Questionnaires

(les fichiers mentionnés dans ce document sont sur le portail.)

Un questionnaire est défini par un ensemble de questions. Chaque question est caractérisée par un texte (la question elle-même), la réponse solution et un nombre de points.
Les réponses aux questions peuvent être de différentes natures : numériques, symboliques (textuelles) ou vrai/faux (le questionné ne peut répondre que `vrai` ou `faux`).
Poser un questionnaire consiste à (en commençant par la première question) :

1. poser la question,
2. saisir la réponse du questionné après lui avoir annoncé le type autorisé et en n'acceptant la saisie que lorqu'elle est conforme à ce type (par exemple un nombre si la réponse attendue est numérique),
3. on a alors deux situations : si la réponse donnée est correcte, l'indiquer et augmenter le score du questionné ; si cette réponse n'est pas correcte, annoncer quelle était la réponse solution correcte.
4. passer à la question suivante si il en reste, sinon annoncer le score global.

Voici un exemple de trace possible.

```
Quel est le nom de l'auteur du Seigneur des Anneaux ?
(symbolique) Tolkien
correct (1 point)
Frodo est un Hobbit ?
(vrai/faux) vrai
correct (1 point)
Combien de membres composent la Compagnie de l'Anneau ?
(numerique) neuf
(numerique) 9
correct (2 points)
Gandalf est un humain ?
(vrai/faux) vrai
incorrect, la bonne réponse est faux
En quelle annee est paru le Seigneur des Anneaux ?
(numerique) 1960
incorrect, la bonne réponse est 1954

Vous avez 4 points.
```

Vous pouvez tester ce comportement à l'aide de l'archive exécutable `questionnaire.jar` fournie. Le fichier `question_tolkien.txt` devra être dans le même dossier.

Q 1 . Créez (et codez) les types (interfaces et classes) nécessaires à la gestion de tels questionnaires.

A partir de maintenant on suppose que le choix de conception fait à la question précédente a amené à définir un type `Answer` qui est la racine d'un hiérarchie de types de réponse et que l'on a une classe `Question` possédant un attribut de type `Answer` (notons qu'une hiérarchie de questions aurait pu être envisagée, il suffirait alors d'adapter ce qui suit à cette logique).

Q 2 . On souhaite pouvoir mémoriser les questionnaires dans des fichiers texte. La structure du fichier est définie par des suites de blocs de 3 lignes : la première contient le texte de la question, la seconde la réponse solution et la troisième le nombre de points associés à la question (un entier). Ces données sont donc lues[1] dans un tel fichier sous la forme de chaînes de caractères.

Il faut donc ajouter d'une méthode à la classe `Questionnaire` une méthode qui permet la création d'**objet question** à partir de ces triplets de données. Cette méthode pourrait être de la forme :

```
public Question createQuestion(String text, String solutionAnswerText, String points)
```

Une méthode de `Questionnaire` prenant en paramètre un nom de fichier et réalisant suffisamment d'appels à `createQuestion` doit permettre de "remplir" le questionnaire.

Pour réaliser `createQuestion` vous vous appuierez sur une classe *singleton* `AnswerFactory` disposant d'une "méthode de fabrique"[2] pour créer les différentes réponses à l'aide de la méthode :

[1] Inspirez-vous du source `src/TestIO.java` fourni pour programmer la lecture du fichier.

[2] Dans sa forme la plus simple puisque vous n'éviterez probablement pas l'énumération de `if...then...else if`, ce qui pénalise l'OCP. Pour le corriger il faudrait utiliser le chargement dynamique de classe.

```
public Answer build(String solutionAnswerText)
```

Pour coder cette méthode, on se basera sur une analyse syntaxique de la chaîne de caractères qui représente la réponse solution. Dans cette analyse on testera dans l'ordre les hypothèses suivantes :

- si la chaîne de caractères corrspond à un entier alors la réponse est de type numérique (`Integer.parseInt`);
- si la chaîne correspond `"vrai"` ou à `"faux"` alors la réponse est de type vrai/faux (utiliser les méthodes des `Enum`) ;
- tous les autres cas sont acceptés comme réponse de type textuelle.

