TP 6 : Le combat des pokemons

Dans ce TP nous allons finalement réaliser le combat entre les pokemons. Dans une première étape, nous allons utiliser l'héritage pour définir quelques types d'attaques. Puis, nous allons permettre aux joueurs de contrôler une bataille entre un de leurs pokemons et un autre pokemon.

Liens utiles : https://pokemondb.net/pokedex/all .

https://pokemondb.net/move

1. Notre deuxième exemple d'héritage concernera les attaques. Nous allons avoir une superclasse Attaque qui pourra ensuite être personnalisée avec des différentes caractéristiques. Ainsi, chaque attaque deviendra sa propre classe.
   Regardez le Pokedex, notamment les attaques en même temps pendant cet exercice pour bien comprendre les attributs et leurs rôles.
   Une attaque a quelques attributs :
   - Son nom ;
   - Les types de pokemons ("PLANTE", "EAU", etc.) qui pourront utiliser cette attaque ;
   - Le type d'action (physique, spéciale) -- ici on ignore le fait qu'il y a également des attaques de statut. Nous allons supposer que les attaques physiques seront disponible à tout pokemon. Les attaques spéciaux affectent des différents types : FEU, EAU, ELECTRIQUE, PLANTE, GLACE, DRAGON, FONCE, TELEPATIQUE ;
   - La puissance d'une attaque (Power dans le Pokedex), qui indiquera combien de points HP le pokemon perdra suite à cette attaque ;
   - La précision (Acc dans le Pokedex) d'une attaque, qui indiquera la fréquence avec laquelle l'attaque marche.
   - La répétition (PP dans le Pokedex) d'une attaque, qui indiquera combien de fois cette attaque pourra être utilisée dans une seule bataille ;
   - Un attribut qui stocke le nombre de répétitions encore disponibles pour cette attaque.

Définissez la classe abstraite Attaque avec les paramètres : nom, types de pokemons compatibles, puissance, précision, répétition, et répétitions restantes. Attention : le type d'action n'est pas un attribut (on verra plus tard comment on va le traiter).

Pour cette classe, réalisez :

- Deux constructeurs. Comme particularité, les deux constructeurs mettent le nombre de répétitions restantes de l'attaque au nombre de répétitions totales possibles. Le premier constructeur aura la signature Attaque(String, String[], int, int, int), où chacun des paramètres en entrée correspond à un des attributs ci-dessus (sauf le dernier, voire la

remarque précédente), dans l'ordre dans laquelle sont listés. Le deuxième constructeur sera appelé lorsqu'une attaque est compatible avec tout type de pokemon ; il aura la signature Attaque(String, int, int, int).

- Des méthodes concrètes qui rendent la valeur de chaque attribut (méthodes get);
- Une méthode concrète void resetRepetitions() qui remettra le nombre de répétitions restantes au nombre maximal de répétitions permises ;
- Une méthode void baisseRepetitions() qui baisse par 1 le nombre de répétitions restantes de l'attaque ;
- Une méthode toString(), qui affichera <nom d'attaque> : <typeAttaque>, <puissance>, <precision>, <repetition>, <repetition restante>, compatible avec <compatibilites>. Si une attaque est compatible avec tout type de Pokemons, alors on remplace <compatibilites> par tous types de Pokemon. Sinon, elles seront affichés comme une liste (sur une seule ligne, avec des éléments précédés par le mot type et séparés par des virgules) ;
- Une méthode abstraite void utiliser(Pokemon attaquant, Pokemon victime) ; Rappel : nous sommes maintenant dans la classe Attaque. Cette méthode est utilisée donc lorsque le pokemon attaquant veut utiliser l'attaque (this) contre le pokemon victime ;
- Une méthode abstraite boolean estCompatible(Pokemon p) .

2. Même si on va devoir simplifier beaucoup des détails du combat des pokemons, nous allons pourtant avoir besoin de quelques attributs de plus pour chaque pokemon. Regardez par exemple Piplup : https://pokemondb.net/pokedex/piplup . Nous allons avoir besoin des scores pour :
    - l'attaque et la défense d'un pokemon,
    - l'attaque spéciale et la défence spéciale d'un pokemon,
    - une fréquence, qu'indiquera la probabilité de rencontrer un pokemon de ce type,
    - les HP (hit points) de base d'un pokemon

    Pour chaque pokemon vous pouvez trouver ses scores de depart sur leur page dans le pokedex. Pourtant, pour l'instant nous n'allons créer que des pokemons avec 30 HP.
    N.B.: Nous aurons également besoin d'un système de rémise à niveau (qui augmentera la valeur de quelques attributs) mais vous allez programmer cela lors des prochains TP.

