

Jeu Questions/Réponses



CRETEUR Samuel
DOE Victoria
DUCHEMIN Stéphane
GUENOUN Déborah
PECH Lucas

Promo 2022
Groupes 03 et 07

SOMMAIRE

Introduction.....page 2

Nouvelles méthodes.....page 3

Interface graphique GUI.....page 5

Administration et serialization.....page 13

Conclusion.....page 27

Introduction

Le but de ce projet était de nous donner l'opportunité, par groupe de 4 ou 5 élèves, de mettre en pratique les notions que nous avons pu aborder durant le cours de Java-2. Pour cela, nous devons réaliser un jeu de questions/réponses en 3 phases. Le jeu devait être entièrement programmé en Java en suivant des consignes précises sur les classes de base que devait contenir le projet, ainsi que certaines méthodes de ces classes.

Après avoir lu attentivement le sujet et avoir discuté en équipe de la compréhension de ce sujet. Nous avons pu réfléchir et mettre par écrit ce que nous devions réaliser.

Comme précisé ci-dessus, le jeu se déroule en 3 phases : lors de la première phase, 4 joueurs sont sélectionnés pour commencer une partie. Un thème parmi les dix est choisi au hasard et chaque joueur répond à 3 questions (aléatoirement un QCM, un Vrai ou Faux et une question à réponse courte) de niveau facile sur ce thème. A chaque bonne réponse les joueurs reçoivent 2 points. A la fin de la phase 1, on choisit les 3 joueurs avec le plus haut score. Si des joueurs sont à égalité alors leur chronomètre est pris en compte. Si encore une fois il y a égalité, alors nous reposons des questions pour les départager.

Ainsi, les 3 meilleurs joueurs passent à la phase 2. Dans cette phase 6 thèmes (différents de celui de la phase 1) sont proposés. Chaque joueur, à tour de rôle va choisir un thème sur lequel une question de niveau moyen lui sera posée. Une fois que les 3 joueurs ont sélectionné et répondu à une question sur leur thème choisis, chaque joueurs doit à nouveau choisir un thème et répondre à une question. A chaque bonne réponse, 3 points sont attribués. De la même façon qu'à la fin de la phase 1, les 2 meilleurs scores accèdent à la phase 3. Si des égalité surviennent, on utilise le même procédé qu'expliqué dans la phase 1.

Lors de la phase 3, les 2 finalistes s'affrontent sur les 3 thèmes restants avec des question d'un niveau difficile. Chaque question vaut 5 points. A la fin des questions, le joueur ayant obtenu le plus haut score gagne le jeu. Si des égalité surviennent, on utilise le même procédé qu'expliqué dans la phase 1.

Nous vous présenterons donc dans ce rapport ce que nous avons dû mettre en place et réaliser en plus des consignes données dans l'énoncé pour créer ce jeu fonctionnel. Nous verrons dans un premier temps les nouvelles méthodes utilisées, puis la mise en place de l'interface GUI et nous terminerons par la partie administration et serialization.

Nouvelles méthodes

Classe Question :

`toString()`: Ici, cette méthode nous permet pour chaque question d'afficher son numéro, sa difficulté et le contenu de cette question.

`getEnonce()`: Cette méthode renvoie le contenu de la question.

`getNumero()`: Cette méthode renvoie le numéro de la question.

`getDifficulte()`: Cette méthode renvoie la difficulté de la question.

Classe Thèmes :

`setto0()`: Dans l'énoncé, il était demandé qu'un indicateur soit incrémenté pour qu'une fois un thème sélectionné, on ne puisse plus le choisir. Il nous fallait donc une méthode pour mettre à 0 cet indicateur.

`removeTheme(String theme)`: Cette méthode a été créée pour supprimer un thème de la liste des thèmes existants.

Classe ListeQuestions :

`getQuestion(int difficulte)`: Cette méthode en fonction de la difficulté qu'on lui donne 1, 2 ou 3 nous renvoie les questions de cette difficulté.

`getQuestionByNb(int numero)`: Cette méthode nous permet d'afficher une question grâce à son numéro.

`supprimerQuestionByNb(int numero)`: L'application de manière générale doit permettre de supprimer une question. Si nous choisissons de supprimer une fonction avec son numéro, nous pouvons le faire grâce à cette méthode. Cette méthode a aussi une sécurité : le jeu a besoin au minimum de 90 questions pour tourner ainsi, l'action se bloque s'il ne reste plus que 90 questions dans la liste de questions.

Classe Joueur :

`getScore()`: Chaque joueur possède un score pour l'afficher, il nous fallait donc aller chercher ce score.

`getNumero()`: Chaque joueur est enregistré dans le vecteur par un numéro ainsi si l'on veut ajouter le score du joueur au nom correspondant cette méthode est utile.

`getEtat()`: Puisque le jeu est divisé en 3 phases, les joueurs peuvent être en jeu ou en attente. Cette méthode nous permet d'afficher dans l'interface graphique l'état dans lequel est le joueur.

`getNom()`: Cette méthode nous renvoie le nom du joueur.

`getEgalite()`: Grâce à cette méthode nous pouvons savoir s'il y a une égalité ou non.

`toString()`: Cette Méthode nous affiche le numéro du joueur son score et on chronomètre, très utile pour passer d'une phase à l'autre.

`getTimer()`: Cette méthode nous permet d'afficher sur l'interface graphique le chronomètre du joueur.

`compareTo(Joueur joueur)`: Cette méthode, lors d'une égalité, nous permet de comparer les temps des joueurs et de trouver celui ayant répondu le plus rapidement.

Classe EnsJoueurs :

`getVectorJoueurs()`: Cette méthode nous donne les 4 joueurs qui ont été sélectionnés pour jouer une partie.

`removeJoueur(int num)`: En saisissant le numéro d'un joueur, cette méthode nous permet de le supprimer.

`JoueursSelectionnes(Joueur joueur)`: Une fois que au hasard un joueur est sélectionné pour jouer une partie, on l'ajoute au vectorJoueurs grâce à cette méthode.

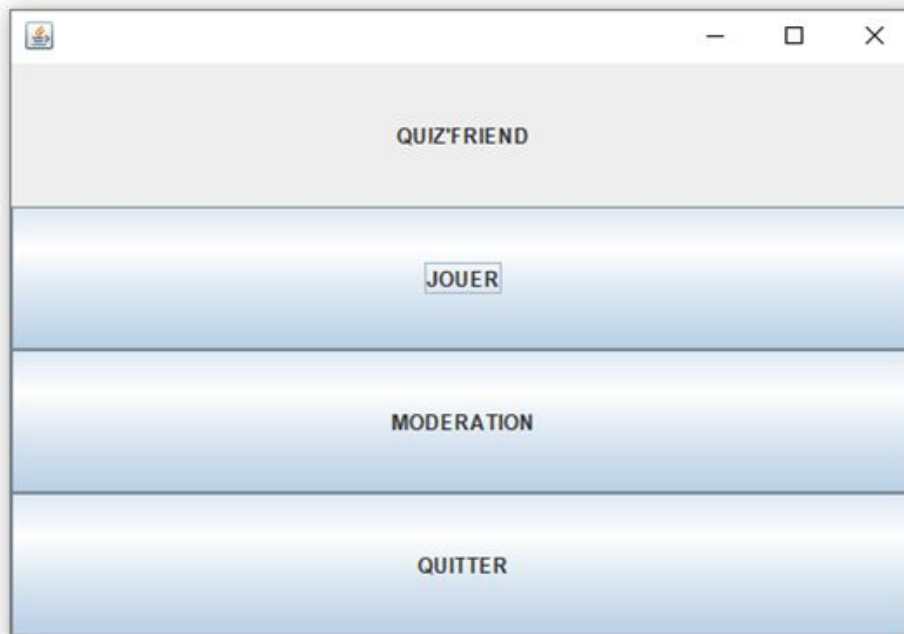
`resetValeurEgalite()`:

`listeNom()`: Cette méthode nous permet d'afficher la liste des noms des joueurs.

Interface graphique GUI

Menu principal

Lorsque l'utilisateur lance notre programme, ce menu s'affiche. Dans cette partie nous discuterons de ce qu'il se passe si le joueur décide de jouer. Nous aborderons la partie modération par la suite.



