Jeu Questions/Réponses

Rapport de projet

https://github.com/nasslamenace/ProjetJava

Nathan Fontaine
Alexandra Gonçalves
Nassim Guettat
Judicael Kamwa

Mme Melekhova
Groupe E03
Promo L3 2020

Sommaire

Introduction

Cette année, une fois n'est pas coutume, en restant dans le cadre de nos études en cycle d'ingénieur du numérique à l'Efrei, il nous a été demandé de réaliser un projet afin de pouvoir appliquer et approfondir les connaissances théoriques que nous avons pu acquérir tout au long des cours de JAVA 2.

Ainsi le projet consiste en un quizz, une suite de plusieurs questions, qui a lieu entre plusieurs joueurs, désignant à la fin de celui-ci un unique gagnant. Le jeu est divisé en trois phases distinctes, regroupant chacune plusieurs questions de types Questions/Réponses, Vrai/Faux ou QCM, de difficultés multiples (facile, moyen, difficile), et ayant un dizaine de thèmes (Histoire, Science, Info, etc).

Ainsi il est demandé que le joueur puisse répondre à toutes ces questions, sans commettre d'erreurs, sans quoi il sera éliminé de la partie et ainsi de suite, le tout comprenant une gestion de conflits, en cas d'égalité des scores notamment.

Autre la fonctionnalité du quizz principal, les joueurs doivent également pouvoir accéder aux thèmes disponibles, ajouter des questions, afficher les questions en fonction d'un thème donné, etc.

Au cours de ce rapport nous verrons donc comment nous avons implémenté ce programme, quels en ont été les apports et nous finirons sur les obstacles que nous avons pu rencontrer tout le long du projet.

I. Analyse Fonctionnelle Générale

Nous allons dans cette partie, présenter de façon générale le concept de notre application et son organisation.

1. Principales données traitées

Notre programme englobe des données qui interagissent entre elles. Les principales classes de notre programme sont les suivantes :

- Question
- Thèmes
- ListQuestions
- Phase
- Joueurs
- EnsJoueurs
- Etat
- Niveau
- Typequestion

2. Principales fonctionnalités du programme et leur organisation

a. Fonctionnalités principales

Les principales fonctionnalités sur lesquelles repose notre application sont les suivantes :

- Afficher les 10 thèmes choisis
- Créer une liste de questions pour chaque thème
- Afficher toutes les questions d'un niveau n donné par thème
- Ajouter une question à la liste pour un thème donné
- Supprimer une question de numéro x de la liste pour un thème donné
- Créer le tableau de joueurs et afficher leurs états
- Lancer une partie du jeu avec 4 joueurs choisis en affichant toutes les étapes du déroulement du jeu
- Quitter le jeu
- La gestion de persistance reste ouverte et à chacun de proposer un moyen qui lui convient

Toutefois, nos différentes fonctionnalités ont une structure bien établie dans notre programme.

b. Organisation des fonctionnalités

Notre programme renferme des fonctionnalités qui décrivent de façon ludique l'ensemble des opérations effectuées et qui permettent au différents utilisateurs d'avoir accès à la création et modification des différentes données du jeu.

- ***Afficher les 10 thèmes choisis***

  Ici nous affichons l'ensemble des dix thèmes que renferme notre application. En plus de cela,en bas de la fenêtre, nous indiquons de même le thème actuel (position de l'indicateur).

- ***Créer une liste de questions pour chaque thème***

  Dans cette branche, nous avons fait des séries de question pour chacun des thèmes.

- ***Afficher toutes les questions d'un niveau n donné par thème***

  Ici le programme affiche toutes les questions d'un niveau (les niveaux allant d'un à trois) en fonction des thèmes.

- ***Ajouter une question à la liste pour un thème donné***

  Cette fonctionnalité donne la possibilité aux joueurs d'ajouter une ou plusieurs questions au quizz, ceci dans le but d'augmenter le nombre de questions à proposer lors d'une partie de jeu. Il s'agit donc de permettre à l'utilisateur de remplir un formulaire permettant d'ajouter une nouvelle question au quizz, par thème, type de question et niveau.

- ***Supprimer une question de numéro x de la liste pour un thème donné***

  Cette fonctionnalité donne la possibilité aux joueurs de supprimer une ou plusieurs questions de la liste des questions pour chacun des thèmes.

