UE Conception Orientée Objet

Questionnaires

(Pour le TP, des fichiers sont disponibles sur le portail, en particulier des "jar de démonstration".)

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 de type oui/non (le questionné ne peut répondre que `"oui"` ou `"non"`).
On représente les questionnaire par des instances d'une classe `Questionnaire` qui dispose d'une méthode `poser` qui 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 pour cette méthode.

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

Vous avez 4 points.
```

En TP 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 . Donnez un algorithme "détaillé" de la méthode `poser` de la classe `Questionnaire` qui soit conforme à la trace donnée ci-dessus.

Q 2 . On choisit d'adopter le point de vue que ce qui change d'une question à une autre c'est le type de réponse. On a donc un seul type pour les questions, la classe `Question`, mais plusieurs types de réponses (numériques, textuelles, etc.).

En respectant ce point de vue, donnez les diagrammes de classe UML pour les types (interfaces et classes) nécessaires à la gestion de tels questionnaires.

Q 3 . On souhaite pouvoir initialiser les questionnaires à partir d'informations contenues dans des fichiers texte. On ajouterait ainsi à la classe `Questionnaire` une méthode

```
public void initQuestionnaire(String fileName)
```

qui initialise les questions du questionnaire à partir des informations contenues dans le fichier de nom `fileName`.

La structure du fichier est définie par des suites de blocs de 3 lignes (cf. Annexe) : 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[2].

Il faut donc ajouter 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 questionText, String answerText, String points)
```

Pour réaliser `createQuestion`, il est nécessaire de pouvoir créer les objets de type réponse associés à chaque question. On décide donc d'ajouter **à chaque classe de réponse** une méthode

[1] En TP, inspirez-vous du source `src/TestIO.java` fourni pour programmer la lecture du fichier.
[2] L'utilisation de fichier structuré au format XML serait certainement une bonne idée ici

```
public static Answer<?> build(String answerText) throws IllegalArgumentException
```

qui renvoie une instance de la classe concernée si la chaîne de caractères passée en paramètre est acceptable (par exemple elle représente un entier dans le cas des réponses numériques) et lève l'exception indiquée sinon. Il faut maintenant déterminer quelle est la classe de réponse dont il faut appeler la méthode `build`. On décide de déléguer ce travail à une classe *singleton* `AnswerFactory` qui disposera d'une "méthode de fabrique" pour créer les différents objets réponses à l'aide de la méthode :

```
public Answer<?> buildAnswer(String answerText)
```

Q 3.1. Donnez le code des méthodes `build` de chacune des classe de réponse.

Q 3.2. Quelle solution proposez-vous pour mettre en œuvre la méthode `buildAnswer` ?

Q 3.3. Votre proposition respecte-t-elle le principe ouvert-fermé ?

Q 4 . Sachant que si l'on dispose du nom de la classe (avec les paquetages) sous la forme d'une chaîne de caractères, la méthode suivante permet d'appeler la méthode statique de cette classe de nom `"build"` et qui prend un paramètre de type `String`[3] :

```
public Answer<?> buildAnswer(String answerClassName) throws ...{
      // on récupère l'objet Class pour la classe de nom answerClassName
   Class c = class.forName(answerClassName);
      // on récupère pour cette classe la méthode de signature build(String)
   Method method = c.getMethod("build",String.class);
      // on invoque cette méthode, 1er param = null car static
   Answser<?> answer = (Answer<?>) method.invoke(null,texteReponse);
   return answer;
}
```

Faites une proposition qui corrige les éventuels défauts de `buildAnswer` identifiés à la question précédente.

Q 5 . On ajoute maintenant un nouveau type de questions (ou plutôt de réponses) pour lesquelles plusieurs réponses, nécessairement textuelles, sont possibles. Les points sont attribués si le questionné fournit l'une de ces réponses. 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 possibles) Pippin
correct (1 point)
```

On fera l'hypothèse que dans le fichier de questionnaire les différentes réponses possibles sont séparées dans la ligne "answerText" par le caractère `';'` (on fera donc l'hypothèse que ce caractère ne peut pas apparaître dans une réponse) (cf. `"question_tolkien_2.txt"`).

Q 5.1. Faites le nécessaire pour pouvoir gérer ce nouveau type de questions.
Pour la méthode `build` vous pouvez utiliser un objet `Scanner` pour analyser votre question ou un `StringTokenizer`.

Q 5.2. Que faut-il modifier la classe `AnswerFactory`, dans la version de la question 3 ? Celle de la version 4 ?

Q 6 . 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 suggérées 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
correct (3 points)
```

On fera l'hypothèse que dans le fichier du questionnaire, les différentes possibilités (textuelles) de réponses sont annoncées dans la ligne "`answerText`" séparées par le caractère `'|'`, la première étant "la bonne" (cf. `"question_tolkien_2.txt"`). Evidemment lors de l'affichage il faudra faire apparaître les propositions dans un ordre quelconque.

