

31 Mai 2023  
INFO1 – ENSSAT Lannion

JAVA  
Rapport de Projet :  
**Application de Quiz.**

BUENO Vincent, MONTEL Théo

## **Sommaire :**

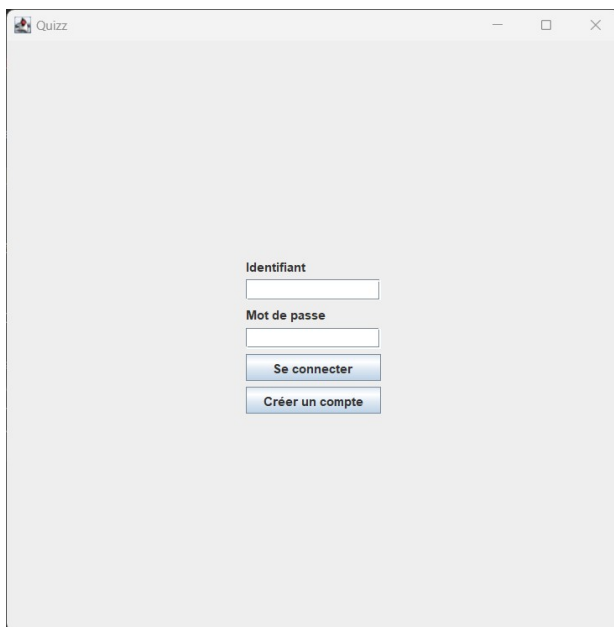
1. Introduction
2. Description de l'application et fonctionnement
3. Choix des classes et d'implémentation
4. Pistes d'amélioration
5. Conclusion
6. Diagramme de classe (Fichier annexe)

## **I – Introduction :**

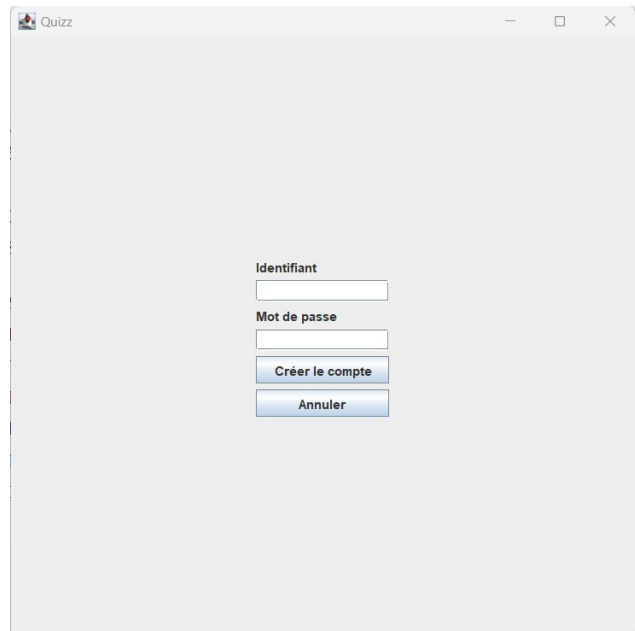
L'objectif du projet est de réaliser une application de quiz sous java. L'application doit proposer deux modes de jeu un mode solo et un mode multijoueur. L'application propose 2 types de comptes : admins et joueurs, suite à la connexion l'utilisateur dispose de l'historique de son compte et d'accès aux différents modes de jeu ainsi qu'à l'espace administrateur pour les comptes concernés. L'espace administrateur permet d'ajouter des questions, de promouvoir des comptes au rang d'admin ou de suspendre des comptes.

## **II – Description de l'application et fonctionnement :**

La connexion et la création de compte sont gérés par deux classes, Connexion et CreerCompte.



The screenshot shows a Java Swing window titled 'Quizz' with a light gray background. In the center, there is a login form. It consists of two text input fields: the first is labeled 'Identifiant' and the second is labeled 'Mot de passe'. Below these fields are two buttons: 'Se connecter' (top) and 'Créer un compte' (bottom). Both buttons have a blue gradient and white text.

