

M1 Info : Ingénierie Logicielle

2008/09

Examen Décembre 2008 (durée 2h)

Tous documents autorisés
Barème indicatif sur 20 points

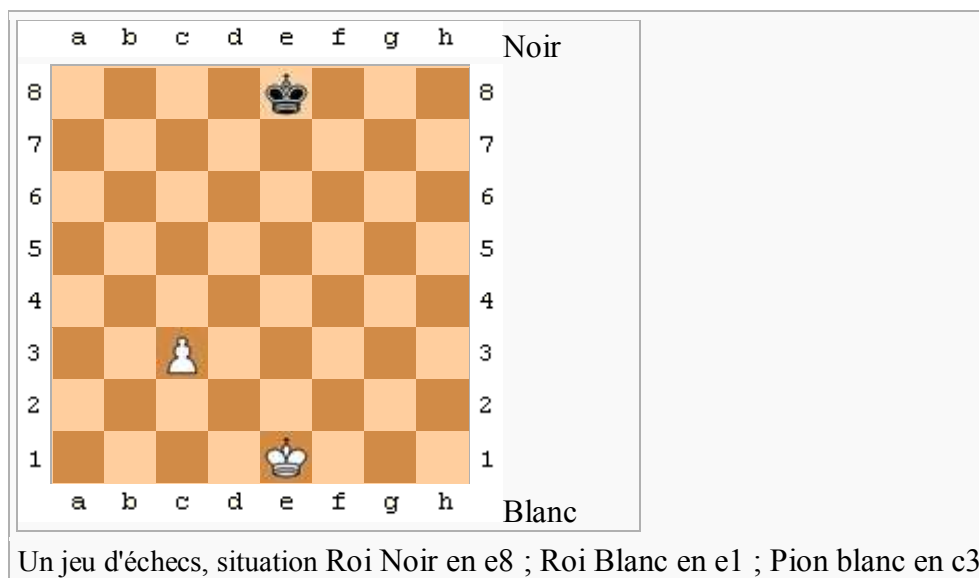
Question de Cours (4 points)

1. En quoi consiste le méta-modèle d'UML ? Comment s'en servir (proposez un exemple) ?
2. Expliquez en quoi une machine à états peut permettre de définir des tests unitaires. Précisez quel type de couverture on peut obtenir.
3. Par quel(s) mécanisme(s) peut-on spécialiser le comportement d'un outil utilisant UML ? Par exemple, comment permettre à l'utilisateur de paramétrer le type utilisé pour représenter une association 0,* dans un outil générant du code ?
4. Pourquoi est-ce qu'on peut dire que les diagrammes de séquence représentant les classes internes d'un composant doit raffiner un diagramme de séquence présentant une ligne de vie par composant ? Pourquoi réaliser ce type de diagrammes fin ?

Problème : ChessFighter

Une société proposant un site web dédié au jeu d'échecs (articles, tutoriaux, compte rendus des compétitions...) désire créer une application destinée à comparer des intelligences artificielles (IA) jouant aux échecs.

Le jeu d'échecs est un jeu à deux joueurs, l'un jouant les pièces blanches et l'autre les pièces noires. Il se joue sur un plateau carré de 8 cases sur 8, appelé échiquier. Les colonnes de l'échiquier portent les lettres de **a** à **h**, et les lignes les chiffres de **1** à **8**. La case **a1** est par convention la case la plus à gauche de la première ligne vu du côté du joueur Blanc (en bas).



Les pièces sont le Roi, la Dame, les Fous, les Cavaliers, les Tours et les Pions. La position initiale des pièces est fixée. Chaque pièce est soumise à des règles de déplacement particulières, dont certaines sont détaillées ci-dessous.

Les joueurs jouent à tour de rôle, un coup consiste à déplacer une pièce de sa couleur. Le joueur Blanc joue le premier. La partie se termine quand un des joueurs mate son adversaire (victoire), ou quand une situation de pat est obtenue (match nul).

L'objectif est de développer un arbitre virtuel pour une partie, qui offre une API de manipulation simple. Les IA seront connectées logiquement à l'application ChessFighter, qui détermine le vainqueur de chaque manche et comptabilise le classement des joueurs virtuels, selon des règles (classement ELO) qui ne seront pas détaillées ici.

