Université de Nice-Sophia Antipolis
E.S.I.N.S.A. 1re année

Vendredi 12 decembre 2003

---

# Projet de Java

# Un jeu de Quizz

---

***Avant de commencer:*** ***Tout ce qui suit (la structure du programme, la description des classes présentées en exemple, les noms des variables, etc...) n'est pas imposé! C'est un guide pour ceux qui n'arrivent pas à commencer.***
***La qualité des commentaires, avec notamment la présence des antécédents et des conséquents avant chaque méthode, les noms donnés aux variables, l'emploi à bon escient des majuscules et la bonne indentation rentreront pour une part importante dans l'apréciation du travail.***
***Ce projet doit permettre de montrer votre autonomie et votre compréhension tant dans la conception du programme que dans sa réalisation.***
***Enfin, si les codes de plusieurs projets se trouvent être identiques, tous les projets concernés seront immédiatement sanctionnés par un zéro.***

---

## 1 Le Quizz

Le but de ce projet est de programmer un quizz en java. Le jeu se déroule en un certain nombre de tours. A chaque tour, une question et posée au joueur qui doit proposer une réponse.

### 1.1 Algorithme du jeu

L'algorithme repose sur une boucle dépendant du nombre de tours et se déroule en plusieurs étapes:

1. Le joueur choisit le niveau de difficulté du jeu
2. Le joueur choisit le nombre de tours
3. Tant que (nombre de tours restants est positif)
   (a) L'ordinateur pose une question tirée au hasard parmi un lot de questions pré-saisies
   (b) Le joueur saisit une réponse au clavier

(c) La réponse du joueur est comparée à la réponse de l'ordinateur
(d) La pertinence de la réponse est évaluée
(e) Le score obtenu à cette question est affiché

4. Le score final est affiché

## 1.2 Exemple de l'exécution du jeu

On vous présente un exemple de ce que pourrait être l'affichage de l'exécution du jeu:

```
Niveau de difficulté : 60.0%.

Qui est l'auteur des << Misérables >> ?
Hugo
Réponse : Victor Hugo.
Pertinence : 90.0% ; score : 4

Qui a mis sa culotte à l'envers ?
le roi dagobert
Réponse : Le roi Dagobert.
Pertinence : 30.0% ; score : 4

Score final : 4.

Process Main finished
```

## 1.3 Point à développer: Le calcul de la pertinence

On souhaite calculer une pertinence, exprimée en pourcentage, des réponses fournies par le joueur. Un niveau de difficulté valide ensuite la réponse si elle est suffisamment précise pour permettre de marquer des points. Pour cela, chaque question possède en plus de sa réponse, un ensemble de mots-clés permettant de donner un pourcentage de pertinence de la réponse du joueur.

Par exemple, à la question "Qui a écrit « Les Misérables »?", qui a pour réponse officielle "Victor Hugo.", on associe les mots-clés "Victor" de poids 3 et "Hugo" de poids 7.

De cette manière, si l'utilisateur saisit comme réponse "Hugo", la pertinence de cette réponse utilisateur sera de 90%, à comparer au niveau de difficulté du jeu (il est fixé dans l'exemple à 60%).

Il est bien entendu obligatoire de disposer d'au moins un mot-clé par réponse (sans quoi on ne pourrait jamais marquer) et que tous les mots-clés apparaissent dans la réponse officielle.

Afin de permettre une gestion plus fine des réponses possibles, on peut souhaiter qu'un mot-clé apporte à lui seul 100% de pertinence. Considérez par exemple la question "Quelle est l'unité de mesure de la quantité d'information?". En fait, il existe deux unités de mesure : la réponse est "Le bit ou le Shannon (Sh).". Dans ce cas, on propose bien les deux mots-clés "bit" et "Shannon" avec des poids de 10, et on fixe un poids maximal à 10 aussi. Ainsi, la réponse "Shannon" aura 100% de pertinence.

Le calcul de la pertinence d'une réponse vis-à-vis d'un mot-clé attendu peut

encore être raffiné en gérant les problèmes de majuscules, accents, voire même des fautes d'orthographe pour les plus courageux (n'hésitez pas à être ingénieux quand au calcul de cette pertinence...).

### 1.4 Structure du programme

Chacune de ces étapes peut être effectuée par une classe. On pourra par exemple définir:

- une classe *Quizz* qui présentera l'interface (textuelle) entre le joueur et la machine
- une classe *Jeu* qui gèrera les questions, le niveau de difficulté, le score,...
- une classe *Question* contiendra la "question" et sa *Réponse*
- une classe *Réponse* qui associera des mots-clé prédéfinis à chaque réponse dans l'ordinateur pour permettre de déterminer la validité de la réponse du joueur. Cette classe pourra aussi lier un mot-clé avec son importance (poids) par rapport à la réponse.
- une classe *motCle* qui permettra de tester la présence de chacun des mots-clés dans la réponse du joueur et d'évaluer un score pour le mots-clés courant