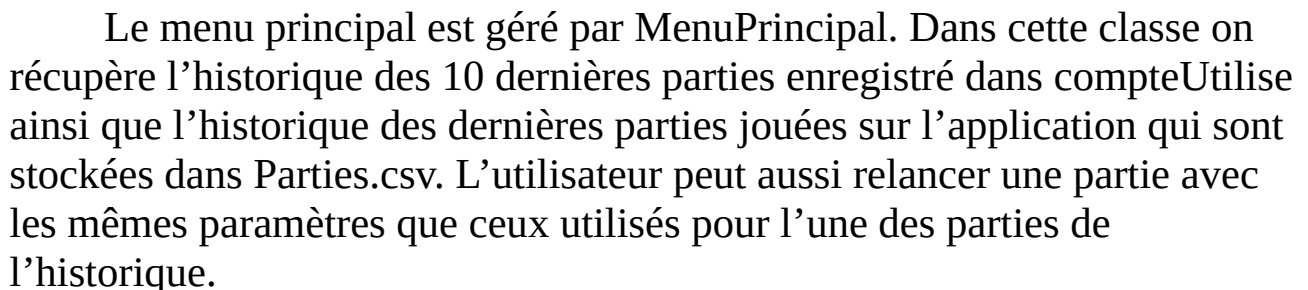


The screenshot shows a Java Swing window titled 'Quizz' with a light gray background. In the center, there is a registration form. It consists of two text input fields: the first is labeled 'Identifiant' and the second is labeled 'Mot de passe'. Below these fields are two buttons: 'Créer le compte' (top) and 'Annuler' (bottom). Both buttons have a blue gradient and white text.

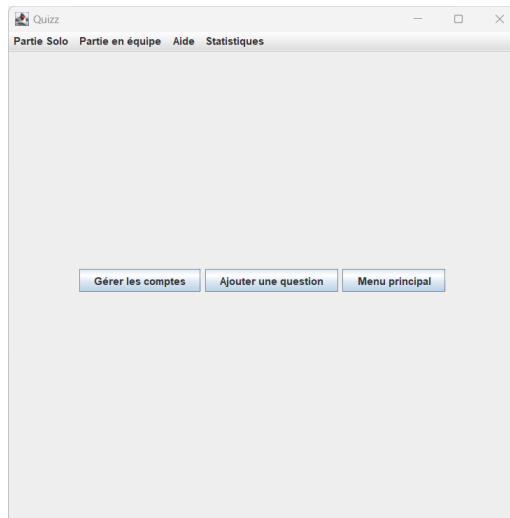
Connexion génère une fenêtre de connexion :

Lors de l'appui sur le bouton Créer un compte, l'utilisateur est redirigé vers CreerCompte.

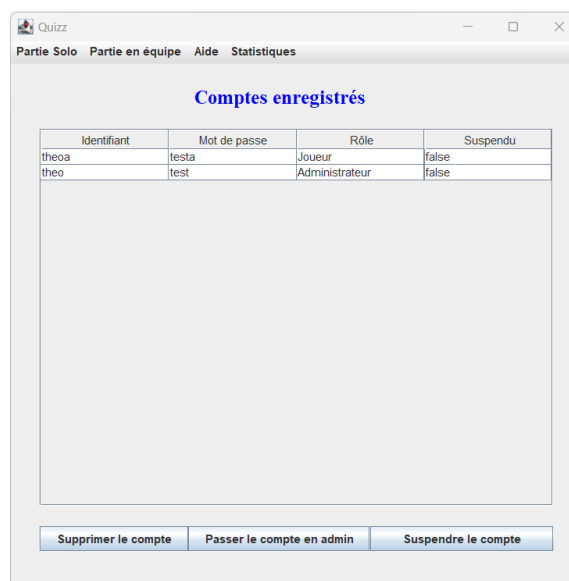
CréerCompte génère une fenêtre qui permet à l'utilisateur d'entrer son identifiant et son mot de passe. Lors de l'appuie sur le bouton créer le compte, on vérifie si l'identifiant n'est pas déjà utilisé dans Compte.data, puis si c'est le cas, on enregistre un nouveau compte dans Compte.data, enfin l'utilisateur est renvoyé à la page de connexion.



L'utilisateur peut aussi lancer une partie en solo ou en multi, se déconnecter pour revenir à l'écran de connexion ou encore accéder à l'espace administrateur si son compte est un compte admin.



L'espace administrateur repose sur les classes EspaceAdministrateur et NouvelleQuestion et GestionCompte et permet à l'utilisateur d'accéder à la gestion des autres comptes, d'ajouter une question ou de retourner au menu principal.



La gestion des autres comptes est implémenté par GestionCompte, cette classe permet de supprimer, promouvoir ou suspendre un compte en modifiant les comptes enregistrés dans Compte.data pour quitter cette page l'utilisateur doit utiliser l'onglet aide → menu principal.

L'ajout de question est géré par NouvelleQuestion qui écrit la nouvelle question dans un fichier csv en remplissant les colonnes du csv avec les infos entrées par l'utilisateur en vérifiant qu'il y ai au moins 3 caractères par champ et en supprimant les virgules et les apostrophes.