Déroulement du jeu

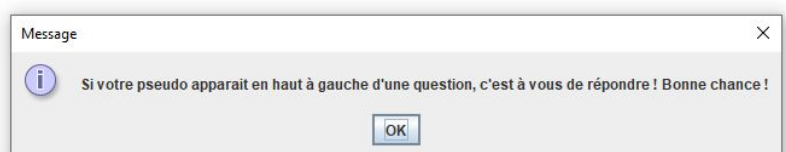
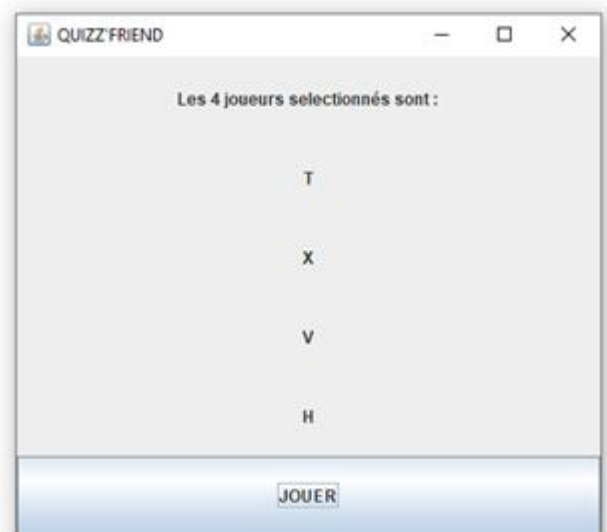
a) Sélection des joueurs

La première fenêtre à s'ouvrir après que l'utilisateur ait cliqué sur jouer est la fenêtre que vous pouvez voir sur la droite.

Cette fenêtre affiche 4 pseudos de joueurs sélectionnés parmi 20. Les 4 personnes voulant faire le quizz doivent donc choisir quel sera leur pseudo.

Imaginons Pierre, Paul, Henry et Nicolas qui veulent jouer : chacun choisira une lettre qui lui correspondra.

Par exemple Paul sera X, Henry sera V, Nicolas sera T et Pierre sera H. Ainsi lorsque ce sera au tour de X de joueur, Paul devra répondre.



b) Phase 1

Une fois les pseudos répartis entre les joueurs, dès que l'utilisateur clique sur jouer, la phase 1 commence. Pour cette phase, l'un des 10 thèmes est sélectionné aléatoirement et les questions des 4 joueurs porteront sur ce thème.

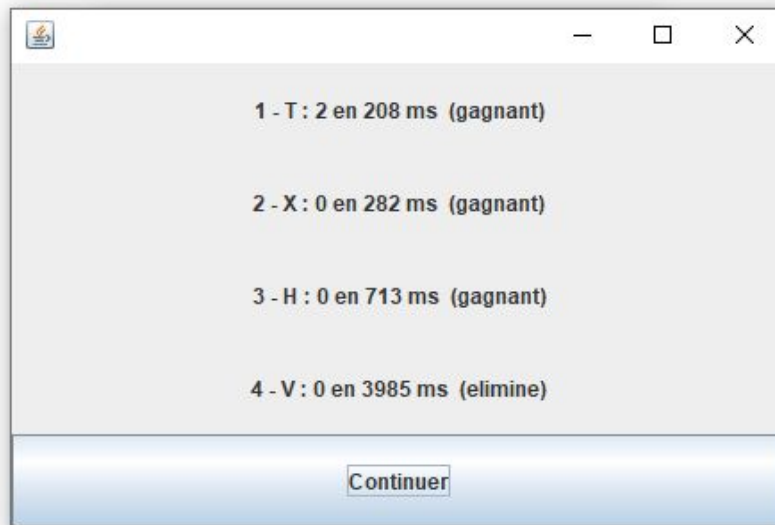
Les questions de chaque joueur sont elles aussi sélectionnées au hasard, mais seront toutes de niveau 1 (facile) et porteront sur le thème sélectionné. Une bonne réponse vaut 2 points dans cette phase.

Ainsi c'est d'abord au tour du joueur V (on trouve le pseudo du joueur qui doit répondre en haut à gauche de la fenêtre, à côté du thème) de répondre à une question sélectionnée aléatoirement parmi toutes les questions faciles de ce thème. Dès lors qu'il va soumettre une réponse en appuyant sur un bouton, une nouvelle question aléatoire sur le thème de la phase apparaîtra pour le joueur suivant. Et ainsi de suite jusqu'à ce que les 4 joueurs aient répondu à une question facile portant sur le thème sélectionné au début de la phase.

Pour faire cela, nous comptons le nombre de tours de la partie, chaque fois qu'un joueur soumet une réponse, le nombre de tour augmente et nous ouvrons une question pour le joueur suivant, forcément différente de celle qui vient de tomber. Par exemple un QCM, ne peut être suivi que d'une question VF ou RC et pareil pour les autres types.



Une fois le nombre de tours arrivé à 4, nous affichons les résultats de cette première phase : les joueurs sont classés par score et temps (les joueurs sont des objets implémentant l'interface *comparable*), ce qui permet d'éliminer le dernier et d'élire les 3 autres premier comme gagnant, qui pourront participer à la phase suivante.



b) Phase 2

Comme évoqué précédemment, seuls les trois premiers de la phase 1 sont sélectionnés pour la phase 2. Ces trois joueurs vont devoir chacun leur tour, choisir 1 thème et répondre à une question de niveau 2 (moyen) sur ce thème. Chaque joueur aura l'occasion de choisir 2 thèmes et répondra donc à 2 questions, chaque bonne réponse rapportant 3 points. Cependant dès qu'un thème est choisi, les joueurs suivant ne pourront plus le choisir (le bouton pour ce thème sera désactivé). Cette idée désavantage le dernier à choisir, notre programme permet donc que l'ordre de passage soit déterminé par le résultat de la phase 1. Les 6 thèmes sélectionnés sont forcément différent du thème de la phase 1.



Le joueur T doit maintenant choisir 1 thème sur lequel il veut répondre à une question, imaginons qu'il choisisse le thème Monde, une question aléatoire de niveau moyen sur ce thème lui sera posée.

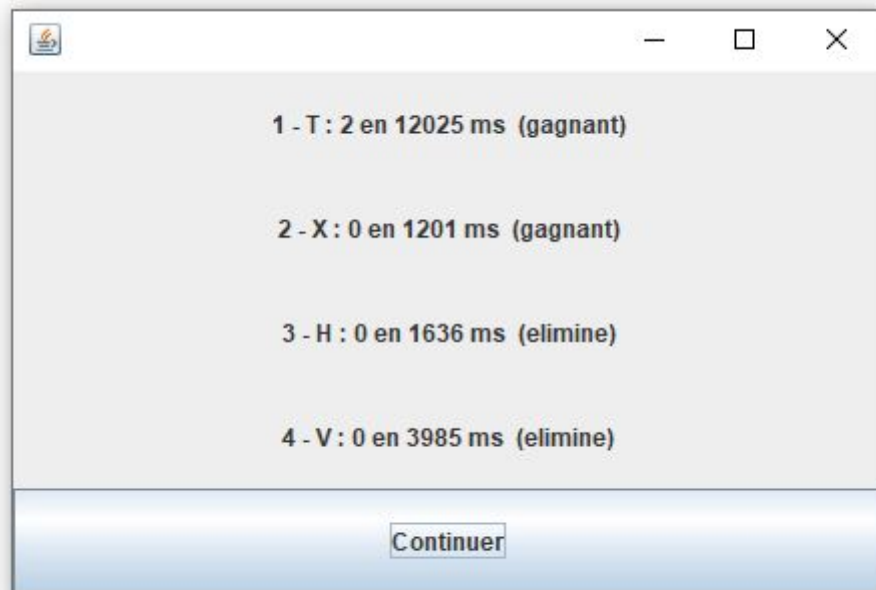


Lorsque le joueur suivant devra choisir, le thème Monde ne sera plus disponible.



Pour faire en sorte que le joueur éliminé précédemment ne joue pas, nous utilisons un numéro de tour qui sera modulo 3, permettant de savoir que le joueur 1 doit jouer au tour 1 et 4 par exemple, le numéro 4 n'existant pas modulo 4, le joueur 4 ne jouera jamais. Puis lorsque nous verrons que tous les thèmes ont été choisis, nous arrêterons cette phase pour afficher les résultats.

