Questionnaires

Pour le projet, 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, textuelles ou de type oui/non (l'utilisateur ne peut répondre que `"oui"` ou `"non"`).
On représente les questionnaires par des instances d'une classe `Questionnaire` qui dispose d'une méthode `askAll` qui consiste pour chaque question à :

1. afficher le texte de la question et la nature de la solution (le « *prompt* ») ;
2. attendre la réponse de l'utilisateur en n'acceptant la saisie que lorsqu'elle est *conforme à la nature de la solution* (par exemple un nombre si la solution est numérique) ;
3. on a alors deux situations : si la réponse donnée est correcte, l'indiquer et augmenter le score de l'utilisateur ; si cette réponse n'est pas correcte, afficher la solution ;
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.
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 (*NumericalAnswer*), textuelles (*TextualAnswer*), etc.).

Q 1 . Donnez un algorithme « détaillé » du processus déclenché par `askAll` qui permet d'interroger l'utilisateur sur un `Questionnaire` et qui soit conforme à la trace donnée ci-dessus.

Ce travail doit vous permettre d'identifier les fonctionnalités nécessaires à la conception de l'application.

Q 2 . En déduire, en respectant le point de vue mentionné précédemment, 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.

La structure du fichier est définie par des suites de blocs de 3 lignes (cf. Annexe) : la première ligne contient le texte de la question, la seconde ligne contient la réponse solution et la troisième le nombre de points associés à la question (un entier). On propose d'utiliser la classe `QuestionFactory` ci-dessous. Elle permet la création d'un questionnaire à partir d'un fichier respectant la forme indiquée. Voici son code :

```
public class QuestionnaireFactory {

  public Question createQuestion(String text, String answer, String points)
                    throws IOException {
    try {
      int nbPoints = Integer.parseInt(points);
      return new Question(text, AnswerFactory.FACTORY.buildAnswer(answer), nbPoints);
    }
    catch(NumberFormatException e) {
```

```
        throw new IOException("bad format");
      }

    }

    public Questionnaire createQuestionnaire(String fileName) throws IOException {
      Questionnaire questionnaire = new Questionnaire();
      File source = new File(fileName);
      BufferedReader in = null;
      try {
        in = new BufferedReader(new FileReader(source));
        String text;
        // read block of 3 lines : text, answer and number of points
        while ((text = in.readLine())!= null) {

          String answer = in.readLine();
          String nbPoints = in.readLine();
          if (answer == null || nbPoints == null) {
            throw new IOException("bad format");
          }
          questionnaire.addQuestion(this.createQuestion(text, answer, nbPoints));
        }
      } catch (FileNotFoundException e) {
        throw new IOException(e);
      }
      finally {
        in.close();
      }
      return questionnaire;
    }
  }
```

Pour réaliser `createQuestion`, il est nécessaire de pouvoir créer les objets de type réponse associés à chaque question. Comme le montre le code ci-dessus, on décide de déléguer ce travail à une classe *singleton* `AnswerFactory` qui disposera d'une méthode de fabrique : `buildAnswer(String)`. Cette méthode permet de créer les objets réponses à partir de la seconde ligne de chaque bloc du fichier.

Proposez un code pour cette méthode `buildAnswer`.

Q 4 . On ajoute maintenant un nouveau type de questions à réponses multiples pour lesquelles plusieurs réponses correctes, nécessairement textuelles, sont possibles. Les points sont attribués si l'utilisateur 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 décide 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. Annexe).

Q 4.1. Que faut-il modifier et/ou ajouter à la première version pour pouvoir gérer ce nouveau type de questions (et ses réponses) ?

Q 4.2. L'architecture de la première version respectait-elle complètement le principe ouvert-fermé ?

Q 4.3. Si ce n'est pas le cas faites une proposition qui permet de respecter au maximum ce principe ?
NB : On peut décider d'imposer que les classes de réponses disposent nécessairement d'un constructeur prenant en paramètre une chaîne de caractères (le texte de la réponse à construire).

Q 5 . On ajoute à nouveau un type de questions : les questions à choix multiples dans lesquelles l'utilisateur choisit sa réponse parmi une liste de propositions parmi lesquelles une seule est la bonne réponse. Les propositions sont annoncées à l'utilisateur et la saisie de l'utilisateur n'est valide que si c'est l'une des propositions suggérées.

