TP 2 : Les joueurs aux pokemons !

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 .

1. Dans notre projet ChasseAuxPokemons, nous allons travailler sur les trois classes que nous avons créées au sein de notre dernier TP. Premièrement, nous allons ouvrir la classe Pokemon.

   - Ajoutez un attribut nomDonne du type String et un attribut monJoueur du type Joueur.
   - Ajoutez un constructeur dans la classe Pokemon avec la signature : Pokemon(String, String, int, boolean, String, Joueur) , qui prendra en entrée, en ordre, les attributs : **nom**, **type, niveau, diurne, nomDonne, monJoueur**.
   - 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.
     Adaptez ainsi l'ancien constructeur de la classe Pokemon (avec la signature Pokemon(String, String, int, boolean)) en initialisant dedans les attributs nomDonne et monJoueur.
   - Modifiez la méthode toString() de la classe Pokemon. Le texte qui sera affiché sur l'écran sera désormais :
     - Si le pokemon a un maître, et si nomDonne n'est pas mis à null alors le texte affiché sera : <nomDonne du pokemon> est un pokemon de genre <nom du pokemon>, du type <type du pokemon>, qui a le niveau <niveau>. Ce pokemon appartient a <le nom et prenom du joueur monJoueur>.
     - Si le pokemon a un maître, mais nomDonne est mis à null alors le texte affiché sera : Voici un pokemon du genre <nom du pokemon>, du type <type du pokemon>, qui a le niveau <niveau>. Ce pokemon appartient a < le nom et prenom du joueur monJoueur >.
     - Si le pokemon n'a pas de maître, alors le texte affiché sera : Voici un pokemon du genre <nom du pokemon>, du type <type du pokemon>, qui a le niveau <niveau>. Ce pokemon n'a pas encore de maitre.

   - 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 ?

- Nous voulons pouvoir accéder aux attributs nomDonne et monJoueur dans la classe Pokemons 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
  - String 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.

2. Maintenant passerons à la classe Joueur. Nous allons créer les interactions suivantes entre les joueurs et les pokemons :
   - **Capturer un pokemon** : pour l'instant ceci sera juste une méthode, qui permettra d'ajouter un pokemon au tableau de pokemons d'un joueur (si ce dernier a moins de cinq pokemons à ce moment-là). Si le joueur a déjà cinq pokemons, il y aura un message d'erreur qui demandera au joueur de renoncer à un de ses pokemons.

     La signature de la méthode qui capture des pokemons sera : void capturer (Pokemon)
     Attention : il n'est pas possible de capturer un pokemon qui appartient à un autre joueur.

     De plus, lorsqu'un pokemon libre est capturé par un joueur, ce dernier devient son maître, et donc l'attribut monJoueur est mis à jour de façon correspondante (voir la méthode setMonJoueur).

   - **Libérer un pokemon** : cette méthode-ci permettra à un joueur de libérer un pokemon qui est déjà présent dans son tableau d'au plus 5 pokemons. La signature de cette méthode sera : void liberer (Pokemon). Un pokemon libéré aura son nomDonne et son monJoueur remis à null. Il redeviendra capturable. Attention : il faut bien prendre en compte que, lorsque le joueur capture un autre pokemon pour remplacer un qu'il a libéré, il faut trouver la ou les position(s) libre(s) du tableau de pokemons.

   - **Nommer un pokemon** : un joueur peut donner des noms aux pokemons dans sa collection. Ceci changera l'attribut nomDonne de ce pokemon. Quelle sera la signature de cette méthode (qui fera partie de la classe Joueur) ?

   - **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.

3. En utilisant votre classe principale, testez le fonctionnement de vos nouvelles méthodes.

4. La dernière partie de ce TP sera libre. Imaginez-vous des extensions à vos travaux. Par exemple, vous pourriez décider que les pokemons ne peuvent être libérés que lorsqu'ils sont révéillés (ou au contraire, endormis). Ou, vous pouvez décider qu'un joueur ne peut collectionner que un certain nombre de pokemons d'un certain type. Explorez comment programmer les modifications que vous voulez apporter à votre programme et testez-les !