Ici seul les deux premiers seront considérés gagnants et pourront accéder à la phase suivante. Ce classement ne prend évidemment pas en compte les joueurs éliminés précédemment qui sont considérés comme plus petits par la fonction de comparaison.



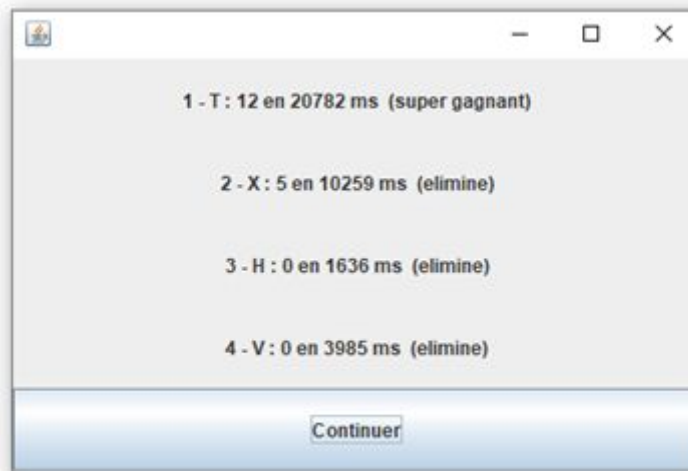
c) Phase 3

Les deux gagnants de la phase 2 accèdent à la phase 3. Cette phase posera 3 questions aux deux finalistes. Ces 3 questions portant sur les 3 derniers thèmes pas encore tombés lors de ce quiz (nous avons 10 thèmes et seuls 7 ont été utilisés pendant les 2 premières phases). Les questions seront cette fois de niveau 3 (difficile) et chacune rapportera 5 points.

Encore une fois, pour s'assurer que seuls les 2 premiers puissent jouer en finale, nous allons utiliser l'astuce du modulo. Nous allons compter les tours et en appliquant un modulo 2 au nombre de tours, nous saurons qui doit jouer entre le joueur classé à la première place et le joueur à la deuxième place. (Le vecteur de joueur est classé par score/timer, le meilleur est en première place tandis que le moins bon est dernier). Accéder à l'index 1 et 2 permet donc d'accéder aux joueurs 1 et 2 qui sont finalistes.

Lorsque 6 tours ont eu lieu (3 questions pour 2 joueurs, 1 tour par question), le classement final est affiché.

Ici T est notre super gagnant avec 12 points obtenus en 20 782 millisecondes.



d) Gestion des égalités

Comme demandé dans le sujet, notre programme permet de gérer les situations d'égalité. Si à la fin d'une phase, deux joueurs ont le même score et le même temps, alors une égalité est détectée. Nous allons donc créer un nouvel ensemble de joueur avec les joueurs à égalité, et poser des questions à cet ensemble de joueur.

Dans un premier temps nous allons poser une question d'un même thème aléatoire à chaque joueur sans les chronométrer, nous utiliserons seulement les points pour départager nos joueurs.

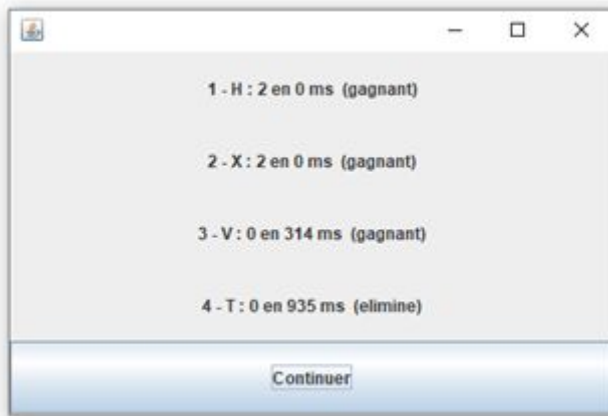
Si après que l'ensemble des joueurs à égalité aient répondues à la question, l'égalité persiste, alors nous recommençons. Nous pouvons recommencer jusqu'à ce que 3 questions faciles aient été posées à chaque joueur (sur 3 thèmes différents). Si après cela l'égalité est toujours présente alors nous gardons le classement tel quel : les joueurs sont de base dans un ordre aléatoire s'ils sont à égalité.

Une fois l'égalité résolue, le jeu reprend son cours et les score des joueurs n'ont pas bougés, seul leur place a potentiellement changée.

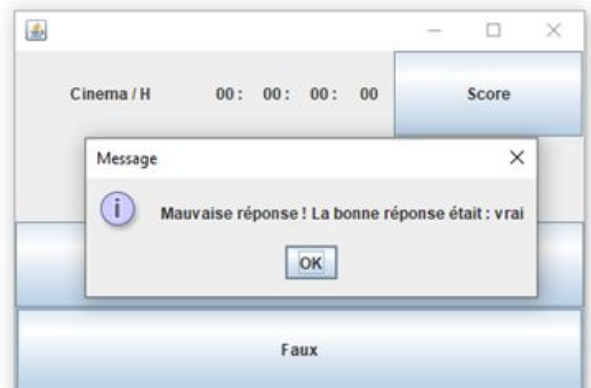
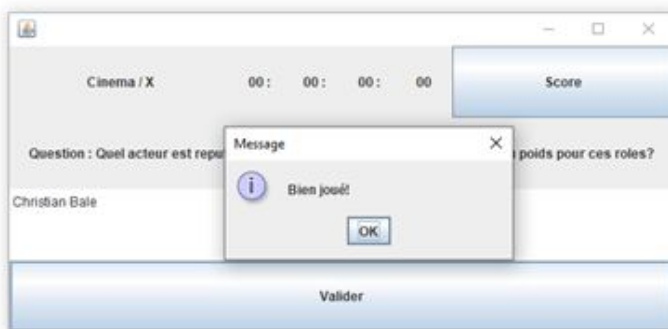
Pour ne pas modifier le classement officiel pour régler une égalité nous faisons en sorte que les points gagnés en répondant aux questions posées pour départager les joueurs, n'aient aucun impact sur le classement officiel. Pour cela nous faisons gagner 0.01 point par bonne réponse ce qui permet de départager 2 personnes qui auraient par exemple 5 points, mais ne permet en aucun cas de dépasser un joueur à 6 points (au maximum l'égalité ajoute 0.03 à un joueur, 0.03 points qui sont d'ailleurs retirés après que l'égalité ait été gérée).

Vous trouverez un exemple de gestion d'une égalité sur la page suivante :

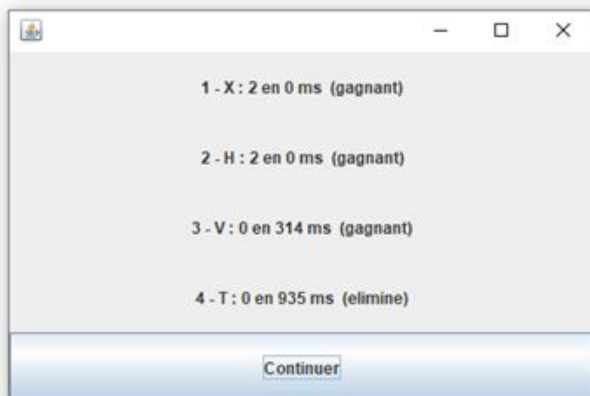
1. Une égalité est détectée : H a été placé en première place de manière aléatoire alors qu'il a le même point et le même temps que X. On doit départager X et H.



2. Chaque joueur répond à une question. X trouve la bonne réponse, H ne trouve pas.



3. X passe en première position car il a gagné lors de la gestion des égalités, le jeu continu.



A noter que si X et H n'avaient pas été départagés par l'étape 2, on peut répéter cette étape encore 2 fois, autrement H restera premier.

Modération

Après avoir vu le déroulement du jeu, passons désormais à la partie modération. Le joueur a en effet accès à cette partie dès le lancement de notre jeu en cliquant sur "MODERATION".