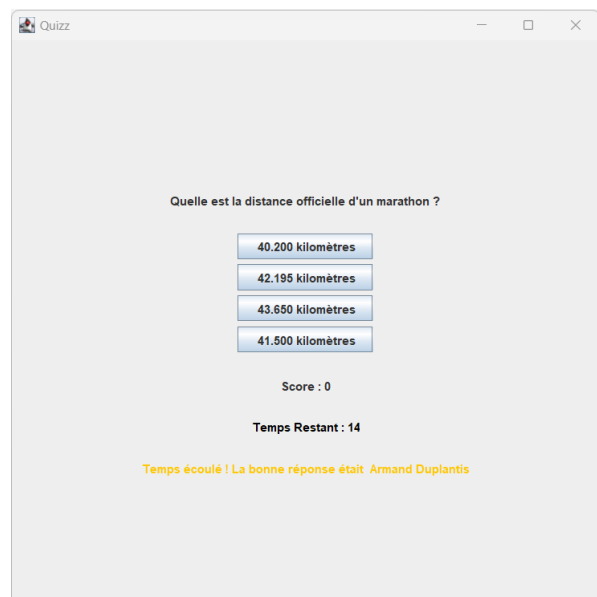
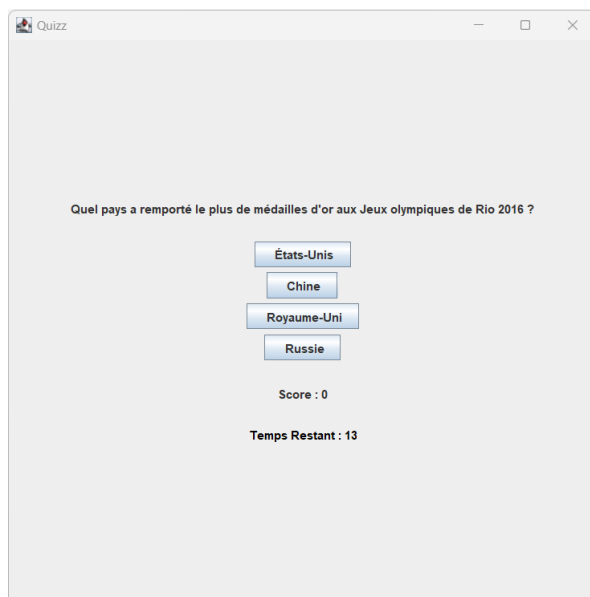
Les lancements de parties solo ou multi sont gérés respectivement par ChoixJeuSolo et ChoixJeuMulti. L'utilisateur à plusieurs moyens d'atteindre la page de création de partie , il peut utiliser les menus déroulants situés en haut de la fenêtre ou cliquer sur les boutons de jeu sur le menu principal.

Ces deux Classes affichent une fenêtre permettant à l'utilisateur de choisir les réglages de la partie puis de lancer la partie en appuyant sur « Jouer ! » pour le mode solo ou sur « Jouer à la V1/V2 ! » pour le mode multijoueur.

L'appuie sur le bouton « Jouer ! » range dans une liste toutes les questions du niveau et des thèmes sélectionnés, puis mélange cette liste et envoie les  $n$  ( $n$  est le nombre de questions choisis par l'utilisateur) premières questions au constructeur de la classe JeuSolo.

L'appuie sur le bouton « Jouer à la V1 ! » range dans une liste toutes les questions du niveau et des thèmes sélectionnés, mélange cette liste et range les  $n$  premières questions dans une liste `listeQuestionsA` puis remélange et range de nouveau les  $n$  premières questions dans une liste `listeQuestionsB`. Enfin on appelle de constructeur de la classe `JeuMultiV1` avec pour argument les deux listes de questions. `JeuMultiV1` gère une partie en mode multijoueur V1, l'équipe A commence et une équipe joue jusqu'à se tromper ou jusqu'à ce que sa liste de question soit vide, la partie se termine lorsque les deux listes de questions sont vides.