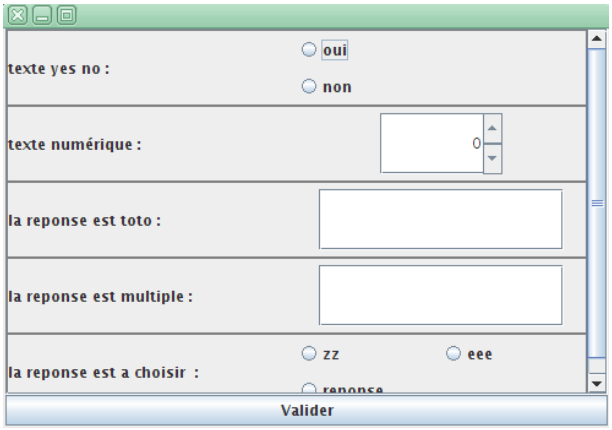
Q 6.1. Faites le nécessaire pour pouvoir gérer ce nouveau type de questions.

Q 6.2. Que faut-il modifier la classe `AnswerFactory`, dans la version de la question 3 ? Celle de la version 4 ?

[3] Cette méthode peut déclencher les exceptions `NoSuchMethodException`, `IllegalAccessException`, `IllegalArgumentException` et `InvocationTargetException`. Voir la documentation de `getMethod()` dans `java.lang.Class` et `invoke()` dans `java.lang.reflect.Method`

Q 7 . 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. Le processus de vérification des réponses proposées et donc le calcul de points est délenché. Une fenêtre annonçant le score est ensuite affichée (voir `javax.swing.JOptionPane : showMessageDialog()`). Cette interface pourrait ressembler à ceci :



L'interface graphique d'un questionnaire est donc constituée

- d'un "panel" (un `JPanel` à décorer par un `JScrollPane`) qui contient les uns après les autres un panel par question,
- du bouton de validation.

Le "panel d'une question" a toujours la même structure :

- une zone de texte pour le texte de la question,
- une zone de saisie de la réponse, cette zone varie uniquement en fonction de la nature de la réponse : des radio boutons pour les "vrai/faux", un "spinner" numérique pour les "numériques", un champ de texte pour les "textuelles", etc.

Pour chaque classe de réponse il faut donc créer la classe d'"`AnswerPanel`" qui correspond (voir par exemple le diagramme en annexe). Il faut ensuite modifier les classes de réponse en y ajoutant une méthode :

```
public AnswerPanel createMyAnswerPanel()
```

qui fournit l'objet `AnswerPanel` approprié à chaque objet de type réponse.

On met ici en œuvre le design pattern "factory method", `createMyAnswerPanel` jouant le rôle de la méthode de fabrique ("factory"). Chaque "sous-type" de réponse est en effet chargé de la création des instances d'objets `AnswerPanel` qui lui correspond.

Programmez cette interface graphique.

Annexe

Exemple de fichier de questions

Cet exemple de fichier ne prend pas en compte la question 4.

```
Quel est le nom de l'auteur du Seigneur des Anneaux ?
Tolkien
1
Frodo est un Hobbit ?
vrai
1
Combien de membres composent la Compagnie de l'Anneau ?
9
2
Gandalf est un humain ?
faux
3
En quelle annee est paru le Seigneur des Anneaux ?
1954
3
Donnez le nom de l'un des hobbits de la Compagnie de l'Anneau ?
Frodo ; Pippin ; Merry ; Sam
1
Comment s'appelle le poney qui accompagne la compagnie jusqu'à la Moria ?
Bill | Bourricot | Robert | Jolly Jumper
3
```

Exemple de diagramme pour les classes d'AnswerPanel

Il n'est pas du tout impératif de respecter ce diagramme.
Pour apprendre à utiliser les différentes classes de composants graphiques, vous pouvez consulter les *Java Tutorials* proposés dans la javadoc des classes de ces commposants.
Par exemple dans la javadoc de la classe `javax.swing.JRadioButton` vous trouvez un lien vers le tutoriel à partir de la phrase *See How to Use Buttons, Check Boxes, and Radio Buttons in The Java Tutorial for further documentation.*. Vous y trouverez des exemples de codes d'utilisation des `JRadioButton` et `ButtonGroup` qui vont avec. Il en est de même pour tous les composants graphiques.