La fenêtre ci-dessous s'affiche alors permettant à l'utilisateur différents choix que l'on développera un à un :

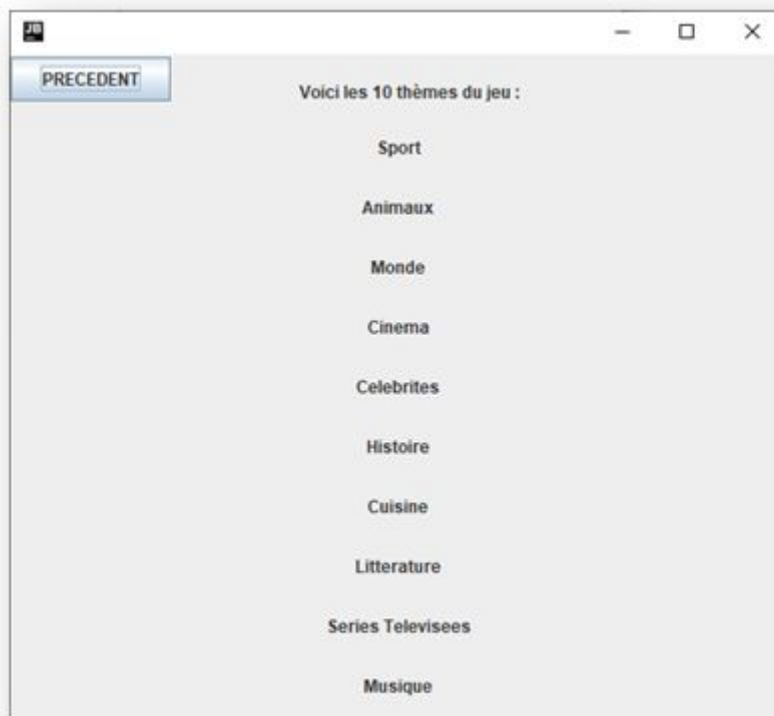
- Afficher la liste des thèmes de notre jeu
- Afficher les questions de notre jeu par thème
- Ajouter une question (RC, VF ou QCM)
- Et enfin supprimer une question dans la limite des questions nécessaires au bon fonctionnement du jeu



a) **Afficher Themes :**



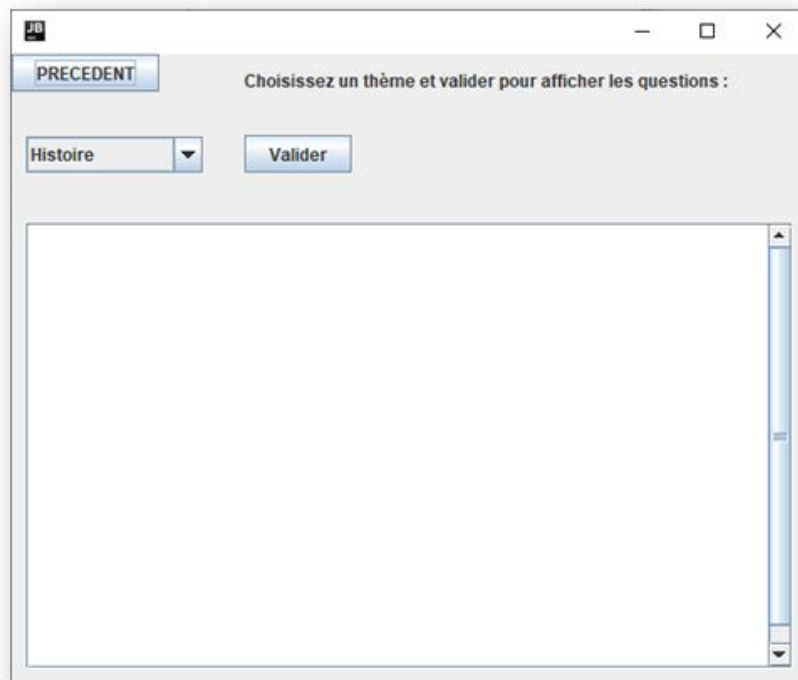
En cliquant sur *AFFICHER THEMES*, la fenêtre ci-dessous s'affiche :



Cette partie permet donc simplement à l'utilisateur de prendre connaissance de tous les thèmes faisant partie du jeu. Pour permettre à cette fenêtre d'afficher les 10 thèmes corrects du jeu, on crée un objet thème dès la première fenêtre de notre jeu que l'on envoie en paramètres aux différentes fenêtres. Ainsi, une fois dans la fenêtre "Afficher theme", nous récupérons la liste de thème dans l'objet envoyé et nous utilisons la méthode "`SelectThemes()`" pour sélectionner les 10 thèmes du jeu et nommer nos labels correspondant aux 10 lignes de notre fenêtre "Afficher themes".

b) **Afficher question par thème:**

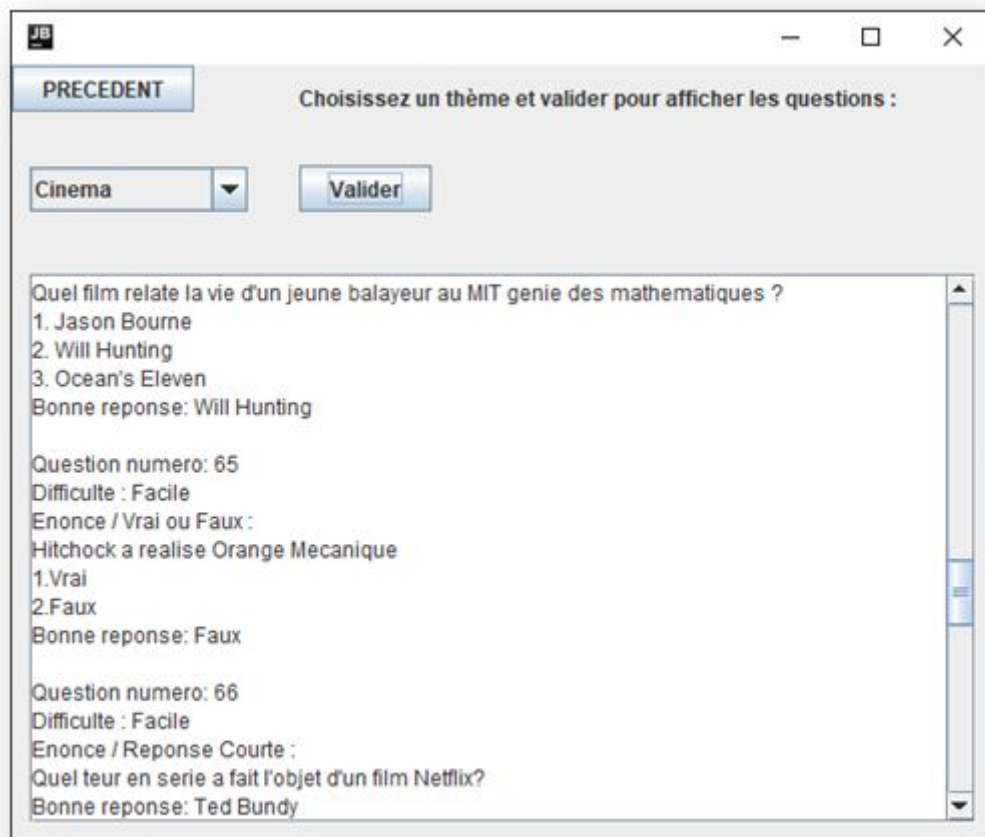




Cette fenêtre s'affiche alors. L'utilisateur a accès à la liste de thème depuis une liste déroulante grâce à la même méthode utilisée pour la fenêtre "Afficher thème". L'utilisateur va alors après avoir sélectionné un thème valider. A la suite de cela, toutes les questions relatives à ce thème vont être affichées dans la JTextArea au milieu de la fenêtre. Ceci est possible grâce à l'envoi d'un objet "ListeQuestion" depuis la première fenêtre. En effet, l'action suivante va être performed au moment où l'utilisateur appuiera sur valider :

l'objet ListeQuestion envoyé va être utilisé pour récupérer toutes les questions. Etant donné que celles-ci sont classées selon leur thème, on récupère celles dont le thème correspond à celui sélectionné par l'utilisateur.

