M2103-- TP2

Dans ce TP nous continuerons à travailler sur la relation entre un pokemon et un joueur. Nous allons implémenter plusieurs constructeurs, nous permettrons aux joueurs de nommer leurs pokemons et de capturer et libérer des pokemons qu'ils possèdent. Pour un aspect plus social, nous donnerons également la possibilité à deux joueurs de s'échanger des pokemons.

Lien utile : https://pokemondb.net/pokedex/all .

Préparation avant de commencer le TP

Attention : c'est votre obligation à vous présenter à chaque TP avec un programme fonctionnel. Si vous finissez par utiliser la correction fournie, OK, mais elle doit avoir été testée sur la machine sur laquelle vous travaillez et elle doit fonctionner !

Nous allons continuer de programmer le jeu aux pokemons. Nous avons un choix : veut-on modifier le projet Java commencé lors du TP précédent ou préférons-nous commencer un nouveau projet ?

Pour ce module nous allons sauvegarder le travail de chaque TP dans un nouveau projet.

Attention : c'est dans votre intérêt de sauvegarder le travail dans un nouveau répertoire à chaque semaine. Au cas où vous oubliez envoyer les sources d'un TP noté vous aurez parfois la possibilité de l'envoyer plus tard (avec un malus pour le délai), pour quand même avoir une note autre que 0.

Pour ce faire, ouvrez Eclipse et créez un nouveau projet Java appelé TP2. Dedans, créez un package tp2.

Dans votre package explorer (à gauche dans la vue standard d'Eclipse) détaillez votre TP1 et sélectionnez les deux classes de ce TP. En Windows faites CTRL + C (ou en utilisant click gauche copiez les deux fichiers). Puis mettez vous sur le package tp2 du projet TP2 et collez les deux fichiers.

Nous allons désormais travailler exclusivement sur le code du TP2.

Dans la classe ChasseAuxPokemons enlevez tout le code sauf la déclaration des trois pokemons et les trois lignes de System.out.println des trois pokemons.

Exercice I

Dans cet exercice nous allons créer des joueurs, qui auront des pokemons.

1. Créez une nouvelle classe `Joueur`. Les attributs caractérisant les objets de cette classe seront :
   - un nom de type `String`
   - un prénom de type `String`
   - un attribut `age` de type `int`
   - un tableau `pokemons` contenant leurs pokemons. La taille de ce tableau sera limitée à 5.

2. Ajoutez à cette classe un constructeur avec la signature :

   ```
   Joueur (String, String, int, Pokemon[])
   ```

   Ce constructeur assigne à chaque attribut la valeur correspondante mise en paramètre.

3. Ecrivez un deuxième constructeur qui aura la signature :

   ```
   Joueur (String, String, int)
   ```

   Ce deuxième constructeur instancie les trois premiers attributs aux valeurs données en paramètre et instancie les cinq éléments du tableau de pokemons à `null`.

4. Ecrivez des getters pour les attributs de cette classe.

Exercice II

Dans cet exercice nous allons ajouter quelques attributs dans la classe `Pokemon`. Désormais, un pokemon aura un nombre d'attributs en plus de ceux déjà existants. Nous donnerons à un pokemon la possibilité d'avoir un joueur et de porter un nom choisi par son joueur.

1. Dans la classe `Pokemon` ajoutez un attribut `nomDonne` de type `String` et un attribut `monJoueur` du type `Joueur`.
2. Ajoutez un deuxième constructeur à la classe `Pokemon` avec la signature : `Pokemon(String, String, int, boolean, String, Joueur)` , qui prendra en entrée, en ordre, les valeurs des attributs : `nom, type, niveau, diurne, nomDonne, monJoueur`.
3. Chaque pokemon est né en liberté, alors initialement les attributs `nomDonne` et `monJoueur` ne seront pas initialisés. Si on veut initialiser un attribut, sans lui donner une valeur, nous pouvons lui attribuer la valeur `null` (par exemple `nomDonne = null;`). Ainsi, l'object fera référence à une location vide.

Nous voulons adapter le code du constructeur de la classe Pokemon (avec la signature `Pokemon(String, String, int, boolean)`) pour initialiser les attributs nomDonne et `monJoueur` à `null`. Cependant, simplement ajouter ces deux lignes de code dans le constructeur serait du code dupliqué. Pour éviter cette déduplication nous allons plutôt appeler le constructeur avec plus de paramètres (réalisé dans l'exercice 1) dans celui avec moins de paramètres.

4. Modifiez la méthode `toString()` de la classe Pokemon pour y ajouter les valeurs des attributs nomDonne et `monJoueur` dans le même format que les autres attributs.