Q 3 . On ajoute maintenant un nouveau type de questions pour lesquelles plusieurs réponses (textuelles) sont possibles, les points sont attribués si le questionné fournit l'une d'elles. Le nombre de réponses possibles est annoncé.

Exemple de question :

```
Donnez le nom de l'un des hobbits de la Compagnie de l'Anneau ?
(4 réponses) Pippin
correcte
```

Pour construire un tel objet réponse, lors de l'analyse syntaxique on fera l'hypothèse que dans le fichier de questionnaire les différentes réponses du "solutionAnswerText" sont séparées par le caractère `';'` (on fera donc l'hypothèse que ce caractère ne peut pas apparaître dans une réponse) (cf. `"question_tolkien_2.txt"`).

Faites le nécessaire pour pouvoir gérer ce nouveau type de questions (vous pouvez utiliser un objet `Scanner` pour analyser votre question ou un `StringTokenizer`). Il faudra donc modifier la classe `AnswerFactory`.

Q 4 . On ajoute à nouveau un type de questions : les questions à choix multiples dans lesquelles le questionné choisit sa réponse parmi une liste proposée, mais seule l'une des réponses est la bonne. Pour la saisie seule une réponse parmi les propositions est acceptée.

Exemple de question :

```
Comment s'appelle le poney qui accompagne la compagnie jusqu'à la
Moria ?
(Robert   Bourricot   Bill   Jolly Jumper) Bob
(Robert   Bourricot   Bill   Jolly Jumper) Bill
correcte
```

Pour construire un tel objet réponse, lors de l'analyse syntaxique on fera l'hypothèse que dans le fichier du questionnaire, les différentes possibilités (textuelles) de réponses sont annoncées dans "`solutionAnswerText`", séparées par le caractère `'|'` (on fera donc l'hypothèse que ce caractère ne peut pas apparaître dans une réponse), la première étant "la bonne" (cf. `"question_tolkien_2.txt"`). Evidemment lors de l'affichage il faut faire apparaître les propositions dans un ordre quelconque.

Q 5 . Pour aller plus loin : interface graphique

On souhaite proposer une interface graphique (IHM par la suite) pour les questionnaires. Dans cette IHM, les questions sont affichées les unes après les autres et le questionné peut y répondre dans l'ordre de son choix. Une fois qu'il considère qu'il a fini de répondre, il valide l'ensemble de ses réponses en cliquant sur un bouton.

La présentation des questions diffère en fonction du type de la réponse acceptée par la question. C'est en fait l'IHM de saisie de la réponse qui change : des radio boutons pour les "vrai/faux", un "spinner" numérique pour les "numériques" et un champ de texte pour les "textuelles".

En vous en inspirant (même fortement et éventuellement en réutilisant mais en l'adaptant car elles ne suffisent pas !) les classes que vous pourrez trouver dans `src/quiz/ihm` de `questionnaire-ihm.jar` (exécutable), créez une IHM pour les questionnaires. La classe `Test` en réalise une "démo" qu'il faudra nécessairement compléter ne serait-ce que pour construire l'IHM à partir d'un questionnaire donné et ensuite pour permettre la validation et le décompte des questions répondues. Les sources concernant l'IHM ne compilent pas, c'est normal car il manque des classes... A vous de vous en inspirer et des les adapter. Cependant vous pouvez "exécuter" la classe compilée `quiz.ihm.Test` fournie (dans `classes`) pour avoir un exemple de l'ihm produite par le code fourni.

On peut mettre ici en œuvre autrement le design pattern "factory method" en ajoutant une méthode de fabrique (la "factory") `createMyAnswerPanel` au type `Answer`. Ce sont ici les "sous-types" de `Answer` qui sont chargés de la création des instances d'objets qu'ils utilisent (les `AnswerPanel`).

Dans un premier temps contentez vous de traiter le cas des 3 premiers types de questions. Une fois que cela fonctionne, ajoutez les deux autres.