Déroulement d'une partie

Cette feuille ne peut-être traitée que lorsque les classes Carte, Joueur et Jeu sont définies.

Les *questions 4 à 8 ne dépendent pas des questions 1 à 3. La question 9 est indépendante des autres. Et la 10 utilise un peu tout mais ne nécessite pas que tout soit terminé pour être commencée.*

Une partie est constituée de deux phase :

I. **Constitution de la pile de cartes du joueur à partir de la réserve**
II. **Bataille**

Vous allez coder les sous-programmes permettant de jouer une partie.

Vous avez normalement déjà une fonction membre de la classe Joueur qui prend un vecteur de cartes et affecte les 20 premières cartes à la pile du joueur.

1. Écrire une fonction interne de la classe Jeu qui fait jouer une carte de chaque paquet (une de chaque joueur) et met à jour les points de prestige de chaque joueur.
2. Écrire une fonction interne de la classe Jeu qui retourne vraie si la partie est finie et faux sinon.
3. Écrire une fonction interne de la classe Jeu qui retourne le vainqueur de la façon suivante : elle retourne +1 si le vainqueur est le `joueur1`, -1 si le vainqueur est le `joueur2` et 0 s'il y a match nul.
   Si la partie n'est pas finie, la valeur retournée n'a pas d'importance.

4. Écrire une fonction membre de la classe `Joueur` qui prend un vecteur de cartes, le mélange dans un ordre aléatoire et affecte le résultat du mélange à la pile.
5. Écrire un sous-programme qui, à partir d'un vecteur de carte, construit un vecteur d'entiers représentant le rang de chaque carte[1].
   Par exemple si dans la case `i` du vecteur de rang figure l'entier 3, cela signifie que la carte à l'emplacement `i` dans le paquet de carte a pour rang 3.
   Construire le vecteur de rang en fonction de l'attaque physique croissante : la carte avec la valeur d'attaque la plus élevée aura le rang le plus élevé et la carte avec la valeur d'attaque la plus faible a le rang le plus faible (c'est-à-dire 0).
   Le paquet de carte pourrait être rangé selon l'attaque décroissante, la défense croissante ou décroissante ou la magie croissante ou décroissante.
   Ne pas coder tous ces sous-programmes mais au moins 2 d'entre eux. **Si vous avez le temps**, vous pourrez toujours ajouter les autres.
6. En utilisant la fonction précédente, écrire une fonction membre de la classe `Joueur` qui prend en argument un vecteur de cartes et qui permet au joueur de remplir sa pile de 20 cartes en sélectionnant à chaque fois la carte (non encore sélectionnée) avec la plus forte attaque.
7. Écrire un sous-programme qui range un vecteur de cartes d'abord en fonction de l'attaque décroissante, en cas d'égalité sur la valeur d'attaque le rang se calculera en fonction de la puissance magique décroissante, et en cas d'égalité sur la magie cela sera en fonction de la

[1] Dans la suite, l'expression « ranger un vecteur de carte » fera référence à cette opération d'attribution d'un rang à chaque carte d'un paquet.

défense décroissante.
Si vous avez le temps, vous pouvez coder différentes variantes de ce sous-programme en échangeant les rôles de l'attaque, la défense et la magie ou en choisissant l'ordre croissant pour certaines valeurs.

8. On va ranger selon des critères plus complexes.
    a. Écrire une fonction membre de la classe Carte qui associe à chaque carte un score valant la somme de la valeur d'attaque, de la valeur de défense et de la puissance magique.
    b. Écrire un sous-programme qui range un vecteur de carte en sous-programme du score décroissant de chaque carte.
    c. **Si vous avez le temps,** proposer d'autres sous-programmes score et coder les.

9. Écrire une fonction membre de la classe `Joueur` qui laisse choisir au joueur l'ordre des cartes. Le joueur saisit le nom des cartes qu'il veut ajouter à son tas.
    d. À chaque saisie, le programme va chercher la carte dans la réserve et l'ajoute au paquet.
    e. Coder une variante de ce sous-programme pour le cas où la réserve de carte est triée par ordre alphabétique.

10. Écrire dans le `main()`, une boucle réalisant le jeu.
    Votre jeu devra implémenter un minimum des fonctionnalités suivantes :
    - ☐ Proposer entre un mode 2 joueurs humains et un mode humain contre IA
    - ☐ Demander le nom du/des joueur(s) humain(s)
    - ☐ Afficher les règles, la liste des cartes disponibles, …
    - ☐ Demander aux joueurs humains via un menu quelle stratégie ils veulent adopter pour constituer leur pile
    - ☐ Laisser le joueur constituer sa pile en choisissant les cartes une à une par leur nom
    - ☐ Exécuter la partie automatiquement et affiche les cartes ainsi que les dégâts infligés à chaque joueur
    - ☐ Afficher (et féliciter) le vainqueur
    - ☐ Proposer de rejouer

Cette liste n'est pas exhaustive, n'hésitez pas à rajouter des fonctionnalités qui vous semble pertinentes[2] si vous avez le temps.

[2] L'ajout de telles fonctionnalités ne doit pas se faire au détriment du reste du projet.