IPO – TP 4 : conception objet

https://www.lri.fr/~blsk/IPO/

Objectifs On souhaite écrire un petit programme permettant de créer et de répondre à des QCM, répondant au cahier des charges suivant. Un QCM est un ensemble de questions. Chaque question contient un énoncé et une liste de plusieurs réponses possibles. Les énoncés comme les réponses sont donnés sous la forme d'un texte. Chaque réponse peut être correcte ou incorrecte. Une question peut avoir plusieurs réponses correctes. Le créateur du QCM veillera à ce qu'au moins une réponse par question soit correcte.

Vous allez devoir créer votre programme et vos classes en partant de zéro. Nous vous fournissons un unique fichier `Utils.java`, contenant une méthode qui demande à l'utilisateur d'entrer une valeur entière entre 0 et un entier donné en paramètre et qui renvoie la valeur entrée par l'utilisateur.

Création d'un QCM Avant de se lancer dans le code, prenez une feuille et un crayon : il va falloir planifier. On commence par s'intéresser aux manières de représenter et d'initialiser un QCM. On aura besoin notamment de pouvoir créer un QCM, des questions, des réponses, et d'associer des réponse à une questions, et des question à un QCM.

1. Quelles sont les classes à créer pour répondre au cahier des charges ci-dessus ? Dans quelle classe mettriez-vous votre fonction `main` ?
2. Pour chaque classe, de quels attributs aurons-nous besoin ?
3. Quels paramètres donner au constructeur de chacune des classes ? Aurons-nous besoin de méthodes supplémentaires pour initialiser un QCM ?

Une fois ceci fixé, vous pouvez commencer à coder.

4. Définir les classes que vous avez décrites dans les questions précédentes, et y coder les constructeurs et les méthodes nécessaire à la création d'un QCM.
5. Dans le `main`, définir un QCM composé des deux questions suivantes.
   - À propos des nombres premiers :
     - Le nomre 1 est premier (faux)
     - Il en existe une infinité (vrai)
     - Un nombre pair peut être premier (vrai)
     - Le produit de deux nombres premiers est premier (faux)
   - Quel temps fait-il aujourd'hui ?
     - Nuageux (vrai)
     - Pluvieux (faux)
     - Orageux (faux)
6. Écrire, dans la classe `QCM`, une méthode `isValid` qui vérifie la validité d'un QCM, c'est-à-dire qui vérifie que chaque question a bien au moins une réponse juste.
   Indication : vous pourrez avoir besoin de définir plusieurs méthodes dans plusieurs classes pour cela.

Affichage Les méthodes de cette section permettront d'afficher un QCM. Vous pourrez également vous en servir pour tester le résultat des questions précédentes.

7. Écrire la méthode `toString` pour une réponse. Cette méthode doit renvoyer une chaîne de caractères donnant le texte associé à la réponse, mais ne doit pas indiquer si la réponse est juste ou fausse.
   Exemple :

```
"Le nombre 1 est premier"
```

8. Écrire la méthode `toString` pour une question. Cette méthode doit renvoyer une chaîne de caractères donnant l'énoncé de la question, suivi de chaque réponse possible. Chaque réponse doit être précédée de son numéro (on commencera la numérotation à 1) et deux réponses doivent être séparées par un saut de ligne.

   Exemple :

```
"Quel temps fait-il aujourd'hui ?
   1- Nuageux
   2- Pluvieux
   3- Orageux
"
```

9. Écrire la méthode `toString` pour un QCM. Cette méthode doit renvoyer une chaîne de caractères donnant l'ensemble des questions et des réponses possibles.

Répondre à un QCM Vous allez maintenant étendre les classes précédentes, pour permettre à un utilisateur de répondre à un QCM. On considère pour l'instant que l'utilisateur ne donne qu'une seule réponse par question (et qu'il répond bien à toutes les questions). La réponse de l'utilisateur est considérée comme juste dès lors que la réponse choisie est l'une des réponses correctes.

On utilisera les mêmes classes pour l'énoncé d'un QCM et pour les QCM donnés à chaque utilisateur. On considère que chaque utilisateur répond à sa propre instance du QCM. Il vous faut donc ajouter à vos classes de quoi enregistrer les réponses données par *un* utilisateur.

10. On ajoute dans la classe représentant les questions un attribut indiquant le numéro de réponse sélectionnée. Quel est l'intérêt d'utiliser le type `Integer` plutôt que **`int`** dans ce contexte ?
11. Écrire une méthode dans la classe `Question` qui indique si l'utilisateur a correctement répondu, c'est-à-dire s'il existe une réponse et que cette réponse est effectivement correcte.

    Indication : vous pouvez ajouter une méthode dans une autre classe si nécessaire.
12. Écrire dans la classe `Question` une méthode permettant de poser la question : elle en affiche l'énoncé, attend que l'utilisateur saisisse un entier valide (pour pouvez utiliser la méthode fournie dans `Utils.java`) et sélectionne la réponse correspondante.
13. Écrire une méthode qui permet de répondre à un QCM et affiche le score de l'utilisateur à la fin du questionnaire.
14. Écrire une méthode qui, à partir d'un QCM d'origine, crée une nouvelle instance ave les mêmes questions (mais sans aucune réponse), de sorte qu'un nouvel utilisateur puisse répondre au QCM.

Raffinements Voici quelques idées pour enrichir le système.

15. Associer à chaque question un barème. On peut y compris prévoir des points positifs ou négatifs réponse par réponse. Ajuster le calcul du score et la méthode `isValid`.
16. Permettre à l'utilisateur de revenir à une question précédente pour annuler, modifier ou compléter sa réponse.
17. Permettre à l'utilisateur de sélectionner plusieurs réponses pour une question, et ne compter comme validées que les questions où l'utilisateur a sélectionné toutes les réponses justes et aucune réponse fausse. Pour saisir plusieurs réponses, vous pouvez vous baser sur le mécanisme précédent, ou en imaginer un autre.
18. Permettre à plusieurs utilisateurs de répondre à un questionnaire, mémoriser leurs résultats, puis donner des statistiques sur les performances du groupe.