- ***Créer le tableau de joueurs et afficher leurs états***

  Ici on crée un tableau contenant les différents joueurs qui prennent part à la partie et on affiche l'état (le niveau et le nombre de questions trouvées) de chacun.

- ***Lancer une partie du jeu avec 4 joueurs choisis en affichant toutes les étapes du déroulement du jeu***

  Cette fonctionnalité permet de démarrer une partie de jeu contenant 4 joueurs et d'afficher tout le déroulement du jeu avec ses différentes phases, en éliminant un joueur par phase, suivant son score.

- **Quitter le jeu**

Ici, un joueur a la possibilité de quitter la partie, d'abandonner le jeu en cours. Ceci permet donc de relancer une autre partie ou arrêter totalement de jouer.

- ***La gestion de persistance reste ouverte et à chacun de proposer un moyen qui lui convient***

Cette fonctionnalité donne la possibilité aux joueurs de quitter le programme et revenir plus tard continuer au niveau où ils se trouvaient. Dans cette partie les données liées aux parties précédentes des joueurs restent conservées.

3. Conception et choix

Durant l'implémentation de notre programme, nous avons eu à faire des choix importants. Dans le but de nous faciliter les tâches lors du développement de notre programme.

- **Choix du langage JAVA**

En plus du fait que nous sommes sur un projet lié au cours de JAVA, le JAVA est un langage qui propose contrairement aux autres langages orienté objets plusieurs facilités, plusieurs modules permettant de concevoir de façon simple nos différentes fonctionnalités.

- **Choix de la classe swing pour nos interfaces**

Pour l'affichage d'une interface graphique nous avons opté pour l'implémentation de trois classes, MyButton, MyLabel, MyPanel, afin de définir un minimum de composants et caractéristiques graphiques, que nous avons par la suite repris dans les autres classes. Après avoir initialisé les classes requises, avec un début de code, nous avons créé un menu permettant aux joueurs de sélectionner les fonctionnalités qu'ils souhaitent utiliser.

- **Choix des fichiers pour contenir nos questions**

Dans le but de faciliter la gestion et le stockage des questions par thème et par niveau, nous avons décidé de choisir la lecture et écriture dans des fichiers .txt.

II. Analyse Fonctionnelle Détaillée

Nous allons apporter dans cette partie, plus de détails concernant l'implémentation de notre programme, en particulier les objets manipulés et l'interaction qui existe entre ces objets. Nous ferons également une description du fonctionnement de notre programme.

1. Description de l'application et phases du jeu

Notre programme consiste à développer une application pour effectuer un Quizz. L'application permet de manière générale de mettre plusieurs joueurs en compétition sur un ensemble de questions données, les questions étant réparties par thèmes et types. Une partie se fait avec 4 joueurs.

L'application affiche toutes les étapes de déroulement du jeu. Les joueurs sont éliminées graduellement, au fur et à mesure que les phases progressent ; ainsi à la **phase1** tous les 4 joueurs sont soumis à des questions de niveau facile. Après comptabilisation de leurs scores, les 3 premiers joueurs ayant eu les meilleurs scores sont sélectionnés pour la **phase2** où ils sont soumis une fois de plus à des questions (en fonction du thème de chacun) mais ici de niveau moyen. De même, à la fin de cette phase, les deux premiers sont sélectionnés pour passer au niveau suivant, la **phase3**. Les deux finalistes sont soumis à des séries de questions et le meilleur scores est le gagnant de la partie.

L'application donne également la possibilité de connaître le meilleur gagnant parmi tous les joueurs. Un joueur peut également ajouter des questions ou supprimer des questions du Quizz.

2. Description fonctionnelle

- **TypeQuestion**

Cette classe implémente l'interface **Serializable**. Elle contient les types de questions de notre jeu. Ainsi, elle renferme 5 méthodes parmi lesquelles la méthode **getquestion(),** qui retourne le contenu d'une question donnée, la méthode **setquestion(String question)**, qui permet d'entrer une nouvelle question. Nous avons trois types de questions : les questions de type **Vrai/Faux**, les **Questions à Choix Multiples** et des questions de type **Réponses courtes,** décrites par une variable qui indique si la réponse est bonne ou pas.