## 2 Exemple de canevas

- **La classe *Quizz***

```
/*
 * Quizz.java
 */

/**
 * La classe permettant d'exécuter le jeu. Une partie est finie quand
 * l'utilisateur interromp le programme.
 */
public class Quizz {
    /** Le seuil de difficulté par défaut en %. */
    private static final double DIFFICULTE_PAR_DEFAUT = 60;

    /** Le seuil de difficulté de la partie en %. */
    private double niveau;

    /** Score obtenu au cours de la partie. */
    private int score;
```

- **La classe *Jeu***

```
/*
 * Jeu.java
 */

import java.util.Random;
```

```
/**
 * La classe Jeu initialise un ensemble de questions/réponses et
 * interagit avec l'utilisateur.
 */
public class Jeu {
    /** L'ensemble des questions du jeu. */
    private Question[] questions;

    /** Un générateur de nombres aléatoires. */
    private static Random aleas = new Random();
```

– **La classe *Question***

```
/*
 * Question.java
 */

/**
 * La classe Question associe un intitulé de question et une réponse
 * attendue.
 */
public class Question {
    /** La question du quizz. */
    private String question;

    /** La réponse attendue. */
    private Reponse reponse;
```

– **La classe *Réponse***

```
/*
 * Reponse.java
 */

/**
 * La classe Reponse contient une liste de mots-clés pondérés.
 * Un poids maximal différent de la somme des poids des mots-clés
 * permet des alternatives dans les réponses (mais la pertinence est
 * toujours majorée par 100).
 * @invar le poids maximal est inférieur ou égal à la somme des poids
 * des mots-clés.
 */
public class Reponse {
    /** La liste des mots-clés. */
    private MotCle[] motsCles;

    /** Total de la pondération. */
    private int poidsMax;
```

```
    /** Le texte complet de la meilleure réponse. */
    private String texte;
```

– **La classe *MotCle***

```
/*
 * MotCle.java
 */

/**
 * La classe MotCle réunit un mot-clé d'une Reponse et une valeur
 * de pertinence du mot-clé.
 *
 * @invar poids > 0
 */
public class MotCle {
    /** Le mot-clé en question. */
    private String mot;

    /** Le poids du mot-clé dans la réponse. */
    private int poids;
```

# 3 Diagramme des classes

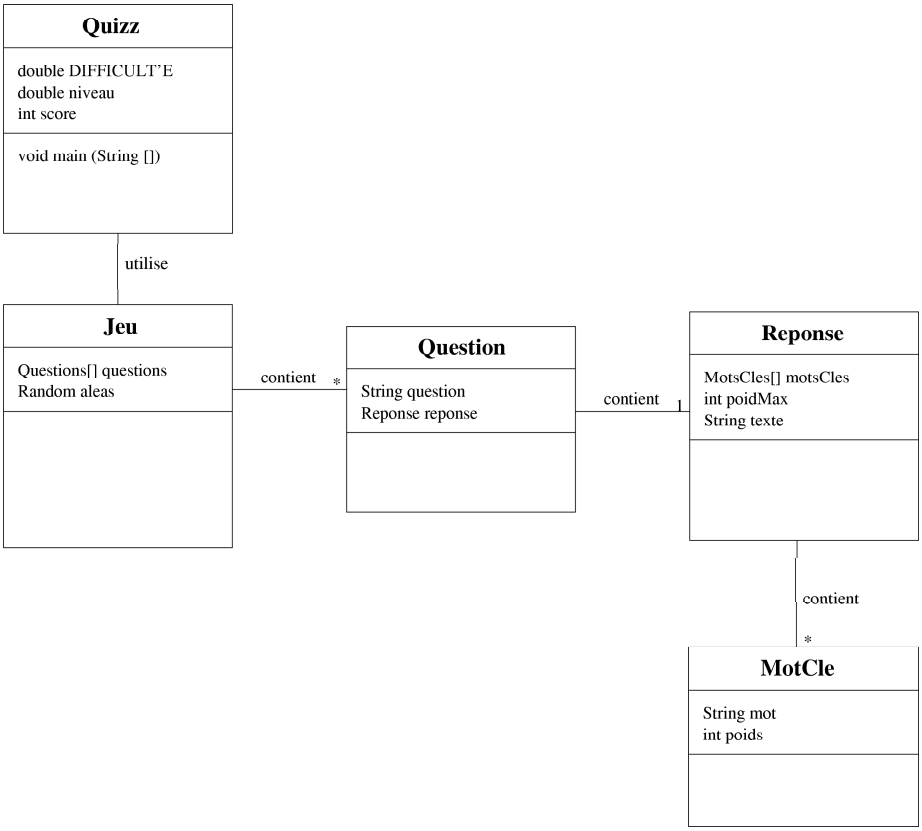


FIG. 1 – *Diagramme des classes*

# Annexe

Nous vous rappelons que la javadoc est très utile si vous avez besoin d'informations sur des classes ou des méthodes que vous ne connaissez pas (comme la classe *Random*, par exemple).
La javadoc se trouve sur le site de Sun:
`http://java.sun.com/j2se/1.4.2/docs/api/index.html.`
Localement, vous pouvez la trouver sur:
`\\Maestro\local\jdk\`