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. Annexe). Evidemment lors de l'affichage il faudra faire apparaître les propositions dans un ordre quelconque.

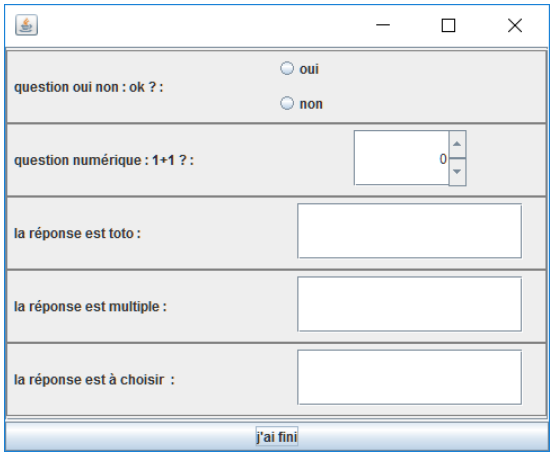
Q 5.1. Que faut-il modifier et/ou ajouter à la seconde version pour pouvoir gérer ce nouveau type de questions (et ses réponses) ?

Q 5.2. La seconde version respectait-elle mieux le principe ouvert-fermé ?

Q 6 . Comment écrire un test unitaire pour vérifier que la contrainte *"attendre la réponse de l'utilisateur en n'acceptant la saisie que lorsqu'elle est* conforme à la nature de la solution*"* est bien implémentée ?

Que faut-il prévoir dans la conception pour qu'un tel test soit possible ?

Interface graphique (partie « Projet »)
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 avec leur zone de réponse et l'utilisateur 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 alors déclenché. Une fenêtre annonçant le score est ensuite affichée[1]. Les « réponses correctes » peuvent éventuellement également être alors présentées.
L'IHM pour la saisie des réponses 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 » est créé à partir d'un objet question et 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. On peut ensuite :

- soit modifier les classes de réponse en y ajoutant une méthode :

  ```
  public AnswerPanel createMyAnswerPanel()
  ```

  qui fournit l'objet `AnswerPanel` approprié à chaque objet de type de réponse.
- soit mettre en place une solution qui ne crée par de dépendance entre les classes d'«`Answer`» et leurs «`AnswerPanel`» (voir le TD casse-briques).

A vous de choisir l'un de ces solutions.

Q 7 . Programmez une application qui reprend l'ensemble des fonctionnalités présentées dans ce sujet.

Il doit être possible d'utiliser les questionnaires en mode texte ou via l'interface graphique. Vous proposerez au moins un autre fichier de questionnaire que ceux fournis.

[1] voir `javax.swing.JOptionPane : showMessageDialog()`

Compléments

- En étudiant les possibilités offertes par la classe `java.util.Properties` faites en sorte que l'on puisse changer facilement la langue utilisée des questionnaires en mode texte (pas la langue du texte des questions ni des réponses) (les « prompts» et autres messages). On doit donc sans changer le code des classes des questions ou réponses pouvoir obtenir l'exécution suivante :

```
Combien de membres composent la Compagnie de l'Anneau ?
(numerical) neuf
(numerical) 9
correct (2 points)
Gandalf est un humain ?
(yes/no) yes
wrong, the right answer was : non
Donnez le nom de l'un des hobbits de la Compagnie de l'Anneau ?
(4 possibles answers) Pippin
correct (1 point)

You get 3 points.
```

- Pour analyser la ligne réponse du fichier pour les questions à réponses multiples ou à choix multiples `buildAnswer` vous pouvez utiliser un objet `Scanner` ou `StringTokenizer`.

- Pour l'IHM jetez un œil aux classes de composants graphiques `JRadioButton`, `JSlider`, `ButtonGroup`, etc. Pour apprendre à les utiliser, 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 les autres composants graphiques.

Annexe

Exemple de fichier questionnaire

Les questions sont par 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). Les deux derniers blocs correspondent aux questions 3 et 4.

Cet exemple de fichier ne prend pas en compte des modifications qui pourraient être apportées par la question 3.3.

```
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
```