- **Question**

Cette classe est une classe fille de la classe TypeQuestion elle implémente également l'interface **Serializable**. Elle contient une méthode **afficher()** qui affiche le niveau de la question ainsi que son énoncé et une méthode **saisir()** permettant de saisir une fonction de type T donnée.

- **Thèmes**

Cette classe contient l'ensemble des thèmes, stockés dans un tableau de chaînes de caractères. Dans cette classe, on a implémenté différentes méthodes, par exemple permettant de changer de thème grâce à la méthode **ModifierTheme()**.Il y a, de même, la possibilité de sélectionner un thème avec la méthode **selectionnerTheme()** et d'afficher un thème avec la méthode **afficher().**

- **ListQuestions**

Elle dispose d'un ensemble de questions dont le nombre et le type sont variables puis stocke l'ensemble des questions relatives à un thème dans une liste chaînée de questions. Grâce aux méthodes **afficherListe(), ajouterQuestion()** et **supprimerQuestion(),** elle donne respectivement la possibilité aux joueurs d'afficher la liste des questions pour un thème donné, d'ajouter une question et de supprimer une question.

- **Phase**

Cette classe est une interface. Les classes **phase1, phase2** et **phase3** implémentent cette interface. Elle possède deux méthodes, **selectionnerjoueur()** permettant de sélectionner un joueur de façon aléatoire et **phaseDeJeu()** qui renvoie la phase (**phase1**, **phase2** ou **phase3**) du jeu à laquelle on se trouve.

- **Joueurs**

Cette classe permet la gestion des joueurs. Elle décrit le nom, le score, le numéro et l'état du joueur dans le jeu, que l'on peut afficher grâce à sa méthode **afficher()**. Cette classe contient également la méthode **changerEtat()**, permettant de changer l'état du joueur en fonction de son score, et la méthode **MAJscore()** qui calcule le score du joueur .

- **Ensjoueurs**

Cette classe permet de créer une liste de joueurs. Elle possède un vecteur de taille 20 dans lequel sont stockés l'ensemble des joueurs. La méthode **selectionnerjoueur ()** de cette classe permet de sélectionner un joueur de manière aléatoire. Elle contient aussi une méthode **afficher()** qui affiche l'ensemble des joueurs du tableau.

- **Etat**

De type Énumération, elle permet de dire si le joueur a été **sélectionné,** s'il est **gagnant, super gagnant, éliminé ou en attente.**

- **Niveau**

Cette classe est également de type Énumération et elle permet la gestion des niveaux des questions, qui peuvent être **Facile, Moyen ou Difficile**.

3. Implémentation des fonctionnalités

Premières classes et affichage

Tout d'abord, nous avons démarré ce projet en implémentant les classes demandées et les plus essentielles au quizz, telles que Questions, les différentes Phase, ou encore ListQuestions. Ensuite, nous avons créé trois classes, MyButton, MyLabel, MyPanel, afin de définir un minimum de composants et caractéristiques graphiques, que nous avons par la suite repris dans les autres classes. Après avoir initialisé les classes requises, avec un début de code, nous avons créé un menu permettant aux joueurs de sélectionner les fonctionnalités qu'ils souhaitent utiliser, comme suit ci-dessous :

Afin de gérer les différents composants et d'une manière générale l'affichage, nous avons utilisé CardLayout.

```
private CardLayout mainLayout = new CardLayout();
private MyPanel menuContainer = new MyPanel();
private MyPanel wholeContainer = new MyPanel();
private MyPanel mainContainer = new MyPanel();
private MyPanel questionContainer = new MyPanel();
```

Afficher les thèmes

Concernant la première fonctionnalité disponible, qui consiste à afficher les thèmes du jeu, on fait appel à un panel où l'on fait appel à la méthode afficher de la classe Thème et qui consiste à parcourir la liste de 10 thèmes et à l'afficher, tout en communicant le currentTheme à l'aide d'un indicateur qui est positionné sur l'index de ce dernier.

```
String[] liste = new String[10];

for(int i = 0; i < liste.length; i++)
        liste[i] = themes.get(i);


container.add(new JList(liste), BorderLayout.CENTER);

container.add(new MyLabel("Le theme actuel est le theme :" + themes.get(currentTheme)), BorderLayout.SOUTH);

return container;
```

Ajouter une question

Afin de commencer la gestion et le stockage des questions par thème et niveau, nous avons créé une classe FileManager qui repose sur le principe de lecture et écriture dans des fichiers particuliers, puis regroupe les questions trouvées dans un ArrayList. Pour plus de simplicité, nous avons décidé d'écrire dans des fichiers .txt.

