Projet JAVA 2

Jeu de Questions / Réponses

---

1) Instructions générales

Ce projet est à réaliser entièrement en langage Java.

Documentation

Vous pouvez vous aider des :

- Supports de cours ;
- Des ressources disponibles sur Internet, notamment sur le site Oracle ou sur la bibliothèque Scholarvox offerte sur myEffrei.

Modules, APIs et concepts concernés par ce programme

La réalisation de ce projet implique l'utilisation des :

- Concepts généraux de Java (héritage, polymorphisme, encapsulation, surcharge, méthodes...)
- Constructeurs, interfaces, …
- Collections
- Threads
- Input/Output
- Gestion de la persistance
- Classes génériques
- Swing
- Pratiques : SOLID , expressions lambda, …

Planning

- Mise en ligne du projet : **04/05/2020**
- Revue de code : semaine du **25/05/2020**
    - Une présentation de l'état d'avancement de chaque équipe ainsi que les perspectives tracées avant la remise du projet final. Elle se déroulera sur une durée de 10 minutes. Une note préliminaire sera attribuée.

Consignes d'organisation

- Le projet est à réaliser en équipes de 4 membres.

- La liste des équipes est à fournir à vos intervenants respectifs au plus tard le **08/05/2020**. Un mallus sera appliqué pour tout retard.
- La note de projet comptera pour 60% de la note finale du cours Java 2.

Rendus attendus

- Les projets sont à rendre au plus tard le **11/06/2020.**
- Les livrables doivent inclure un fichier **.zip** ou **.jar** (bien faire attention à ce que les classes **.java** y soient présentes) et un rapport.
- Les livrables seront soumis dans des zones de dépôt dédiées sur Moodle.
- Le rapport doit synthétiser la conception technique de votre projet ainsi que vos choix de solutions justifiés.

2) Cahier des charges

L'objectif de ce projet est d'implémenter un jeu de Question / Réponse entre X (X >=4) joueurs au début et qui effectue à travers les phases du jeu des éliminations jusqu'à ce qu'il ne reste plus qu'un seul gagnant.

Le jeu se base sur une liste de questions auxquelles les candidats doivent répondre à tour de rôle. En fonction des réponses, un score est cumulé pour chaque joueur et les candidats ayant les meilleurs scores sont sélectionnés pour les phases suivantes.

Les questions traitent de H (H >=10) thèmes et sont de 3 types T différents :

- QCM : Le joueur peut choisir une réponse entre plusieurs ;
- VF : Le joueur peut répondre uniquement par vrai / faux ;
- RC : Le joueur peut saisir une réponse courte.

De plus, les questions ont 3 niveaux de difficulté :

- 1 : Niveau facile
- 2 : Niveau moyen
- 3 : Niveau difficile

La réalisation de cette application s'appuie sur les spécifications fonctionnelles détaillées dans la section qui suit.

Description fonctionnelle

Pour réaliser le jeu de Question / Réponse, nous avons besoin d'implémenter les classes suivantes :

Les classes des types des questions

Une question de type QCM est décrite par un texte, 3 variables de réponses proposées et une variable bonne réponse.

Une question de type VF est décrite par son texte et sa bonne réponse de type booléen.

Une question de type RC est décrite par son texte et une variable bonne réponse de type chaine de caractères.

Pour chacune de ces classes, implémenter au minimum :

- Un constructeur
- Une méthode afficher qui affiche la question dans la forme adéquate.

Question

Une question est décrite par un numéro, un thème, un niveau de difficulté, un énoncé pouvant avoir 3 types T (QCM, VF, RC). Les questions sont numérotées de manière séquentielle indépendamment de leurs thèmes.

Donner l'implémentation de la classe **Question<T>** comportant :

- Un constructeur
- Une méthode **Afficher** qui affiche le niveau de la question ainsi que son énoncé incluant le texte la question ainsi que ses champs de réponse en fonction de son type T.
- Une méthode **Saisir** qui permet la saisie d'une fonction d'un type T donné.

Thèmes

Un thème est désigné par une chaine de caractères (Exemple : Sciences, Sport, Histoire, ..). Nous supposons que l'on dispose de 10 thèmes différents qui sont stockés dans un tableau de type String. On peut remplacer un thème par un autre. Lorsqu'un thème est sélectionné, un indicateur est positionné sur ce thème pour l'éviter à la prochaine sélection.

Donner l'implémentation de la classe **Thèmes** comportant :

- Un initialiseur qui initialise le tableau des thèmes ;
- Une méthode **ModifierTheme** ;
- Une méthode **SélectionnerTheme** qui sélectionne un thème et renvoie l'indice du dernier thème choisi lors du jeu ;
- Une méthode **SelectionnerCinqThemes**
- Une méthode **Afficher** qui affiche tous les thèmes et la valeur de l'indicateur.

ListeQuestions

Pour chaque thème, l'on dispose d'un ensemble de questions dont le nombre et les types sont variables. L'ensemble des questions relatives à un thème est stocké dans une liste chainée de questions. A chaque fois, un indicateur de la question sélectionnée est mis à jour.

Donner l'implémentation de la classe **ListeQuestions** contenant :

- Un constructeur
- Une méthode **AfficherListe**
- Une méthode **AjouterQuestion** à la liste
- Une méthode **SupprimerQuestion** de numéro n de la liste

- Une (ou plusieurs) méthode(s) **SelectionnerQuestion** qui sélectionne une question pour un joueur selon une politique qui sera définie dans les différentes phases du jeu.

Phase

Une partie de jeu se déroule en plusieurs phases. A chaque phase, le joueur ayant le plus faible score est éliminé et les autres sont sélectionnés pour la phase qui suit.

