Mini projet Domino

Le but du projet est de compléter l'implémentation du jeu de dominos du TD 4 sur "Encapsulation et Polymorphisme". Dans la suite, on rappelle le problème et on précise quelques points.

On veut modéliser un jeu de domino. Ce jeu comporte 28 dominos et peut être joué par 2, 3 ou 4 joueurs. Au départ, on donne 7 dominos à chaque joueur. Les dominos restants forme un tas dans lequel pourront piocher les joueurs quand ils ne peuvent pas poser un domino de leur jeu. Pour simplifier, on va tirer au sort un ordre entre les joueurs. Le premier joueur pose initialement un domino de son choix. Pour poser un nouveau domino, il faut qu'une des extrémités ait le même nombre de points qu'un des côté du domino. En suivant l'ordre des joueurs, chaque joueur tente de poser un domino. Si un joueur ne peut pas et qu'il reste des dominos dans la pioche, il pioche un domino et passe son tour. Le premier joueur qui n'a plus de dominos gagne. Si aucun joueur ne peut déposer de domino, chaque joueur fait la somme des points des dominos qu'il possède et celui qui gagne est celui qui a la plus petite somme.

Pour implémenter ce jeu, on va implémenter une classe qui va gérer la partie (classe `Gestionnaire`) et des classes pour au moins deux types de joueurs : des joueurs automatiques (plus ou moins sophistiqués) et un joueur qui demandera de choisir le domino à poser à un utilisateur via la console.

le gestionnaire gère une partie :

- il distibue les dominos aux joueurs en début de partie
- il demande à chaque joueur de poser un domino quand c'est leur tour. Pour ce faire, il indique la valeur des deux extrémités de la suite des dominos posés.
- il donne un domino de la pioche si un joueur n'a pas pu poser un domino (on impose ce choix : ce n'est pas le joueur qui pioche s'il ne peut pas poser de domino, c'est le gestionnaire qui lui donne un domino en appelant une méthode du joueur).
- à chaque fois qu'un domino est posé par un joueur, il va avertir tous les autres joueurs que ce domino a été posé
- il arrête le jeu dès qu'il a detecté un vainqueur (les joueurs ne peuvent pas eux-mêmes se déclarer vainqueur).

Un joueur est un participant d'une partie. Il ne sait pas combien d'autres joueurs participent à la partie. Comme le gestionnaire le tient informé des dominos joués, il peut suivre le jeu (même s'il a une incertude sur l'agencement précis des dominos, il connait seulement ceux qui ont été posés). On va considérer que quand un joueur ne peut pas poser de domino, il va envoyer **null**. Par contre, c'est au gestionnaire du jeu de se rendre compte qu'un joueur a déposé tous ses dominos.

On *interdit* à un joueur d'avoir accès au gestionnaire (i.e., un joueur ne peut pas avoir comme attribut une instance du gestionnaire). A la rigueur, il peut accéder à des attributs **static** du gestionnaire.

Etape 1

Codez le gestionnaire. Ayez en tête qu'en théorie, quelqu'un d'autres que vous peut coder un joueur, en particulier, l'enseignant pourra préparer un joueur qui ne jouera pas forcément de manière honnête. Votre gestionnaire devra donc être robuste et faire en sorte que les joueurs respectent les règles.

Etape 2

Codez un joueur automatique naïf : lorsque plusieurs dominos sont possible, le joueur naïf ne va pas faire de raisonnement pour choisir le meilleur, il va en choisir un de manière arbitraire.

Etape 3

On va maintenant ajouter un joueur qui va jouer via la console. Implémenter la classe `JoueurClavier` qui va demander à un utilisateur de choisir le Domino à placer. Lorsque c'est au tour du `JoueurClavier` de jouer, il faut donc afficher l'état de la partie et demander à l'utilisateur de faire un choix.

Doit-on changer quelque chose au gestionnaire de partie pour jouer avec ce nouveau joueur ?

Etape 4

Ecrire des tests unitaires pour tester le joueur automatique.
Peut-on utiliser ces tests pour le joueur clavier ?

Etape 5 : traitements pour des exceptions

Si le gestionnaire demande au joueur de jouer alors qu'il n'a plus de domino, l'appel devrait lever une exception. Implémentez cette exception et ajouter un test pour vérifier que votre exception est bien levée.

Dans l'implémentation proposée, c'était le gestionnaire qui appelle une méthode du joueur pour qu'il pose son domino (ou renvoyer `null` s'il ne peut déposer de dominos). Que propose votre code dans le cas où le joueur donne un domino non valide ? Prenez en compte cette possibilité et écrivez un test `Junit` pour tester votre solution.

Soumission

- Le projet et à rendre le *3 janvier* sur la plateforme `mycourse`
- vous devez soumettre l'ensemble des fichiers sources dans une archive `zip`.
- tous vos fichiers sources doivent être commentés : avec des éléments pour génerer de la `javadoc` et avec éléments explicatifs dans le code si besoin.
- le projet est *individuel*