Pour la lecture des questions (prend le thème en argument) :

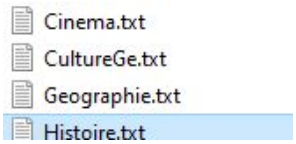
```
        ArrayList<Question> questionList = new ArrayList<Question>();
try
{
    FileInputStream fis = new FileInputStream(theme + ".txt");
    ObjectInputStream ois = new ObjectInputStream(fis);

    questionList = (ArrayList) ois.readObject();

    ois.close();
    fis.close();
```

Pour l'écriture des questions (prend en argument un tableau de questions et le thème) :

```
FileOutputStream fos = new FileOutputStream(theme + ".txt");
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(questionList);
oos.close();
fos.close();
```



Fichiers textes regroupant les questions par thème, créés et utilisés par le programme

Ensuite, pour ajouter une question, nous avons affiché un panel regroupant les composants nécessaires à remplir par le joueur. Tout d'abord une comboBox afin de choisir le type de question (VF, etc), une comboBox listant les thèmes disponibles, puis des JTextField afin de permettre au joueurs de rentrer les informations de la question. Une fois le formulaire validé, on récupère toutes les informations, puis en fonction d'abord du type de question puis du niveau de la question, on crée une nouvelle Question qui est ajoutée à un Arraylist qui contient déjà les questions du fichier texte selon le thème choisi, puis on fait appel à la méthode updateQuestion de FileManager afin de réécrire le nouveau tableau dans le fichier texte correspondant.

```
ArrayList<Question> questions = FileManager.retrieveQuestions((String)themesBox.getSelectedItem());
```

```
Question q = null;
```

```
q = new Question(Niveau.facile, (String)themesBox.getSelectedItem(), new QCM(enonceTf.getText(), choix, reponseITf.getText()));
```

```
questions.add(q);
```

```
FileManager.updateQuestion(questions, (String)themesBox.getSelectedItem());
```

Afficher les questions

Pour cette fonctionnalité, qui consiste à afficher les questions du jeu en fonction du thème et du niveau, on fait appel à la classe ListeQuestionPan et qui consiste, de manière graphique, à retrouver les questions voulues et les retranscrire sur un JTable.

Pour le JTable des questions faciles par exemple :

```
facileModel = new QuestionTableModel(myTheme.getMesQuestions().selectionnerQuestions(Niveau.facile));

facileQuestionTable = new MyTable(facileModel);
facileQuestionTable.setCellSelectionEnabled(false);
MyTable.applyDesign(facileQuestionTable);

TableComponent facilebuttonRenderer = new TableComponent();
facileQuestionTable.getColumn("Afficher").setCellRenderer(facilebuttonRenderer);
facileQuestionTable.addMouseListener(new JTableButtonMouseListener(facileQuestionTable));
```

Afficher les joueurs

La fonctionnalité suivante a comme utilité d'afficher la liste de l'ensemble des joueurs participants, grâce à l'utilisation de la classe EnsJoueurs et de sa fonction d'affichage "*afficher*".

```
playersPan.removeAll();
playersPan.add(EnsJoueurs.afficher());

playersPan.revalidate();
playersPan.repaint();



mainLayout.show(mainContainer, "Players");
wholeContainer.add(cancelContainer, BorderLayout.NORTH);
revalidate();
repaint();
```

Lancer le jeu

Cette dernière fonctionnalité ayant pour but de lancer le jeu, va dans un premier temps sélectionner les joueurs, pour pouvoir par la suite lancé la première phase de jeu pour ces joueurs choisis, en affichant la fenêtre de jeu.

```
phase1.selectionnerJoueurs();
phase1.phaseDeJeu();

mainLayout.show(mainContainer, "Jeu");
wholeContainer.add(cancelContainer, BorderLayout.NORTH);

revalidate();
repaint();
```

III. Limites, difficultés du projet et graphes

1. Limites du projet

Notre programme, bien qu'il soit finalisé, présente des limites. Nous allons donc présenter ici même ces limites d'un point de vu singulier.