Ensuite, on utilise la méthode `ToString()` des questions et on les ajoute une à une à ce JTextArea. Un exemple après avoir appuyé sur valider en sélectionnant les questions relatifs au thème du Cinema :



Etant donné que nous possédons un grand nombre de questions (plus de 90), une liste déroulante était nécessaire.

c) Ajouter question



La fenêtre qui s'affiche après que l'utilisateur ait sélectionné "Ajouter Question" est l'une des plus complexes. En effet la fenêtre suivante s'affiche:

The screenshot shows a Java Swing window with a title bar containing a small icon and standard window controls (minimize, maximize, close). Inside the window, there is a button labeled 'PRECEDENT' in the top-left corner. Below it, a text label reads 'Choisissez un niveau de difficulté, un thème, et un type de question pour en ajouter une'. The main area of the window contains three vertically stacked dropdown menus. The first dropdown is labeled 'Facile', the second 'Histoire', and the third 'QCM'. Each dropdown has a small downward-pointing arrow on its right side. At the bottom of the window, there is a button labeled 'Valider'.

Il est alors demandé à l'utilisateur de choisir pour la question qu'il souhaite ajouter : Un niveau de difficulté, un thème et le type de la question (QCM, Vrai/Faux ou Réponse courte). On pose ces questions à l'utilisateur afin de pouvoir envoyer les informations nécessaires à la fenêtre suivante. Enfin, on a besoin de savoir le type de la question pour afficher la bonne fenêtre à l'utilisateur. Pour chaque type de question ajoutée une fenêtre différente sera affichée. Développons les chacune :

1) AJOUT QCM

This screenshot is identical to the one above, showing the 'Ajouter Question' window. However, a red circle is drawn around the 'QCM' option in the third dropdown menu, indicating it is the selected option for this section.

The image shows a Java Swing window titled "JB". It has a standard title bar with minimize, maximize, and close buttons. Inside the window, there is a button labeled "PRECEDENT" in the top left. Below it, the text "Ecrivez la question et les 3 réponses possibles :" is centered. The main area contains four labels: "Question:", "Réponse 1:", "Réponse 2:", and "Réponse 3:", each followed by a large text input field. At the bottom, there is a label "Entrez le n° de la bonne réponse et validez:" followed by a small dropdown menu currently showing "1", and a button labeled "Valider".

On arrive alors sur cette fenêtre. On demande ainsi à l'utilisateur d'entrer sa question, puis les 3 réponses possibles et d'indiquer le numéro de la réponse correcte.

Ensuite, grâce aux informations envoyées depuis la fenêtre précédente et celles que l'on récupère avec les champs remplis par l'utilisateur, on crée la question avec le type QCM, le thème, la difficulté et le numéro de la question. Puis, étant donné qu'on utilise la sérialisation comme principe pour la gestion de la persistance de notre jeu afin que les questions ajoutées subsistent malgré la fermeture de notre application on fait les manipulations suivantes :

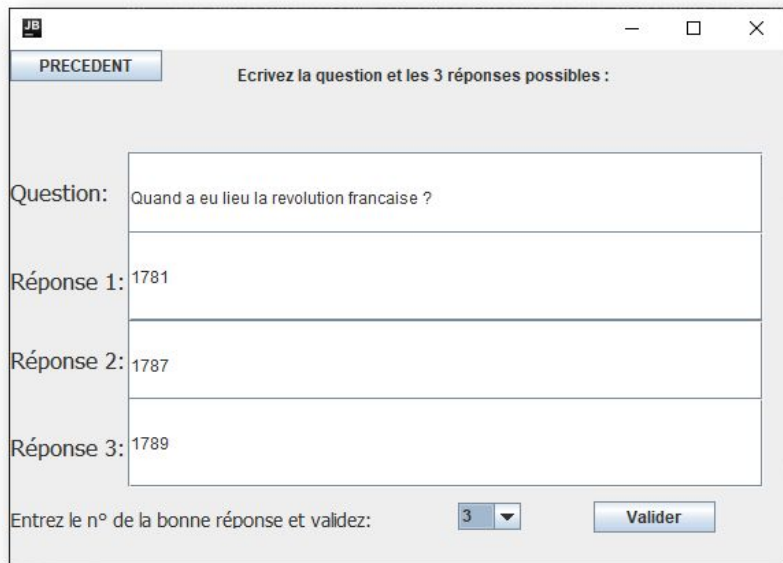
- On ajoute la question QCM créée à la liste de question totale.

- On retire la liste de question de la hashmap correspondant au thème sélectionné étant donné que notre question créée n'a pas été ajoutée à cette liste au moment où elle a été entrée.

- Puis, on remet la liste de questions avec le thème sélectionné par l'utilisateur dans notre hashmap de questions dès lors que la nouvelle question y a été ajoutée.

- Enfin, on sérialise la liste totale des questions pour permettre la persistance comme expliqué précédemment. De cette manière, à chaque ouverture du jeu il suffit de désérialiser pour récupérer le fichier mis à jour dans modération pour notre jeu.

Voici un exemple à la fenêtre ci-dessous :



The dialog box has a title bar with 'JB' and standard window controls. It features a 'PRECEDENT' button in the top left. The main text reads 'Ecrivez la question et les 3 réponses possibles :'. Below this, there are three text input fields. The first is labeled 'Question:' and contains the text 'Quand a eu lieu la revolution francaise ?'. The second is labeled 'Réponse 1:' and contains '1781'. The third is labeled 'Réponse 2:' and contains '1787'. Below these, there is a fourth field labeled 'Réponse 3:' containing '1789'. At the bottom, there is a label 'Entrez le n° de la bonne réponse et validez:' followed by a dropdown menu showing '3' and a 'Valider' button.

JB

PRECEDENT

Ecrivez la question et les 3 réponses possibles :

Question: Quand a eu lieu la revolution francaise ?

Réponse 1: 1781

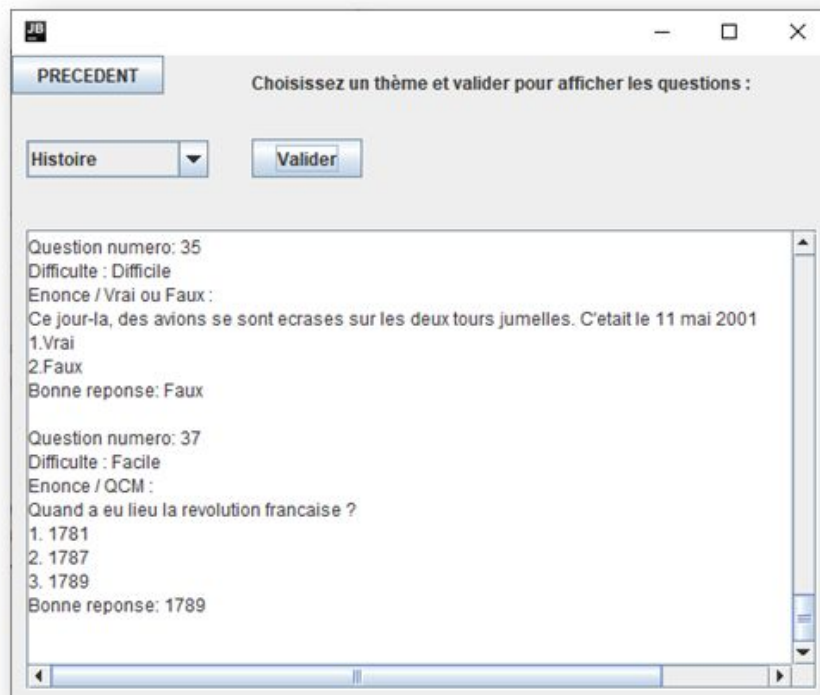
Réponse 2: 1787

Réponse 3: 1789

Entrez le n° de la bonne réponse et validez: 3 Valider



Ensuite, quand on va revient au menu précédemment expliqué pour voir notre liste de question par thème et qu'on sélectionne les questions du thème de l'histoire on peut voir :



The dialog box has a title bar with 'JB' and standard window controls. It features a 'PRECEDENT' button in the top left. The main text reads 'Choisissez un thème et validez pour afficher les questions :'. Below this, there is a dropdown menu showing 'Histoire' and a 'Valider' button. The main area is a large text box containing two questions. The first question is 'Question numero: 35', 'Difficulte : Difficile', 'Enonce / Vrai ou Faux : Ce jour-la, des avions se sont ecrases sur les deux tours jumelles. C'etait le 11 mai 2001', with options '1. Vrai' and '2. Faux', and the correct answer 'Bonne reponse: Faux'. The second question is 'Question numero: 37', 'Difficulte : Facile', 'Enonce / QCM : Quand a eu lieu la revolution francaise ?', with options '1. 1781', '2. 1787', '3. 1789', and the correct answer 'Bonne reponse: 1789'.