    - Ajoutez les attributs listés ci-dessus à la classe Pokemon ;
    - Ajoutez un tableau d'attaques sesAttaques à chaque pokemon ;
    - Ajoutez une méthode void addAttaque(Attaque a) à cette classe. Si l'attaque est compatible avec le pokemon et si le pokemon a encore de la place pour cette attaque, on rajoute l'attaque sur la première position libre ; les positions non-occupés du tableau contiendront des éléments null : pour l'utilisation d'une attaque il sera important de vérifier que l'attaque est non-null.
    - Remettez à jour les constructeurs et la méthode toString() de la classe Pokemon. Attention : la taille de sesAttaques est de 4. Les deux constructeurs prendront en entrée

un tableau d'attaques, mais on n'a pas de garantie par rapport à sa taille ou ses contenus. Utilisez la méthode addAttaque pour rajouter les attaques une par une ;

- Ajoutez une méthode void removeAttaque(int index), qui doit vérifier si l'index est entre 0 et la taille de sesAttaques-1 et mettre à null l'élément de sesAttaques à la position index.
- Ecrivez une méthode void regarderAttaques() qui fera afficher les attaques existantes dans sesAttaques, avec leur position dans le tableau et le nombre de répétitions restantes. Si l'attaque est null, alors on ne l'affiche pas du tout.
- Ajoutez une méthode void resetAttaques() qui remettra les répétitions restantes de toute attaque dans sesAttaques à sa valeur maximale (quelle méthode de la classe Attaque pouvez-vous appeler ?)
- Réalisez une méthode boolean sEstEvanoui() qui retourne true si le pokemon n'a que 0 HP
- Ajoutez une méthode void estBlesse(int domage) qui baisse les HP d'un pokemon par la valeur domage (toutefois, les HP ne doivent jamais baisser en dessous de 0).
- Ecrivez une méthode utiliseAttaque(int index, Pokemon victime). Si le pokemon ne s'est pas évanoui, si l'index a une valeur entre 0 et sesAttaques.length et si l'attaque indiquée est valide (non-null), alors on appelle la méthode utiliser de la classe Attaque, avec le pokemon courant en tant qu'attaquant et le paramètre victime en tant que victime.

3. Les attaques physiques fonctionnent différemment que les attaques spéciales. Une attaque physique sera compatible avec tout type de pokemon. Au contraire, les attaques spéciales ne sont compatibles qu'avec un type de pokemon. Une attaque physique sera effectif (marchera) si : (i) l'attaque du pokemon attaquant est supérieur à la défense du pokemon victime ; et (ii) si un nombre généré aléatoirement entre 0 et 100 sera inférieur à la précision de l'attaque en question. Pour les attaques spéciales, dans la première condition l'attaque du pokemon sera remplacé par son attaque spéciale. Si le pokemon victime est compatible avec l'attaque, alors on compare ce score avec sa défense spéciale ; sinon, on le compare avec sa défense normale. N'oubliez pas de vérifier si vous avez assez de répétitions pour faire l'attaque marcher et de diminuer le nombre de répétitions par 1.

   Dans cet exercice nous allons créer deux classes concrètes, AttaquePhysique et AttaqueSpeciale qui héritent de la classe Attaque.

   - Ecrivez le constructeur de chacune de ces classes.
   - Les attaques physiques sont compatibles avec tout type de pokemon. Pour les attaques spéciales il faut regarder les types compatibles donnés en entrée du constructeur. Ecrivez des implémentations concrètes de la méthode boolean estCompatible(Pokemon p).
   - Ecrivez des implémentations concrètes de la méthode void utiliser(Pokemon attaquant, Pokemon victime) dans chacune de ces classes.

4. Nous allons créer 5 attaques physiques et 4 attaques spéciales. Chaque attaque aura sa propre classe et chaque classe ne contiendra qu'un constructeur sans paramètres en entrée.

- On va créer les prochaines attaques physiques : AttaqueMorsure(dans le Pokedex anglais, Bite), AttaqueCroquer (Crunch), AttaqueCoupDeTete (Headbutt), AttaqueFeinte(Feint), AttaqueTackle(Tackle). Trouvez à chaque fois les scores correspondants dans le tableau https://pokemondb.net/move/all .
- Nous allons également créer les attaques spéciales : AttaqueBulle (Bubble), AttaqueEnfer (Inferno), AttaquePistoleEau(Water Gun) et AttaqueTournadeFeuilles (Leaf Tornado).

5. Dans votre classe ChasseAuxPokemons vous allez premièrement commenter la plupart de la méthode main (minimalement vous allez commenter les modifications faites lors du TD5, ainsi que la plupart des tests faites lors des TD précédents). Votre méthode main vous donne normalement des erreurs. Pourquoi ?
    - Mettez à jour la déclaration de chacun de vos pokemons pour prendre en compte vos nouveaux attributs. Vous pouvez inclure n'importe quelles attaques vous voulez, de la liste de 9 attaques déjà créées.
    - Essayez de donner en entrée d'un de vos pokemons une attaque spéciale avec laquelle il n'est pas compatible. Qu'est-ce qui se passe ? Affichez les pokemons pour vérifier le fonctionnement de vos constructeurs ;
    - Vous allez maintenant écrire une bataille entre deux pokemons de votre choix. Au contraire de ce que va se passer dans une vraie bataille, vous allez contrôler tous les deux pokemons. Vous allez utiliser un objet de type Scanner pour vous donner la possibilité de choisir les attaques que vous voulez utiliser pour chaque pokemon. Vous pouvez utiliser quel pokemon commence. La bataille continuera jusqu'à ce qu'un des pokemons s'évanouisse. Le pokemon de votre choix commencera à chaque ronde et l'autre pokemon répondra. A chaque ronde vous allez afficher les attaques du pokemon qui va attaquer et vous demanderez l'index de l'attaque qu'on veut utiliser. Vous allez lire la réponse des pokemons en utilisant la méthode nextInt() de la classe Scanner. Attention : il faut vérifier que la valeur de l'Index est bonne. Puis, vous faites la même chose pour l'autre pokemon. Une fois qu'un pokemon s'évanouit, la bataille est finie et on reset les attaques de chaque pokemon.