- **L'utilisation des fichiers au détriment d'une base de données**

Dans notre programme, nous avons utilisé des fichiers .txt pour stocker nos différentes questions. L'utilisation des fichiers peut ainsi rendre notre programme redondant. L'utilisation d'une base de données pour stocker nos

différentes questions devrait rendre le programme plus optimal, permettre d'avoir des relations entre les différentes questions et leurs niveaux. Il se pose également le problème du temps d'accès aux données. En effet, celui-ci semble être plus long pour un fichier que pour une base de données.

- **La possibilité pour un joueur d'ajouter ou supprimer des questions**

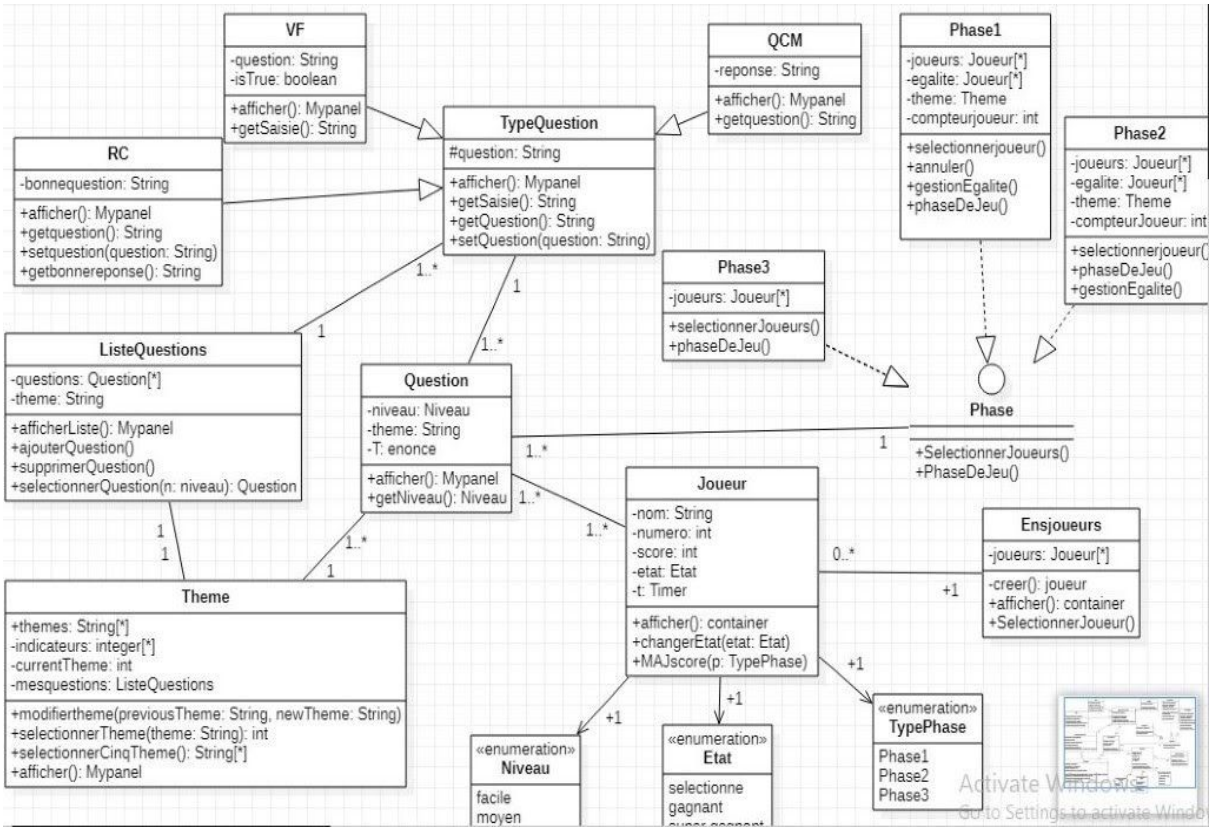
L'application donne la possibilité aux joueurs d'ajouter ou de supprimer une question du questionnaire. Cette pratique n'est pas optimale car aucun contrôle n'est fait sur les questions ajoutées par le joueur. De même, nous pouvons, dans certains cas, être confrontés aux problèmes de redondance des données si le joueur ajoute une question qui figurait déjà dans le fichier de questions.