JB

PRECEDENT

Choisissez un thème et validez pour afficher les questions :

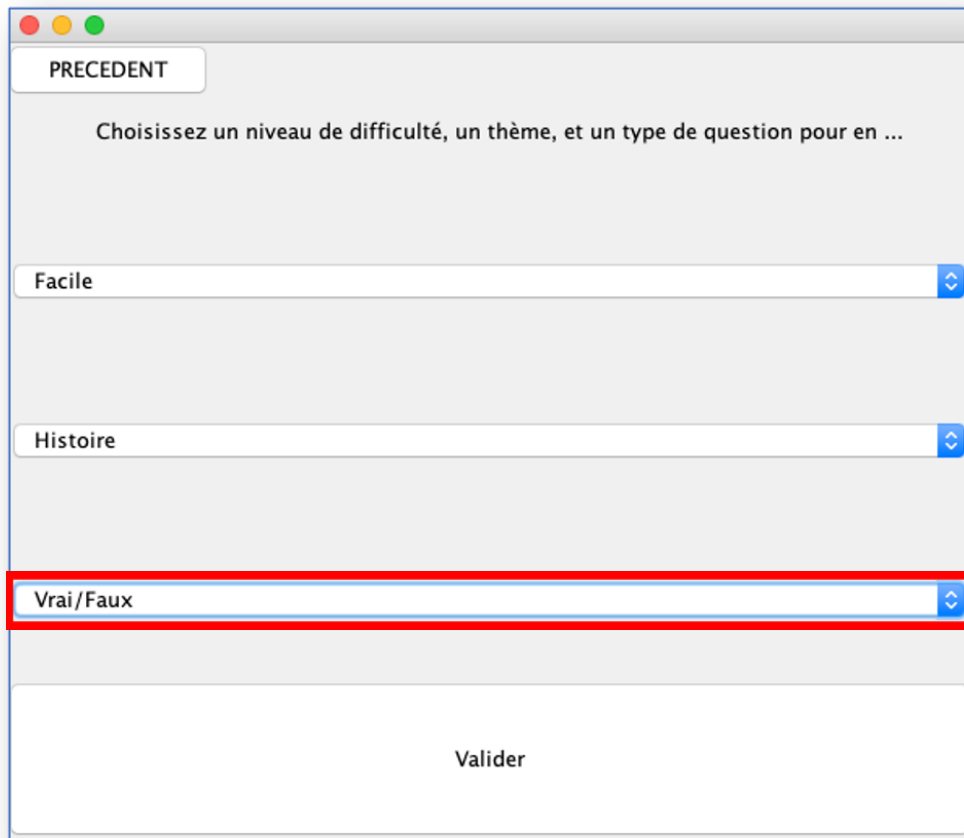
Histoire Valider

Question numero: 35
Difficulte : Difficile
Enonce / Vrai ou Faux :
Ce jour-la, des avions se sont ecrases sur les deux tours jumelles. C'etait le 11 mai 2001
1. Vrai
2. Faux
Bonne reponse: Faux

Question numero: 37
Difficulte : Facile
Enonce / QCM :
Quand a eu lieu la revolution francaise ?
1. 1781
2. 1787
3. 1789
Bonne reponse: 1789

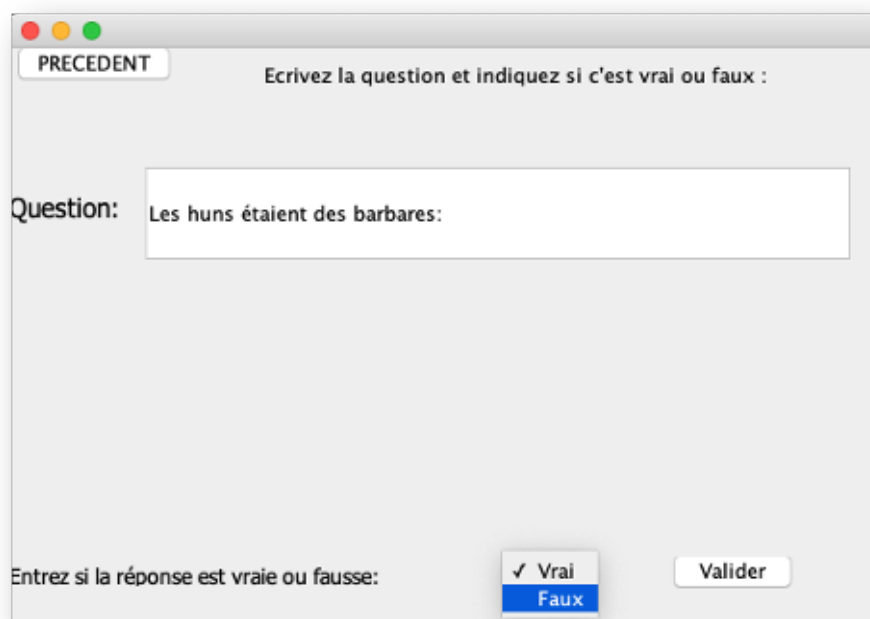
Comme on peut le voir, notre question ajoutée est disponible dans notre jeu et est considérée comme n'importe quelle autre question.

2) AJOUT VRAI/FAUX



A screenshot of a Java Swing window titled 'Ajout Vrai/Faux'. The window has a title bar with red, yellow, and green buttons. Inside, there is a button labeled 'PRECEDENT' in the top left. Below it, a text label says 'Choisissez un niveau de difficulté, un thème, et un type de question pour en ...'. There are three dropdown menus: the first is set to 'Facile', the second to 'Histoire', and the third, which is highlighted with a red rectangle, is set to 'Vrai/Faux'. At the bottom, there is a 'Valider' button.

Dans le même esprit qu'énoncé précédemment, ici nous choisissons d'ajouter une question de type Vrai/Faux, ce qui nous redirige sur la fenêtre ci-dessous:

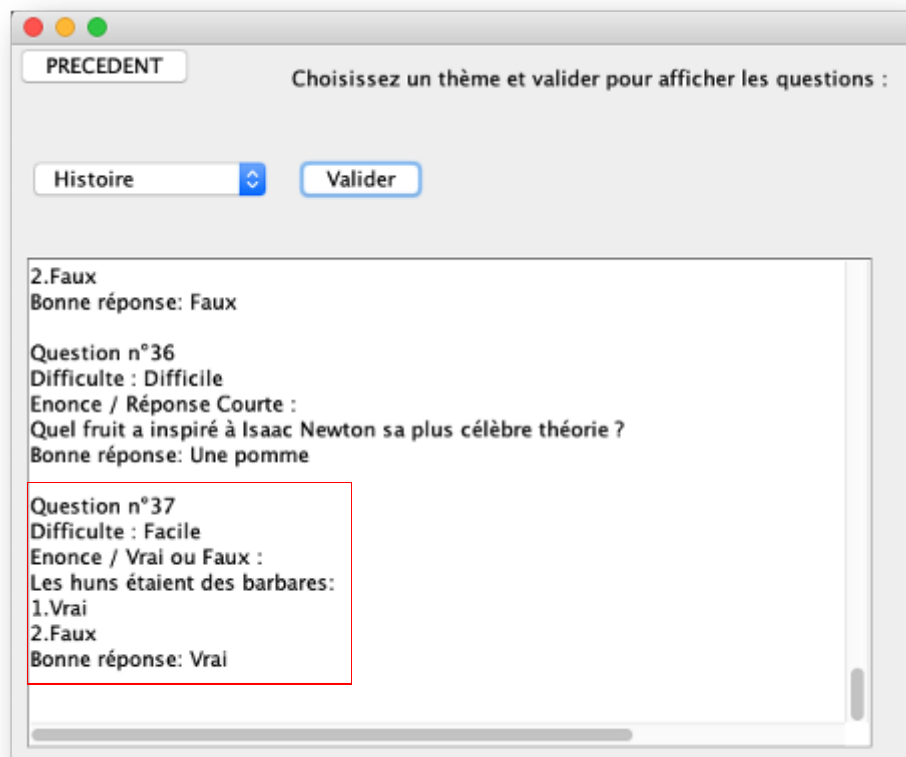


A screenshot of the same Java Swing window, now showing the 'Ajout Vrai/Faux' dialog. The 'PRECEDENT' button is still in the top left. The text label now says 'Ecrivez la question et indiquez si c'est vrai ou faux :'. Below this, there is a text area labeled 'Question:' containing the text 'Les hunns étaient des barbares:'. At the bottom, there is a label 'Entrez si la réponse est vraie ou fausse:' followed by two buttons: '✓ Vrai' and 'Faux'. The 'Faux' button is highlighted with a blue background. To the right of these buttons is a 'Valider' button.

