SCIENCES SORBONNE UNIVERSITÉ

LU2IN002-2021oct
Éléments de programmation
par objets avec Java

Examen du 10 janvier 2022 – Durée : 1 heure 30 minutes

Seul document autorisé : **feuille A4 manuscrite, recto-verso.**
Pas de calculatrice ou téléphone. Barème indicatif sur 40.

Remarques préliminaires : *la visibilité des variables et des méthodes à définir, ainsi que leurs aspects statique, ou final ne sera pas précisé. C'est à vous de décider de la meilleure solution à apporter. Si la méthode* `toString` *n'est pas demandée dans la question, il n'est pas nécessaire de la fournir. Sans aucune précision donnée dans la question, une méthode ne rend rien.*

On souhaite écrire des classes pour gérer une version simplifiée de serveur de tournois entre joueurs. Plusieurs classes composent ce programme :

- les classes de représentation des joueurs : `Joueur`, `JoueurAbo`, et `JoueurFTP`. Les joueurs sont soit des joueurs abonnés (`JoueurAbo`), soit des joueurs "Free to play" (`JoueurFTP`).
- la classe `Serveur`. Un serveur est unique et il gère les tournois. Il s'occupe de créer un tournoi, d'enregistrer des joueurs dans un tournoi, de lancer un tournoi (quand celui-ci contient suffisamment de joueurs) et de le clôturer quand celui-ci est terminé.
- la classe `Tournoi` qui est dédiée à l'organisation d'un tournoi entre joueurs. Les joueurs y sont inscrits par le serveur, puis le tournoi est lancé pour désigner le joueur vainqueur du tournoi.

Question 2.1 (4 points) Écrire les classes suivantes :

La classe `Joueur` qui possède un unique constructeur avec un argument donnant le `nom` du joueur (String) et un assesseur (getter) permettant d'obtenir ce nom. Elle possède un attribut `serveur` qui contient la référence d'un `Serveur`. Cette variable est initialisée à `null`. Ajouter une méthode `enregistrement()` qui prend un argument pour donner une valeur à cet attribut et ne rend rien. Cette classe possède aussi une méthode `statut()` qui rend un booléen : `true` si le joueur est abonné, `false` sinon. Cette classe ne doit pas être instanciable.

La classe `JoueurAbo` qui possède un attribut `nbPoints` (entier) donnant le nombre de points que le joueur a gagné lors des tournois auxquels il a participé, un constructeur à 2 arguments (nom et nombre de points de tournoi initiaux), un constructeur avec un seul argument (nom, le nombre de points étant alors initialisé à 0), un assesseur (getter) fournissant le nombre de points obtenus par le joueur et une méthode `addPoints()` qui prend un argument un entier qui est ajouté au nombre de points du joueur.

La classe `JoueurFTP`, sans attribut mais avec une variable permettant de compter le nombre de joueurs FTP créés. Le nom d'un tel joueur est créé automatiquement de la forme "j1", "j2",... où le nombre correspond au numéro de création du joueur. Cette classe ne contient qu'un constructeur sans argument et une méthode pour connaître le nombre de joueurs FTP qui ont été créés.

Question 2.2 (5 points) On souhaite doter les 3 classes précédentes d'une capacité de clonage mais, à cause de la classe `Joueur` qui n'est pas instanciable, il n'est pas possible de définir une méthode `clone()`. On utilise donc des constructeurs par recopie. Donner le code des constructeurs par recopie pour chacune des classes `Joueur`, `JoueurAbo`, et `JoueurFTP`.

Question 2.3 (4 points) Écrire la classe `Tournoi` contenant 2 constantes entières `MAXJOUEURS` qui donnent le nombre maximum de joueurs pouvant participer à un tournoi (tous les tournois ont le même maximum), initialisée à la valeur 12, et `PTSVAINQUEUR` qui donne le nombre de points de tournoi affecté au vainqueur d'un tournoi, initialisée à 10. Ces 2 constantes sont valables pour tout tournoi créé.