2. Difficultés rencontrées lors du développement

Tout au long de l'implémentation de notre application, nous avons rencontré de nombreuses difficultés. La première difficulté était liée à la communication et au partage du code. Etant donné que nous ne programmons pas tous de la même façon, nous avons eu des difficultés à comprendre nos codes mutuellement, à partager nos codes. L'un des problèmes majeurs était le partage du code à partir de la plateforme GitHub car c'était la première fois pour bon nombre d'entre nous à utiliser cette plateforme, qui n'est pas particulièrement instinctive et demande une utilisation particulière.

3. Diagramme des classes du projet

Afin de donner un visuel des interactions entre les différentes classes que nous avons pu implémenter, nous avons décidé de réaliser le diagramme des classes suivant :


VF
-question: String
-isTrue: boolean
+afficher(): Mypanel
+getSaisie(): String
QCM
-reponse: String
+afficher(): Mypanel
+getquestion(): String
Phase1
-joueurs: Joueur[*]
-egalite: Joueur[*]
-theme: Theme
-compteurjoueur: int
+selectionnerjoueur()
+annuler()
+gestionEgalite()
+phaseDeJeu()
Phase2
-joueurs: Joueur[*]
-egalite: Joueur[*]
-theme: Theme
-compteurJoueur: int
+selectionnerjoueur()
+phaseDeJeu()
+gestionEgalite()
RC
-bonnequestion: String
+afficher(): Mypanel
+getquestion(): String
+setquestion(question: String)
+getbonnereponse(): String
TypeQuestion
#question: String
+afficher(): Mypanel
+getSaisie(): String
+getQuestion(): String
+setQuestion(question: String)
Phase3
-joueurs: Joueur[*]
+selectionnerJoueurs()
+phaseDeJeu()
ListeQuestions
-questions: Question[*]
-theme: String
+afficherListe(): Mypanel
+ajouterQuestion()
+supprimerQuestion()
+selectionnerQuestion(n: niveau): Question
Question
-niveau: Niveau
-theme: String
-T: enonce
+afficher(): Mypanel
+getNiveau(): Niveau
Phase
+SelectionnerJoueurs()
+PhaseDeJeu()
Joueur
-nom: String
-numero: int
-score: int
-etat: Etat
-t: Timer
+afficher(): container
+changerEtat(etat: Etat)
+MAJscore(p: TypePhase)
Ensjoueurs
-joueurs: Joueur[*]
-creer(): joueur
+afficher(): container
+SelectionnerJoueur()
Theme
+themes: String[*]
-indicateurs: integer[*]
-currentTheme: int
-mesquestions: ListeQuestions
+modifiertheme(previousTheme: String, newTheme: String)
+selectionnerTheme(theme: String): int
+selectionnerCinqTheme(): String[*]
+afficher(): Mypanel
«enumeration»
Niveau
facile
moyen
«enumeration»
Etat
selectionne
gagnant
«enumeration»
TypePhase
Phase1
Phase2
Phase3
1
1..*
0..*
+1

Conclusion

En somme de ce projet riche en apprentissage et en ressources, nous avons pu mettre en pratique les enseignements reçus durant les cours de JAVA2, mais aussi pu connaître et savoir implémenter les concepts d'analyse du langage UML.

Malgré les difficultés rencontrées au début lors de la conception, nous avons réussi à fournir une conception claire. L'application **JEU QUIZZ**, a été un projet capital pour nous, car il nous a permis de penser comme des futurs ingénieurs, qui seront amenés à résoudre les problèmes applicatifs de la société à l'aide du numérique. Elle nous a également permis de communiquer mutuellement entre nous à travers les réseaux sociaux sans toutefois se rencontrer en présentiel.

Il est vrai qu'il reste un bon nombre de choses à améliorer et c'est un défi nouveau qui s'ouvre à nous car cette application peut devenir bien plus qu'un simple jeu de QUIZZ. Nous sommes convaincus que nous avons pu apporter une solution aux différents problèmes qui nous ont été proposés. Nous restons à l'écoute et disponibles pour d'autres projets à venir, tout en étant au service de la société dans le but d'éduquer.

Annexes

Lien github :

https://github.com/nasslamenace/ProjetJava/?fbclid=IwAR1RLLbjqHmqAGzGRGu3Ywt3uB5mvYwGHYnPuESm_OL4_E9JPYTw6Snuztc