On remplit donc le champs correspondant à la question et on indique simplement si cette question est vraie ou fausse dans le menu déroulant, puis on clique sur le bouton "Valider".



La question s'ajoute puis par la suite on revient observer la liste de questions du thème Histoire pour vérifier si notre question de type Vrai/Faux a été ajoutée.



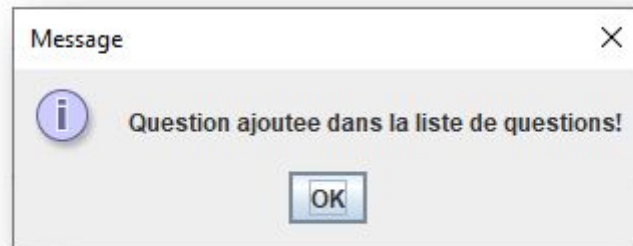
3) AJOUT REPONSE COURTE

A Java Swing window titled "PRECEDENT" with a standard macOS-style title bar (red, yellow, green buttons). The window contains a label "Choisissez un niveau de difficulté, un thème, et un type de question pour en ...". Below this are three dropdown menus. The first dropdown is set to "Facile". The second dropdown is set to "Histoire". The third dropdown is set to "Réponse Courte" and is highlighted with a red rectangular border. At the bottom of the window is a button labeled "Valider".

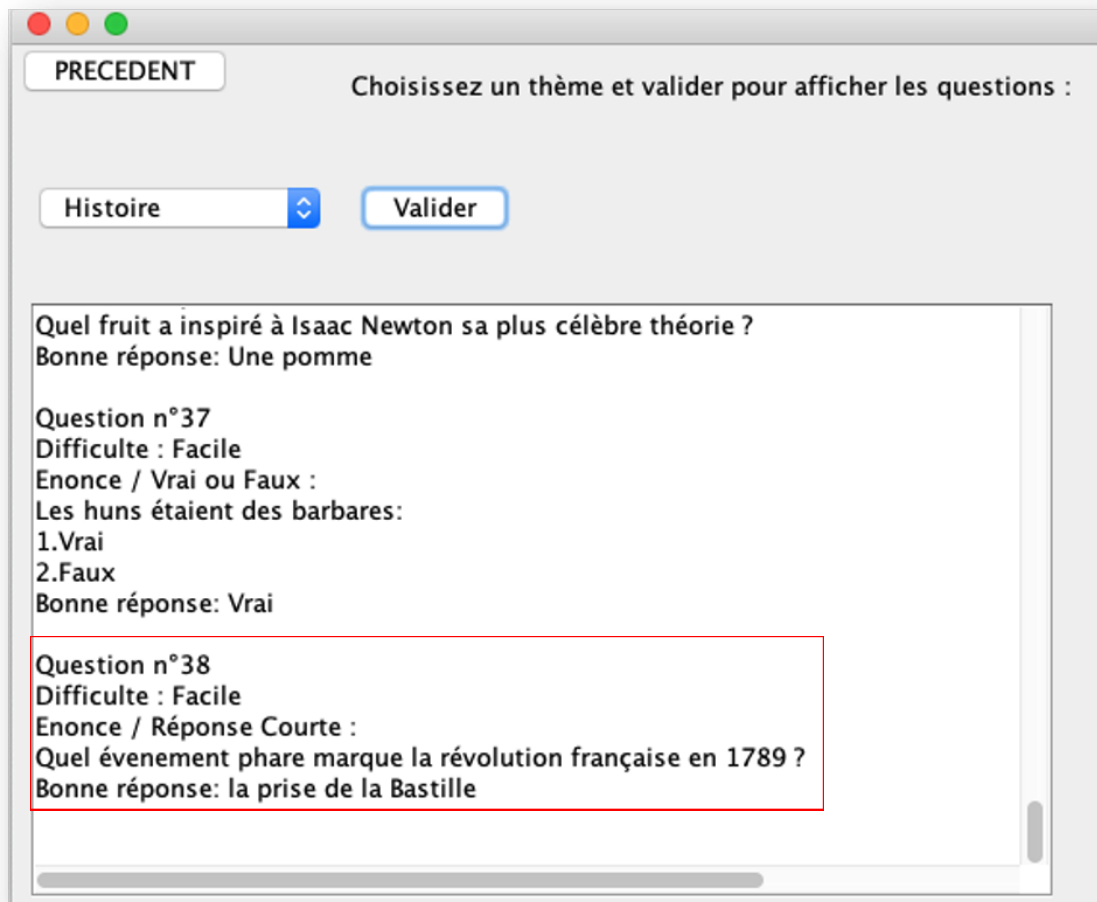
Dans le même esprit qu'énoncé précédemment, ici nous choisissons d'ajouter une question de type réponse courte, ce qui nous redirige sur la fenêtre ci-dessous:

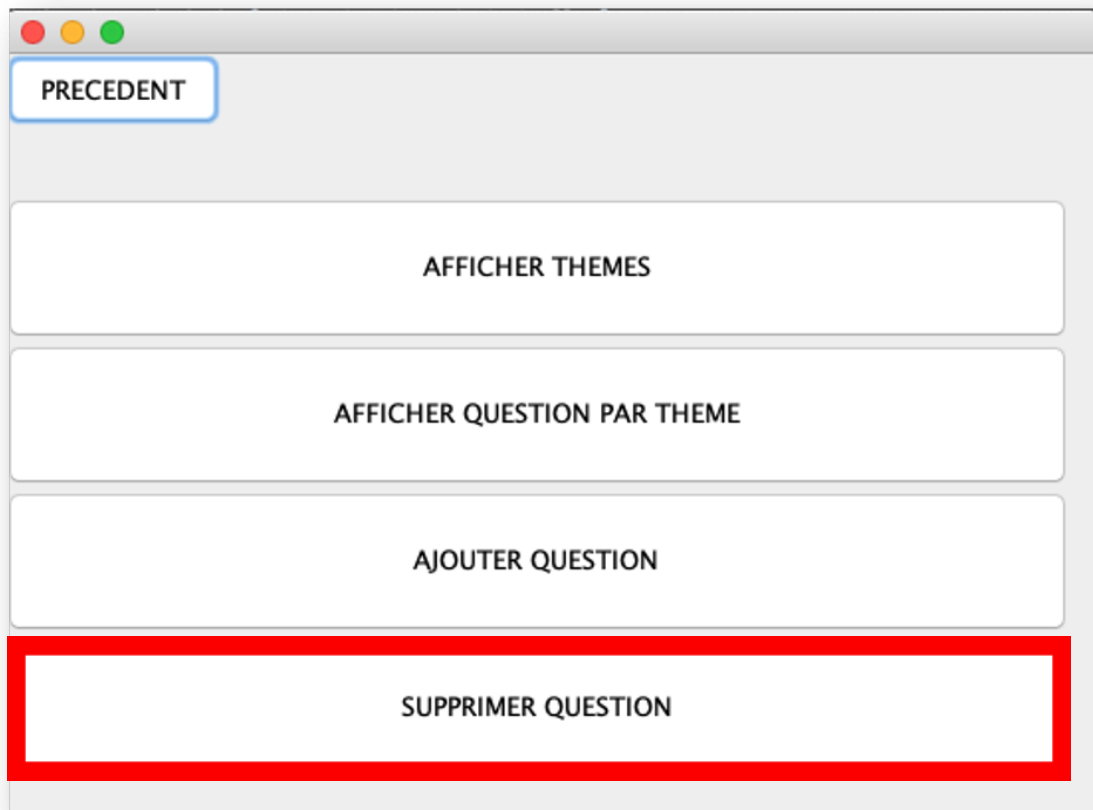
A Java Swing window titled "PRECEDENT" with a standard macOS-style title bar. The window contains a label "Ecrivez la question et la réponse :". Below this are two text input fields. The first field is labeled "Question:" and contains the text "Quel événement phare marque la révolution française en 1789 ?". The second field is labeled "Réponse:" and contains the text "la prise de la Bastille". At the bottom right of the window is a button labeled "Valider".