5. Ecrivez une méthode `void sePresenter()` qui fait afficher un texte sur l'écran. Ce texte sera :
   `Voici un pokemon <nom du pokemon> de niveau <niveau>.`
   Si le pokemon a un maître, alors on rajoute le texte suivant :
   `Il appartient à <nom du monJoueur>.`
   Si de plus le pokemon a un nom donné par le joueur, alors on rajoute le texte :
   `Il s'appelle <nomDonne>.`

6. Nous voulons pouvoir accéder aux attributs nomDonne et `monJoueur` dans la classe Pokemon pour les modifier au besoin (un pokemon peut avoir un maître, puis en changer). Pour cela il faut avoir une méthode qui retourne la valeur stockée en ce moment par chacune de ces variables, et une méthode qui change la valeur de chacune de ces variables. Créez les méthodes suivantes :
   - `String getNomDonne()` – qui retourne la valeur actuellement stockée par nomDonne
   - `Joueur getMonJoueur()` – qui retourne la valeur actuellement stockée par `monJoueur`
   - `void setNomDonne(String)` – qui remplace la valeur actuellement stockée dans nomDonne par la valeur mise en entrée
   - `void setMonJoueur(Joueur)` – qui remplace la valeur actuellement stockée par `monJoueur` par le joueur donné en entrée.

7. Dans la méthode principale (`main`) de la classe `ChasseAuxPokemons`, regardez les instructions que vous avez utilisées pour créer vos trois pokemons (Piplup, Rowlet et Totodile). Est-ce que ces instructions vont compiler ? Pourquoi (ou pourquoi pas) ? Quel sera l'affichage des instructions que vous avez ensuite utilisées pour « afficher » ces pokemons ?

Exercice III

Finalement dans cet exercice nous allons donner la possibilité aux joueurs de capturer et libérer un pokemon, ainsi que de lui donner un nom. Nous allons travailler donc dans la classe `Joueur`.

1. Dans le constructeur avec la signature `Joueur(String, String, int, Pokemon[])` nous avons assigné a `this.pokemons` la valeur en entrée de type `Pokemon[]`. Nous allons s'assurer

que chacun de ces pokemons aura la valeur de `monJoueur` mise au joueur actuel (`this`). Modifiez donc le constructeur tel que cela soit réalisé.

2. Ecrivez une méthode privée avec la signature `int trouverPokemon(Pokemon pokemon)` qui aura le fonctionnement suivant : nous allons chercher dans le tableau `this.pokemons` jusqu'à trouver le pokemon en paramètre de la méthode. Si on trouve ce pokemon, la méthode retourne sa position dans le tableau. Si on ne trouve pas ce pokemon, on retourne la valeur -1 (qui fonctionne comme une valeur d'erreur, indiquant que le pokemon n'existe pas dans le tableau).

3. Comment peut-on utiliser la méthode `trouverPokemon` pour chercher si on a une place disponible dans le tableau ?

4. Ecrivez une méthode avec la signature `void capturer(Pokemon)` qui se comportera comme indiqué ci-dessous : si le pokemon en paramètre n'a pas de maître et si le joueur a encore de la place dans son tableau de pokemons, alors on rajoute le pokemon dans le tableau sur une place disponible. Le joueur deviendra le maitre de ce pokemon (le joueur deviendra la valeur de `monJoueur` du pokemon en question). Si le joueur a déjà cinq pokemons la méthode retournera un message d'erreur qui demandera à un joueur de renoncer à un de ses pokemons.

5. Ecrivez une méthode avec la signature `void liberer(Pokemon)`, qui permettra à un joueur de libérer un de ses pokemons. Un pokemon libéré sera enlevé du tableau de pokemons de son ancien maître et il aura les attributs `nomDonne` et `monJoueur` remis à `null`. Ce pokemon redeviendra capturable.

6. Un joueur pourra donner des noms à ses pokemons. Ceci résultera dans une méthode void nommer(Pokemon, String) qui changera l'attribut du pokemon en entrée au String en entrée.

Un petit extra

Si vous avez déjà fini l'exercice précédent, utilisez vos méthodes pour écrire du code dans la classe Joueur qui vous permettrait de faire les actions suivantes :

- **Donner ses pokemons** : un joueur pourrait vouloir donner un de ses pokemons à un autre joueur (sans faire un échange). Écrivez une méthode qui permet cela, sans oublier de faire les vérifications nécéssaires sur le joueur qui reçoit le pokemon (sa collection ne doit pas excéder cinq pokémons) et sans oublier de mettre à jour les informations du pokemon donné.

- **Échanger des pokemons** : deux joueurs peuvent échanger des pokemons. Écrivez une méthode qui effectue l'échange des pokemons entre deux joueurs.