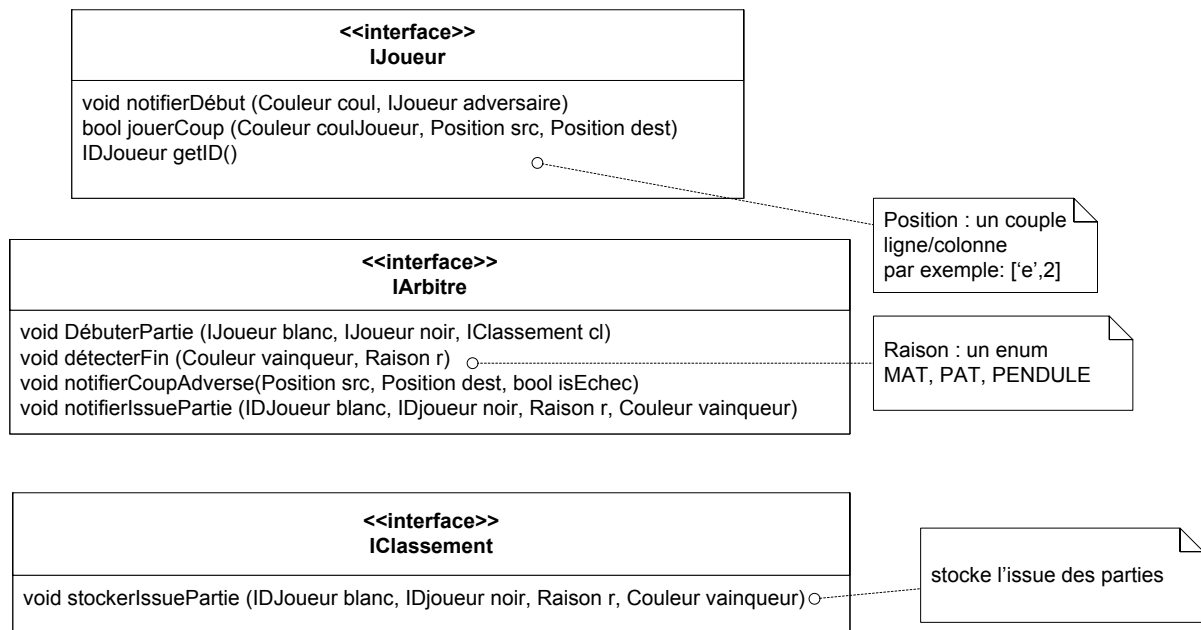
On démarre une partie en configurant les deux joueurs virtuels, décrits par des interfaces à préciser. Les joueurs jouent alors chacun leur tour en informant ChessFighter de leurs déplacements. Un déplacement peut être décrit par deux coordonnées donnant la case de départ et d'arrivée de la pièce, par exemple e2-e4. La case de départ doit correspondre à une case occupée par une pièce de la couleur du joueur. Le déplacement doit également être légal vis-à-vis de règles qui dépendent de la pièce manipulée (détaillées question 4), sans quoi le coup sera refusé par l'application, qui demandera au joueur de jouer un autre coup.

Question 1 (5 point)

On définit une découpe de l'application en trois parties :

- Un composant « classement » dont le rôle est de classer les joueurs en comptabilisant victoires, défaites et nuls. Il enregistrera de façon persistante le bilan de chaque partie. Il peut également être interrogé pour connaître le classement d'un joueur, des statistiques... mais on ne détaillera pas les opérations de cette interface d'interrogation « Interrogation ».
- Un composant « joueur » qui joue des parties. Il doit pouvoir être informé du début et de la fin d'une nouvelle partie. Ensuite il sera informé des coups de l'adversaire par l'arbitre, et fournira en retour son propre coup.
- Un composant arbitre, instancié pour chaque nouvelle partie. Il vérifie que les coups joués par les composants « joueur » sont légaux sans quoi le coup n'est pas validé. Le même joueur doit alors jouer un autre coup. Quand le coup est validé, il informe l'adversaire du coup joué. Il contrôle également à chaque coup la position des pièces et annonce les échecs, échecs et mat et fin de partie sur situation particulière (Fin du temps de pendule, situation de Pat) quand ils se produisent. Enfin il est chargé d'informer le composant classement de l'issue de la partie.

a) (2,5 pt) On propose les interfaces suivantes pour ces composants :



Critiquez et corrigez cette proposition.

b) (1,5 pt) Précisez à l'aide d'un diagramme de composants les interfaces requises et offertes des différents composants.

c) (1 pt) Représentez sur un diagramme une instantiation possible des composants, dans une situation de match opposant « Deep Blue » à « Simply Red ».

Question 2 (3 point)

a) (2 pt)

Définissez un diagramme de classes permettant de représenter l'état d'une partie (dans le composant arbitre) : situation de l'échiquier, des joueurs...

Détaillez les attributs (type précis) et les associations (navigabilité, composition/aggregations...) autant que possible.

On ne demande pas à cette question de détailler les opérations de ces classes.

b) (1 pt)

Précisez les liens entre vos classes et les interfaces de la question 1.

Question 3 (4 point)

Le jeu d'échecs se caractérise également par l'utilisation d'une pendule pour limiter la durée de la partie.