On remplit donc le champs correspondant à la question et on indique la réponse dans le champs "Réponse" juste en dessous. Puis on valide encore une fois pour ajouter la question dans la liste de questions.

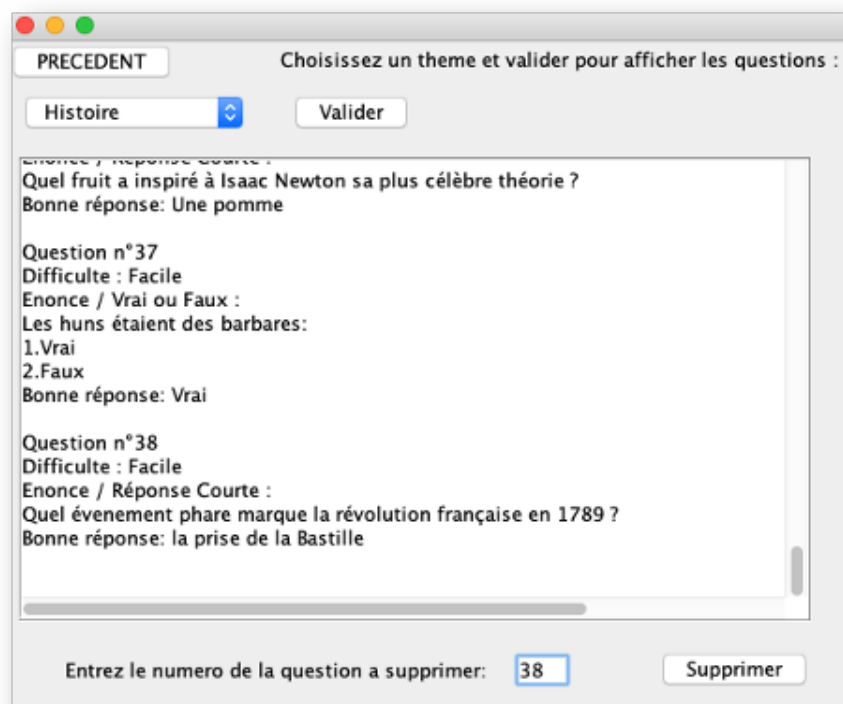


Par la suite on revient observer la liste de questions du thème Histoire pour vérifier si notre question de type réponse courte a été ajoutée.



d) Supprimer Question

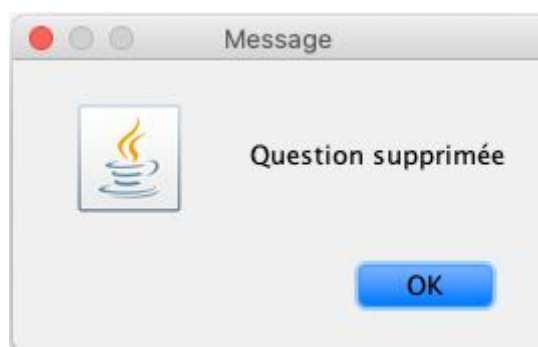
Dans le menu modération nous pouvons également trouver la fonction de pouvoir supprimer une question de notre choix. Lorsque l'on clique sur le bouton dans le menu modération, cela nous redirige directement vers la fenêtre ci-dessous basée sur le même modèle que la fenêtre affichant les questions par thème:



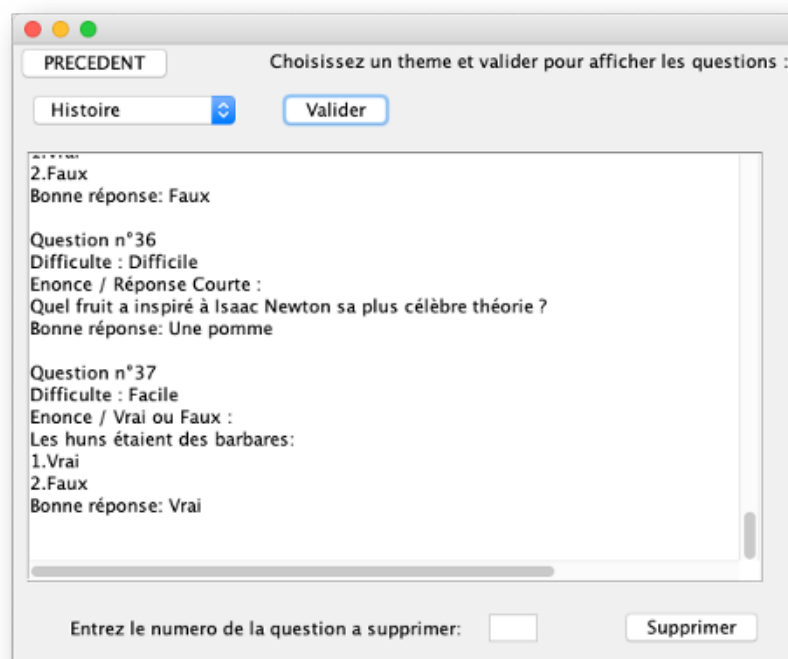
Il nous reste qu'à sélectionner le thème où se trouve la question que nous voulons supprimer, nous pouvons repérer son numéro indiqué une fois les questions de ce thème affichées, puis à l'inscrire dans le champs en bas de la fenêtre.

Nous devons faire extrêmement attention avec cette fonction car une suppression abusive d'un grand nombre de questions peut atteindre le fonctionnement de notre jeu. Par conséquent avons dû ajouter un certain nombre de conditions afin que la question puisse être supprimée ou non selon les conditions. En effet notre jeu nécessite un minimum de 3 questions de chaque difficulté pour chaque type de questions pour chaque thème ce qui fait 90 questions. Si cette condition n'est pas respectée alors nous mettons en péril le bon déroulement du jeu.

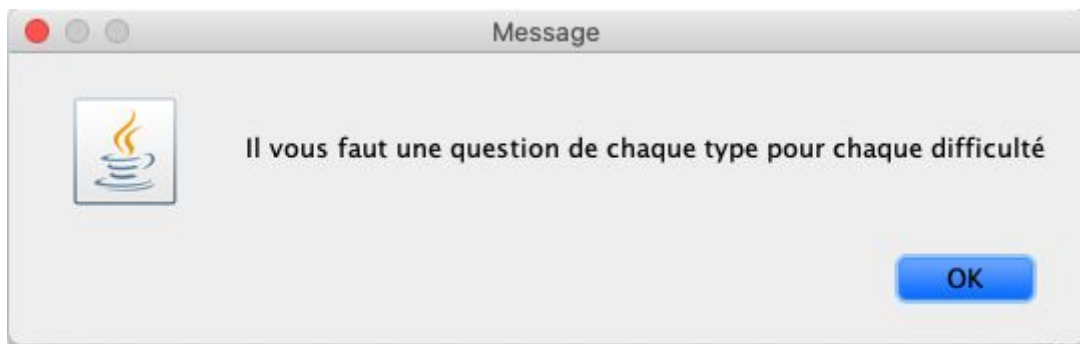
Dans le cas d'une suppression faite avec succès nous obtenons ce message:



Avec l'effet immédiat sur notre liste de questions grâce à la sérialisation, sur la même base que la fonction d'ajout de questions, à chaque modification des questions nous devons modifier la HashMap. Pour que ces questions restent bien supprimées et **persistant** dans le futur nous devons sérialiser la HashMap mise à jour une fois la question supprimée. Nous pouvons voir ici le résultat de la suppression de la question n°38 dans le thème histoire qui reste supprimée même après redémarrage du jeu:



Dans le cas d'une suppression non réussie nous obtenons ce message:



Conclusion

Grâce à ce projet nous avons pu mettre à profit ce que nous avons appris au cours de ce semestre de Java-2. Nous avons pu pratiquer quasiment tous les points abordés durant les cours, ce qui nous a permis de consolider nos acquis. Même si nous réalisons des projets en équipe depuis maintenant trois ans, il est toujours intéressant et parfois difficile de toujours s'entendre et surtout de se comprendre. Malgré tout, nous avons réussi à nous comprendre et avons su trouver un rythme de travail régulier qui convenait à tous.

En outre nous considérons ce projet comme une réussite que nous pouvons compter à l'actif de notre équipe.