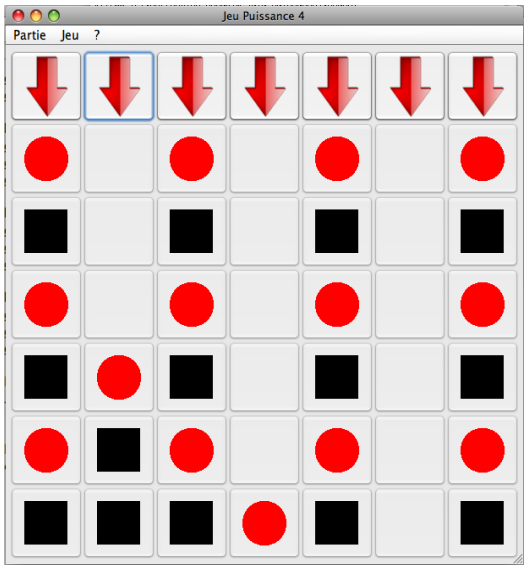
Projet HLIN 505 – 2014/2015

Jeu Puissance 4

Sujet du projet :



Le plateau de jeu est une grille de 7 colonnes et de 6 lignes dans lesquelles il est possible d'insérer des jetons. C'est un jeu à deux joueurs (humains) et les jetons sont, dans une partie donnée, de deux couleurs et formes différentes. A tour de rôle, un joueur met un jeton (de sa couleur) dans une des colonnes. Sous l'effet de la gravité le jeton tombe tout en bas de la grille ou sur le dernier jeton joué dans cette colonne. Dès que l'un des deux joueurs a aligné 4 de ses jetons soit verticalement, soit horizontalement ou en diagonale (voir la figure en face), il a gagné. Si la grille est remplie avant qu'un des joueurs ait pu aligner 4 jetons, il y a match nul.

1. Créer un diagramme de classes représentant chaque concept du domaine métier de l'application : Jeton, Case, Grille, Joueur, Partie et Jeu, entre autres. Donner les attributs de chaque classe. Par exemple, la classe Jeton doit contenir au moins deux attributs, un premier représentant la couleur et un autre représentant la forme (carré ou cercle). Mettre en place les associations entre les différentes classes. Identifier les opérations que doivent implémenter ces classes, les ajouter au diagramme, et commenter leur implémentation dans des notes UML.

2. Traduire les classes en code Java.

3. Implémenter les différentes méthodes et leur ajouter pré-conditions, post-conditions et invariants. Utiliser les exceptions pour signaler les pré-conditions non respectées, et les assertions pour vérifier les post-conditions et les invariants. Pour ce faire, définir un certain nombre de classes d'exceptions : ColonnePleineException et TimeoutException, par exemple.

4. Développer une interface graphique pour visualiser la grille de jeu et placer des jetons, et proposer un menu pour démarrer et arrêter des parties, afficher l'historique des parties (pseudos des joueurs qui ont gagné, record de la partie la plus rapide, ...), etc. Écrire plusieurs classes pour cette interface graphique et plusieurs classes pour les écouteurs d'événements (ne pas tout mettre dans une même classe !!!). Ne pas modifier les classes du domaine métier après la construction de l'interface graphique. Ces classes ne doivent pas dépendre des classes de l'interface graphique (qui peut être changée). Ce sont les classes des écouteurs d'événements qui doivent mettre en relation ces deux catégories de classes. Toutefois, les classes de l'interface graphique peuvent contenir des références vers les classes du domaine métier. Cette structure correspond à un schéma connu dans le développement des applications interactives, qui s'appelle le patron MVC (Modèle-Vue-Contrôleur). Le « Modèle » est composé des classes du domaine métier, la « Vue » est constituée des classes de l'interface graphique, et le « Contrôleur » est représentée par les classes des écouteurs d'événements.

5. Ajouter au code un invariant permettant de vérifier que les classes du domaine métier

(modèle) ne comportent pas de références vers des objets des classes de l'interface graphique (vue). Nous allons limiter cette vérification aux attributs de ces classes qui : i) ne doivent pas être d'un type qui correspond à une classe de la vue, et ii) ne doivent pas avoir comme valeur une référence vers un objet d'une des classes de la vue. Pour pouvoir identifier si une classe appartient à l'interface graphique (vue), au domaine métier ou correspond à un écouteur d'événements, appliquer aux différentes classes les annotations suivantes, après les avoir définies dans le code de votre application : @Modele, @Vue, @Controleur. Ensuite, il suffit d'utiliser la méthode isAnnotationPresent(...) sur les objets Class.

6. Concevoir un autre type de cases : les cases réservées à un des deux joueurs. Cette case refusera d'accueillir un jeton de l'autre joueur. Ce type de case servira pour le mode de jeu « cases spéciales », voir ci-après. Vous pouvez imaginer d'autres types de cases si vous le souhaitez.

7. Proposer des grilles pour différents modes de jeu. Pour un niveau «Simple», la grille de jeu est composée de 7 colonnes et 6 lignes, comme précisé ci-dessus. Le nombre de jetons à aligner est de 4. Pour un mode du jeu « Puissance N », proposer une grille de jeu plus grande et un nombre de jetons à aligner plus important. Pour un mode de jeu « chronométré », mettre en place un mécanisme de compte à rebours (ou un *timeout)* au delà duquel le joueur passera automatiquement la main à l'autre joueur. Pour un mode de jeu « cases spéciales », la grille sera construite en plaçant aléatoirement une case réservée pour chaque joueur.

8. En exploitant le mécanisme de sérialisation des objets, proposer dans le menu la possibilité aux joueurs d'enregistrer dans un fichier une Partie commencée mais pas encore terminée. De cette manière, les joueurs peuvent complètement arrêter l'application, puis plus tard, redémarrer celle-ci et relancer la Partie à nouveau (au même point où elle s'est arrêtée). L'historique des parties doit être sérialisé aussi pour qu'il soit accessible après la fermeture de l'application. L'implémentation de la gestion de l'historique des parties est en bonus. La laisser en dernier.

Modalités pour rendre le projet :

- Créer une archive ZIP composée des éléments suivants :
  - un fichier PDF contenant les diagrammes de classes et les captures d'écran de toute l'application
  - les sources (fichiers .java) de l'application
  - les éventuelles autres ressources de l'application (images, ...)

- Envoyer cette archive par e-mail à l'enseignant qui assure les TD/TP de votre groupe : Clementine.nebut@lirmm.fr ou Abdelhak.Seriai@lirmm.fr

- La date limite de la remise de ces archives est fixée au lundi 5 janvier 2015 à 18h.

- Le travail doit être réalisé soit individuellement ou par binôme (les projets qui seront rendus par des groupes de 3 étudiants ou plus ne seront pas recevables). Toute similarité anormale entre les rendus de 2 groupes sera pénalisée.