Diagrammes de séquence UML

Exercice 1

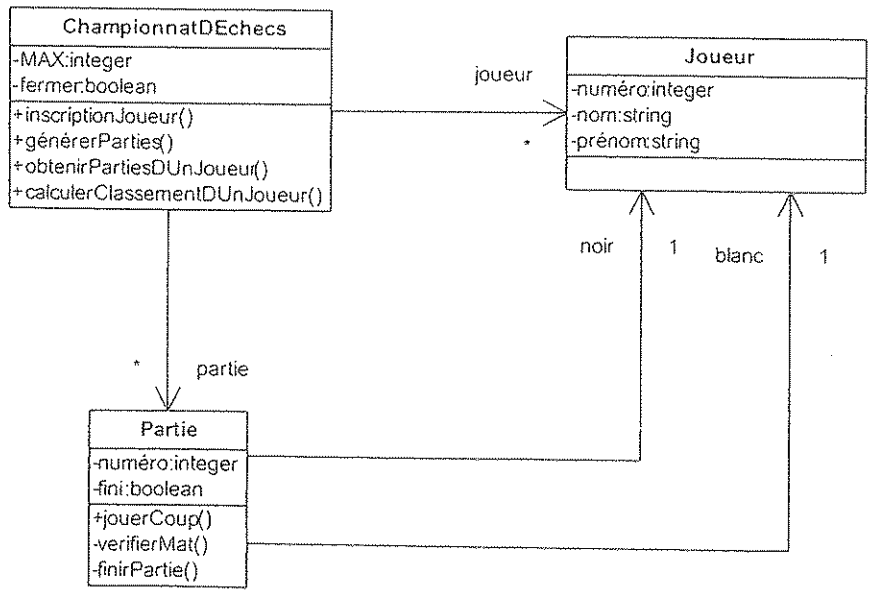
L'application `ChampionnatEchecs`, qui devra permettre de gérer le déroulement d'un championnat d'échecs est actuellement en cours de développement. L'équipe de développement n'a pour l'instant réalisé qu'un diagramme de classes de cette application *(voir figure 6.8)*.

La classe `ChampionnatDEchecs` représente un championnat d'échecs. Un championnat se déroule entre plusieurs joueurs (voir classe `Joueur`) et se joue en plusieurs parties (voir classe `Partie`). La propriété `MAX` de la classe `ChampionnatDEchecs` correspond au nombre maximal de joueurs que le championnat peut comporter. La propriété `fermer` permet de savoir si le championnat est fermé ou si de nouveaux joueurs peuvent s'inscrire.

`ChampionnatDEchecs` possède les opérations suivantes :

- `inscriptionJoueur(in nom:string, in prénom:string)` : integer permettant d'inscrire un nouveau joueur dans le championnat si le nombre de joueurs inscrits n'est pas déjà égal à `MAX` et si le championnat n'est pas déjà fermé. Si l'inscription est autorisée, cette opération crée le joueur et retourne son numéro dans le championnat.
- `générerPartie()` : permet de fermer le championnat et de générer toutes les parties nécessaires.
- `obtenirPartieDUnJoueur(in numéro :integer) : Partie[*]` : permet d'obtenir la liste de toutes les parties d'un joueur (dont le numéro est passé en paramètre).
- `calculerClassementDUnJoueur(in numéro :interger)` : integer permettant de calculer le classement d'un joueur (dont le numéro est passé en paramètre) pendant le championnat.

Figure 6.8
Classes de l'application ChampionnatEchecs



La classe `Partie` représente une des parties du championnat. La classe `Partie` est d'ailleurs associée avec la classe `ChampionnatDEchecs`, et l'association précise qu'un championnat peut contenir plusieurs parties. Une partie se joue entre deux joueurs. Un joueur possède les pièces blanches et commence la partie alors que l'autre joueur possède les pièces noires. Les associations entre les classes `Partie` et `Joueurs` précisent cela. La propriété `numéro` correspond au numéro de la partie (celui-ci doit être unique). La propriété `fini` permet de savoir si la partie a déjà été jouée ou pas.

