esprit Se former autrement	Devoir Surveillé Semestre : 1 ■ 2 ☐

Modules : Modélisation et Programmation Objet
Enseignant(s) : UPs GL-BD , JAVA
Classe(s) : 3A
Documents autorisés : OUI ☐ NON ■ Nombre de pages : 3
Date : 30/10/2017 Heure : 13H15 Durée : 1H30

Etude de cas :

La fédération tunisienne de basket-ball souhaite disposer d'un système informatique permettant de gérer les clubs, leurs membres (joueurs et entraineur), les matchs et les tournois. Ainsi, un administrateur peut ajouter, mettre à jour ou supprimer un joueur, ajouter un tournoi et ajouter un match. À tout moment, les joueurs, l'administrateur et l'entraineur peuvent visualiser les données sur un tournoi avec ses matchs.

Un membre est caractérisé par un Identifiant (ID), un nom et un prénom. Un joueur par une taille et un poids. Un entraineur a un nombre d'années d'expérience.

Durant sa carrière, le joueur peut jouer dans plusieurs clubs, pendant une période de temps et avec un salaire bien déterminé. Un club est caractérisé par un code et peut contenir au maximum 29 joueurs et un seul entraineur.

Les tournois sont caractérisés par un numéro unique, un nom, une date de début, une date de fin, une adresse et un type (national ou international). Un match est caractérisé par un numéro unique, une date et une durée.

L'entraineur du club a la possibilité de proposer un match. Pour ce faire, il envoie une demande à l'administrateur avec le nom du club adverse. Après réception de cette demande, l'administrateur contacte, via l'application, l'entraineur du club adverse afin de confirmer sa disponibilité. Si la réponse de l'entraineur du club adverse est favorable, l'administrateur finalise le traitement de la demande par l'enregistrement d'un nouveau match. Une notification automatique est alors envoyée aux deux entraineurs. Dans le cas inverse, un message de refus sera envoyé à l'entraineur du club.

Partie A

Langage de modélisation UML

Travail demandé :

1. Elaborer un diagramme de cas d'utilisation pour le système de la fédération. **(7 pts)**
2. Elaborer le diagramme d'activités relatif au processus de traitement d'une demande de match. **(7 pts)**
3. Elaborer le diagramme de classes d'analyse relatif au système de la fédération en complétant le diagramme de la figure 1. **(6 pts)**

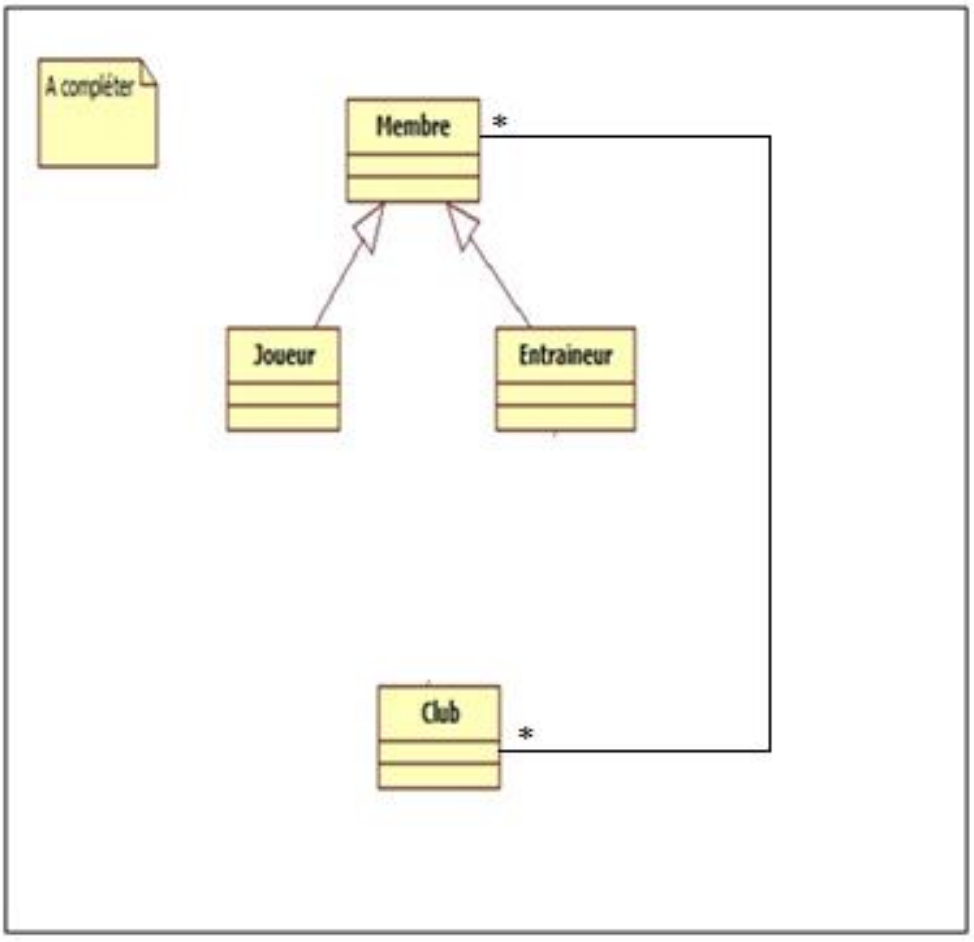

Figure 1 - Diagramme de classes d'analyse à compléter

Partie B JAVA

Travail demandé :

Respecter les Règles (Conventions) du Nommage Java

Les Getters et les Setters ne sont à implémenter que si leur demande est explicite.

Toute la partie Java est basée sur le diagramme de classe non complet fourni dans la partie UML (Figure 1)

I. Implémenter la classe **Membre**. Elle doit comprendre : (2.5 points)
   - o Uniquement un constructeur paramétré. (0.5)
   - o Les Getters et les Setters **uniquement** pour l'attribut identifiant. (0.5)
   - o La méthode **String toString()**.(0.5)
   - o La méthode **boolean equals(Object o).** Sachant que deux membres sont égaux s'ils ont le même identifiant et le même nom. (1 point)

II. Implémenter la classe **Joueur**. Veiller à ce qu'elle contient un constructeur et la méthode **toString()**. On suppose que les getters et les setters sont implémentés au niveau de la classe joueur et que la classe Entraineur est déjà implémentée. (2 points)

III. Implémenter la classe **Club**. Elle doit comprendre : (1 point)
   1. La méthode **void ajouter (Membre membre)** : cette méthode permet d'ajouter un nouveau membre à un club, tout en prenant en considération qu'un membre ne peut être affecté deux fois au même club et qu'un club ne peut contenir qu'un seul entraineur. (3.5 points)
   2. La méthode **void supprimer(Membre membre)** : cette méthode permet de supprimer un membre du club. (2 points)
   3. La méthode **String toString()** : cette méthode retourne une chaine de caractère contenant le code du club et **uniquement** la liste de ses **Joueurs**. (2 points)
   4. La méthode **void afficherTailles()** : cette méthode affiche uniquement les tailles des joueurs du club. (2 points)
   5. La méthode **void supprimerJoueurs(String nom)** : cette méthode permet de supprimer tous les joueurs d'un club dont le nom est passé en paramètre (2 points).
   6. La méthode **float moyennePoids()** : cette méthode permet de retourner la moyenne des poids des **joueurs** d'un club. Si la moyenne des poids est supérieure à 80, une Exception personnalisée **PoidsException** sera lancée avec le message suivant « Poids des joueurs trop élevé » (3 points). Vous êtes amenés à implémenter la classe **PoidsException**