Implémenter une interface **Phase** comportant les méthodes suivantes :

- **SelectionnerJoueurs**
- **PhasedeJeu**

NB : Il est possible d'ajouter toute autre classe jugée nécessaire pour implémenter une partie de jeu à partir de l'interface **Phase**.

Joueur

Un joueur est décrit par un numéro, un nom, un score et un état (sélectionné, gagnant, super gagnant, éliminé ou en attente). Pour chaque réponse correcte, le score est incrémenté d'une valeur dépendant de la phase du jeu. Le numéro du joueur est un numéro qui commence à 100 et est incrémenté de 10 à chaque fois (100, 110, 120, 130, …).

Donner l'implémentation de la classe **Joueur** comportant :

- Un constructeur
- Une méthode **Saisir**
- Une méthode **Afficher**
- Une méthode **MAJScore**
- Une méthode **ChangerEtat**.

EnsJoueurs

L'ensemble des joueurs est stocké dans un vecteur de taille 20 (l'on suppose qu'on a 20 joueurs candidats).

Donner l'implémentation de la classe **EnsJoueurs** qui implémente le vecteur de joueurs, comportant :

- Un constructeur
- Une méthode **Creer** qui crée l'ensemble des joueurs
- Une méthode **Afficher** qui affiche l'ensemble des joueurs
- Une méthode **SelectionnerJoueur** qui sélectionne de manière aléatoire un joueur de la liste.

Remarques :

1. Pour des besoins de tests, il faut prévoir à l'intérieur des classes des constructeurs supplémentaires permettant de remplir par programme les structures nécessaires au déroulement du test afin d'éviter la saisie de données longue et fastidieuse.
2. Il faudra remplir avec des données correctes et sensées.
3. Pour les noms des joueurs, utiliser les lettres de l'alphabet A, B, C, …, Z.
4. S'il y a lieu redéfinir les méthodes ou dériver les classes.

Application à réaliser

Phases du jeu

Le jeu de Question / Réponse à réaliser comprendra trois phases où 4 joueurs s'affrontent autour de questions sur 10 thèmes :

Phase I :

Dans cette phase, le jeu se déroule entre 4 joueurs choisis aléatoirement. Un thème parmi 10 est sélectionné automatiquement du tableau dans un ordre séquentiel (un indicateur du dernier thème sélectionné est mis à jour, après le choix du 10ème thème, on revient au premier).

Une question de niveau facile est sélectionnée pour chacun des joueurs selon une politique Round-Robin (i.e de manière circulaire). Les 4 joueurs répondent à leurs questions séparément, le score est incrémenté de 2 si la réponse donnée est correcte.

Phase II :

Le jeu se déroule entre les trois joueurs gagnants de la phase I. Cette phase propose deux questions de niveau moyen pour chaque joueur (une question par thème choisi).

A ce niveau, les questions porteront sur uniquement 6 thèmes choisis du tableau aléatoirement.

A tour de rôle et de manière alternée, chaque joueur choisit un thème (ensuite un deuxième) qui est supprimé des choix aussitôt sélectionnés.

Une question de niveau moyen (dans le thème choisi) est sélectionnée selon la politique Round-Robin et présentée au joueur.

Le score du joueur qui donne la bonne réponse est incrémenté de 3.

Phase III :

Dans cette phase, le jeu se déroule entre les deux joueurs gagnants de la phase II. Les questions porteront sur trois thèmes choisis par le concepteur du jeu. Le score du joueur donnant la bonne réponse est incrémenté de 5.

Gestion de conflits

Pour chaque joueur un timer régis par un Thread est associé. Il démarre lorsque son joueur obtient la main pour répondre à la question, et s'arrête dès que la réponse est fournie.

En cas d'égalité de scores entre les joueurs à une phase donnée, ils seront départagés grâce aux valeurs des timers. Ainsi, les joueurs ayant été les plus rapides seront ceux qui se qualifient à la phase d'après.

Dans le cas d'égalité, à la fois, des scores et timers, proposer jusqu'à trois questions supplémentaires pour les départager. Après cela, faire une sélection aléatoire pour passer à l'étape suivante.

Description de l'application

L'application doit, de manière générale, permettre de réaliser les actions suivantes :

1. Afficher les 10 thèmes choisis ;
2. Créer une liste de questions pour chaque thème ;
3. Afficher toutes les questions d'un niveau n donné par thème ;
4. Ajouter une question à la liste pour un thème donné ;
5. Supprimer une question de numéro x de la liste pour un thème donné ;
6. Créer le tableau de joueurs et afficher leurs états ;
7. Lancer une partie du jeu avec 4 joueurs choisis en affichant toutes les étapes du déroulement du jeu ;
8. Quitter le jeu ;
9. La gestion de persistance reste ouverte et à chacun de proposer un moyen qui lui convient ;
10. D'autres actions peuvent être ajoutées si nécessaire.

3) Fonctionnalités optionnelles

Le grand jeu

Faire en sorte que l'application permette de considérer 3 groupes de 4 joueurs chacun, choisis dans le vecteur de joueurs.

- Implémenter une partie de jeu pour chaque groupe de 4 joueurs ;
- Récupérer les numéros des trois joueurs gagnants ;
- Implémenter le grand jeu entre les trois joueurs gagnants, la même politique est reprise commençant à la phase II qui comportera trois thèmes au choix et une question par joueur.
- Affichant toutes les étapes du déroulement du jeu.

Implémenter une IA

Utiliser les threads pour jouer le rôle de joueurs et répondre aléatoirement aux questions dans le cadre d'une partie de jeu complète.