Cette classe contient 4 attributs : un entier `ident` qui est l'identifiant du tournoi créé. La valeur de cet identifiant est donné en argument de l'unique constructeur de cette classe ; un tableau de `Joueur` de nom `tabJoueurs` ; un entier `nbJoueurs` qui donne le nombre de joueurs inscrits dans le tournoi, et qui vaut donc 0 au départ ; une variable `vainqueur` de type `Joueur` qui contiendra le joueur qui aura gagné le tournoi à l'issue de son lancement, au départ cette variable contient la valeur `null`.

Pour compléter, la classe contient 2 accesseurs (getter) pour obtenir l'identifiant et le nombre de joueurs du tournoi. Elle contient aussi une méthode `ajoute` qui prend un joueur en argument et l'ajoute dans le tableau des joueurs inscrits au tournoi, s'il reste encore de la place dans ce tournoi (ie. le nombre de joueurs maximum n'a pas été atteint).

Question 2.4 (4 points) Donner le code de la méthode `clone()` qui permet de cloner un tournoi.

Attention : on rappelle qu'un `Joueur` n'est pas instanciable.

Question 2.5 (4 points) Écrire la classe `Serveur`. Cette classe doit garantir qu'il ne peut exister qu'une unique instance de serveur dans le programme. Elle contient un attribut qui est un `ArrayList` de tournois de nom `tabTournois` ; une méthode `getInstance()`, sans argument, qui rend la référence du serveur ; une méthode `creeTournoi()` sans argument qui créé un nouveau tournoi et l'ajoute à `tabTournois`. L'identifiant qui est donné au tournoi créé correspond au nombre de tournois qui ont été créés depuis le lancement du programme.

Question 2.6 (2 points) Pour qu'un joueur s'inscrive dans un tournoi, donner le code de la méthode `inscription()` de la classe `Serveur`. Cette méthode prend en argument un joueur et l'inscrit dans un des tournois de `tabTournois`. Le choix du tournoi est fait de façon aléatoire.

Question 2.7 (2 points) Donner le code de la méthode `inscritTournoi()`, sans argument, de la classe `Joueur` qui permet à un joueur de demander au serveur de l'inscrire dans un tournoi.

Question 2.8 (3 points) Donner le code de la méthode `lance()`, sans argument, de la classe `Tournoi`. Cette méthode simule (très simplement) le déroulé du tournoi : le vainqueur est choisi aléatoirement parmi les joueurs qui sont inscrits dans ce tournoi. Ce joueur est alors affecté à l'attribut `vainqueur`. De plus, si le vainqueur est un joueur abonné, il faut ajouter `PTSVAINQUEUR` à son nombre de points.

Remarque : s'il existe déjà un vainqueur, ou s'il n'y a aucun joueur inscrit, cette méthode ne fait rien.

Question 2.9 (3 points) Donner le code de la méthode `lanceTournoi()` de la classe `serveur`. Cette méthode prend en argument l'**identifiant d'un tournoi**. Le tournoi demandé, s'il existe dans `tabTournois`, est lancé. Une fois que ce tournoi s'est terminé, il est alors supprimé de la liste des tournois du serveur.

Question 2.10 (7 points) Afin de gérer la saturation d'un tournoi (nombre de joueurs maximum atteint), on choisit d'utiliser une solution à base d'exception.

1. Donner le code de la classe `PlusDePlaceException` pour créer une exception. Cette classe ne contient qu'un unique constructeur qui prend en argument une chaîne de caractères.
2. Donner une nouvelle version de la méthode `ajoute()` afin qu'elle lève une exception de type `PlusDePlaceException` dans le cas où l'on demande d'ajouter un joueur dans le tournoi qui est déjà au maximum. Le message associé à cette levée d'exception doit être "Tournoi X plein !" où X est l'identifiant du tournoi.
3. Donner une nouvelle version de la méthode `inscritTournoi()` de la classe `Serveur`. Cette méthode doit récupérer l'exception `PlusDePlaceException` susceptible de se produire. Si tel est le cas, un nouveau tournoi est créé par le serveur et une demande d'inscription est relancée pour le joueur.

Question 2.11 (2 points) Donner le schéma UML **fournisseur** des classes réalisées.