons qui combattent dans l’arène.
    - l’effet permanent s’applique à chaque tour de jeu, aux deux pokémons qui combattent dans l’arène.
- Un combat oppose deux pokémons. Il consiste en plusieurs phases :
    1. chaque pokémon a ses points de vie initialisés à la valeur de ses points de vie maximum.
    2. l’arène applique son effet unique par combat (optionnel)
    3. un pokémon est choisi au hasard, il sera désigné sous le nom de pokémon1, l’autre sera désigné sous le nom de pokémon2
    4. début du tour de jeu
    5. l’arène applique son effet qui s’applique à chaque tour de jeu(optionnel)
    6. le pokémon1 attaque, l’effet de l’attaque s’applique.
    7. si le pokémon2 est KO (plus de point de vie), le match est terminé : le pokémon1 a gagné.
    8. le pokémon2 attaque, l’effet de l’attaque s’applique.
    9. si le pokémon1 est KO (plus de point de vie), le match est terminé : le pokémon2 a gagné.
    10. Si aucun pokémon n’est KO, on retourne à l’étape 3.
- A la fin du match, le pokémon vainqueur remporte des points d’expérience : (nombre de points d’expérience du pokémon vaincu) / 3 . Le minimum d’expérience acquis est de 100 points d’expérience. Il est sauvegardé dans le référentiel.

Desiderata techniques

- Pour modéliser les malus et bonus des pokémons selon leur type, utiliser le polymorphisme.
- Pour modéliser les effets de l’arène selon leur type, utiliser le polymorphisme.
- Le référentiel de pokémons ne propose qu’une méthode par fonctionnalité, pour tous les pokémons.
- Les deux modules doivent faire partie de packages Java différents.

Livrables attendus

- Le code source en Java, exécutable. L’exécutable interprète des entrées au clavier dans la console et affiche des messages dans la console.
- Le diagramme UML de cas d’utilisations.
- Le diagramme UML des différentes classes utilisées.
- D’autres diagrammes UMLs si nécessaire.
- Une présentation orale du projet, avec un support écrit, comprenant :
    - le plan initial
    - la description des diagrammes
    - comment le code a été produit, les problèmes rencontrés, et les solutions trouvées.
    - des idées d’évolution ou d’amélioration.

Fonctionnalité optionnelles, mais considérées comme « bonus » si développées

L’ordre ci-dessous n’est pas à suivre obligatoirement :

- Ajouter plus d’attaques
- Créer un nouveau type de pokémon et un nouveau type d’attaque
- Gérer les niveaux des pokémons. Selon le tableau des points d’expérience, un pokémon a un certain niveau. Ce niveau détermine un bonus à ses points de vie maximum, et un bonus à l’attaque.
- Créer un modèle physique de données du référentiel de pokémons. Créer ensuite des requêtes de DDL et DML pour créer les tables et les requêtes nécessaires à la création et la mise à jour de la base de données.
- Permettre à un pokémon d’avoir plusieurs attaques

Annexes

Tableau 1: Espèces de pokémons de base

Espèce	Type	Points de vie maximum	Attaque
Carapuce	Eau	100	plouf
Salamèche	Feu	100	flamèche
Roucoul	Air	90	battement
Racaillou	Sol	110	écrasement

Tableau 2: Attaques

Nom attaque	Type	Dégats
Plouf	eau	20
Flamèche	feu	20
Battement	air	23
Écrasement	sol	18

Tableau 3: Bonus d'attaque selon le type d'attaque et le type de pokémon(en pourcents)

type d’attaque\ type de pokemon	espèce type eau	espèce type air	espèce type sol	espèce type feu
attaque eau	100	75	100	125
attaque air	125	100	75	100
attaque sol	100	125	100	75
attaque feu	75	100	125	100

Tableau 4: Exemples d’arènes

Nom d’arène	Effet unique	Effet permanent
Prairie	Aucun	Aucun
Volcan	Perte de 20 points de vie	Aucun
Mare acide	Aucune	Perte de 5 points de vie

Objectifs pédagogiques

- **Imaginer une application client/serveur ou consommateur/référentiel**. Le référentiel pourrait être Salesforce Marketing Cloud, appelé via le SDK Fuel (FuelSDK). Les appels REST sont ici remplacés par des appels directs à des méthodes Java, mais le principe est le même. Le consommateur utilise les données du référentiel (ici les pokémons) et ses données propres (ici les arènes) pour appliquer ses algorithmes, et au final, mettre à jour le référentiel (en l'occurence, avec l'expérience des pokémons).
- **Utiliser les principes de programmation orientée objet** pour :
  - encapsuler les données correctement, et rendre le code lisible et maintenable (voire évolutif).
  - communiquer avec une API qui utilise des classes génériques, dont héritent des classes spécifiques.
  - créer des algorithmes qui utilisent le polymorphisme.
  - voire trouver les limites de la programmation orientée objet.
- **Utiliser les notions vues en Java pour coder des algorithmes** répondant à une spécification.