La classe `Partie` possède les opérations suivantes :

- `jouerCoup(in coup:string)` : permet de jouer un coup tant que la partie n'est pas finie. Le traitement associé à cette opération fait appel à l'opération `vérifierMat` afin de savoir si le coup joué ne met pas fin à la partie. Si tel est le cas, l'opération `finirPartie` est appelée.
- `vérifierMat()` : boolean permettant de vérifier si la position n'est pas mat.
- `finirPartie` : permet de préciser que la partie est finie. Il n'est donc plus possible de jouer de nouveaux coups.

Source : UML 2 pour les développeurs, X. Blanc et I. Mounier, Eyrolles, 2006

La classe `Joueur` représente les joueurs du championnat. La classe `Joueur` est d'ailleurs associée avec la classe `ChampionnatDEchecs`, et l'association précise qu'un championnat peut contenir plusieurs joueurs. La propriété `numéro` correspond au numéro du joueur (celui-ci doit être unique). Les propriétés `nom` et `prénom` permettent de préciser le nom et le prénom du joueur.

Un championnat d'échecs se déroule comme suit :

- Un administrateur de l'application crée un championnat avec une valeur `MAX`.
- Les participants peuvent s'inscrire comme joueurs dans le championnat.
- L'administrateur crée l'ensemble des parties.
- Les participants, une fois inscrits, peuvent consulter leur liste de parties.
- Les participants, une fois inscrits, peuvent jouer leurs parties. Nous ne nous intéressons qu'aux coups joués par chacun des deux joueurs. Nous ignorons l'initialisation de la partie (identification du joueur qui a les pions blancs et donc qui commence la partie).
- Les participants peuvent consulter leur classement.

Dans les questions suivantes, nous allons spécifier des exemples d'exécution de `ChampionnatDEchecs` avec des diagrammes de séquence.

Question 52 *Comment modéliser les administrateurs et les participants ?*

Question 53 *Représentez par un diagramme de séquence le scénario d'exécution correspondant à la création d'un championnat et à l'inscription de deux joueurs. Vous assurerez la cohérence de votre diagramme avec le diagramme de classes fourni à la figure 6.8.*

Question 54 *Représentez par un diagramme de séquence le scénario d'exécution correspondant à la création de l'ensemble des parties pour le championnat créé à la question 53. Vous assurerez la cohérence de votre diagramme avec le diagramme de classes fourni à la figure 6.8.*

Question 55 *Représentez par un diagramme de séquence le scénario d'exécution correspondant au déroulement de la partie d'échecs entre deux joueurs. Vous pouvez considérer une partie qui se termine en quatre coups. Vous assurerez la cohérence de votre diagramme avec le diagramme de classes fourni à la figure 6.8..*

Exercice 2 : Illustration de cas d'utilisation

Une équipe de développement souhaite réaliser une application `Calculus` qui permet à des utilisateurs d'effectuer des opérations arithmétiques simples sur des entiers : addition, soustraction, produit, division. Cette application a aussi une fonction mémoire qui permet à l'utilisateur de stocker un nombre entier qu'il pourra ensuite utiliser pour n'importe quelle opération. Les opérations peuvent directement s'effectuer sur la mémoire. L'utilisateur se connecte et ouvre ainsi une nouvelle session. Puis, dans le cadre d'une session, l'utilisateur peut demander au système d'effectuer une suite d'opérations.

Question 57 Réalisez le diagramme des cas d'utilisation et listez les scénarios d'exécution alternatifs

Question 58 *Utilisez des diagrammes de séquences pour représenter les différents scénarios d'exécution du service `Calculus`.*

Question 59 *Pour chacune des instances apparaissant dans votre diagramme de classes, créez la classe correspondante.*