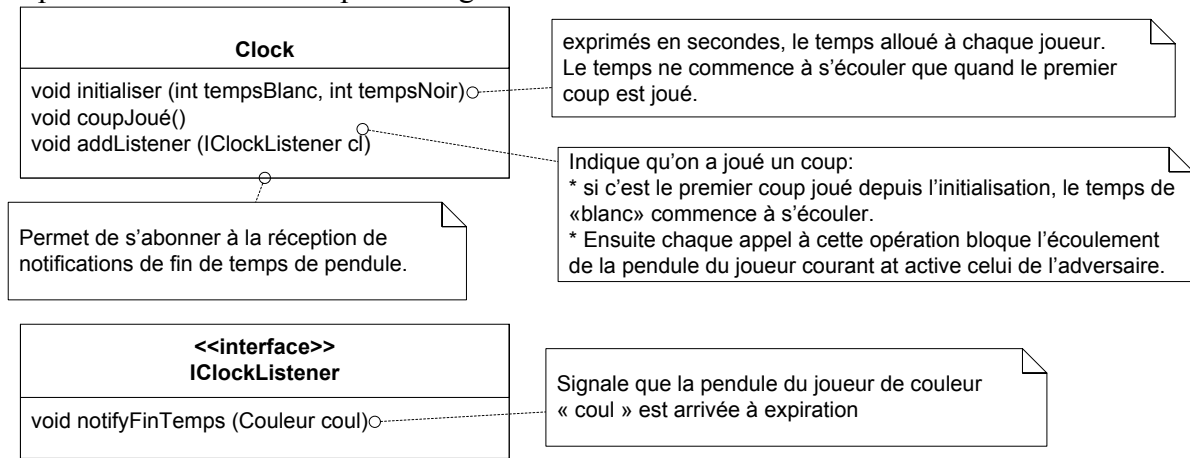
Chaque joueur dispose du même temps de réflexion pendant lequel il doit jouer tous ses coups. En d'autres termes, la pendule du joueur Blanc est décrémentée quand c'est son tour de jouer, et celle du joueur Noir quand c'est aux noirs de jouer.

Les parties sont souvent rapides, d'une durée de quelques minutes par pendule en général. Si la pendule d'un joueur atteint 0, il perd la partie. La partie peut également se terminer avant à travers les règles citées ci-dessus (mat, ou nul).

Ce comportement sera intégré dans le composant Arbitre, dont l'opération d'initialisation `DébuterPartie(...)` prendra maintenant en argument le temps de réflexion initial des joueurs en plus de ses autres arguments (cf. Question 1).

On suppose qu'une classe Clock et une interface IClockListener ont déjà été réalisées dans le cadre d'un autre développement. La Clock sert à gérer une pendule d'échecs, c'est-à-dire gérant l'écoulement du temps pour les deux joueurs.

La partie réalisée se décrit par le diagramme de classes suivant.



- a) (1 pt) Quel Design Pattern reconnaissez-vous ici ? Expliquez brièvement son utilité.
- b) (1 pt) Expliquez comment intégrer et utiliser cette classe Clock et l'interface liée en enrichissant le diagramme de classe défini question 2.
- c) (2 pts) Détaillez comment se fait l'interaction entre Clock et son ClockListener, à l'aide de diagrammes de séquence ou les lignes de vie représentent respectivement la classe Clock et un IClockListener. On représentera :
- c.1) La phase de mise en place de la partie et de la pendule, avec 5 minutes par joueur.
 - c.2) L'interaction engendrée à fois qu'un joueur joue un coup valide.
 - c.3) Ce qui se passe quand l'horloge du joueur blanc atteint 0.

Question 4 (4 point)

On a identifié lors de l'analyse le test de validation suivant :

Test TV2 : Fin de partie sur pendule

Contexte : Une partie a été démarrée avec une durée de pendule de 1 minute, cf test de validation TV1, à exécuter avant celui-ci.

Entrée : le coup blanc : « e2-e4 »

Scénario :

1. Le joueur Blanc joue le coup « e2-e4 ».
2. On attends une minute, sans toucher l'interface.

Résultat attendu : ChessFighter doit annoncer une victoire des Blancs.

Moyen de Vérification : utiliser un chronomètre externe à l'application.

On souhaite mettre en place des tests d'intégration pour s'assurer du bon fonctionnement de l'arbitre.

A cette fin, on souhaite implémenter une classe bouchon minimale permettant de réaliser un test d'intégration du composant Arbitre correspondant au test de validation ci-dessus.

- a) (1 pt) Quel(les) interface(s) doit réaliser une classe bouchon pour représenter l'environnement entier du point de vue du composant Arbitre ?
- b) (3 pts) Expliquez comment on pourrait implémenter ce test d'intégration en écrivant des opérations de cette classe Bouchon et/ou un script de test. Formulez le test en pseudo-code.