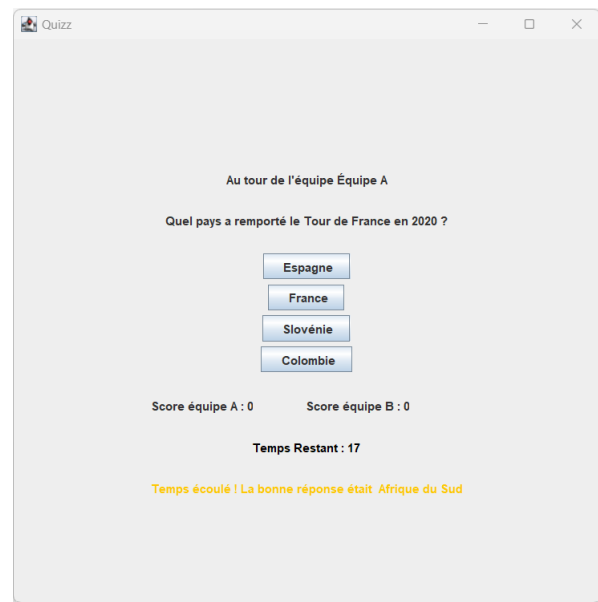
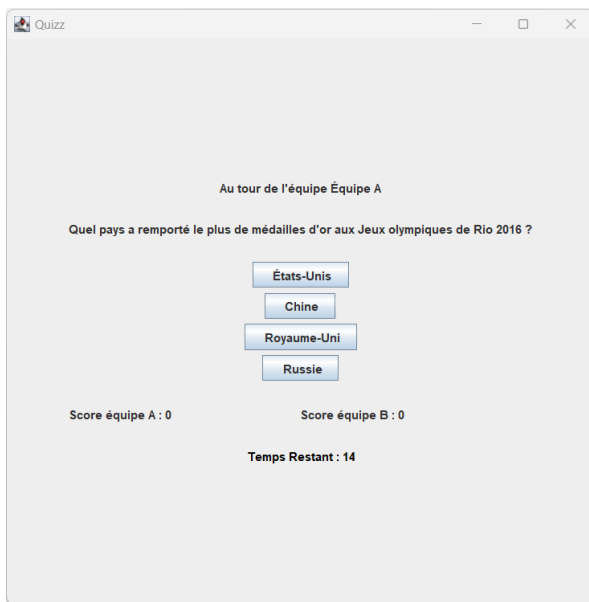
L'appuie sur le bouton « Jouer à la V2 ! » range dans une liste toutes les questions du niveau et des thèmes sélectionnés, mélange cette liste et appelle le constructeur de la classe `JeuMultiV2` avec pour argument les  $2*n$  premières questions de la liste mélangée. `JeuMultiV2` gère une partie en mode multijoueur V2, l'équipe A commence et les équipes jouent tour à tour, la partie se termine lorsque la liste de questions est vides.



JeuSolo gère les parties en mode solo. Lors de l'appel de son constructeur on affiche la première de la liste question, les 4 boutons contenant les réponses possibles, le score actuel du joueur et le temps restant pour répondre à la question.

Lorsque le joueur appuie sur un bouton de réponse ou lorsque le temps est écoulé, on rappelle le constructeur de JeuSolo avec le score mis à jour et la liste des questions restantes.

Quand la liste de question est vide l'utilisateur est redirigé vers l'écran de fin de partie géré par FinJeu.



JeuMultiV1 fonctionne de façon similaire à JeuSolo, cependant plusieurs nouveaux cas sont possible :

- L'équipe ne répond pas ou se trompe

- si l'autre équipe à encore des questions à jouer

- Alors on appelle JeuMultiV1 en changeant la valeur indiquant quelle équipe doit jouer.

- sinon on appelle JeuMultiV1 en gardant la même équipe.

- L'équipe n'a plus de questions

- si l'autre équipe à encore des questions à jouer

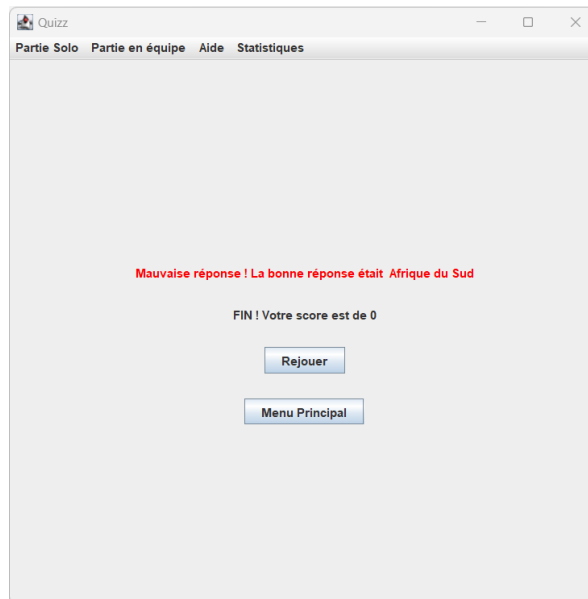
- Alors on appelle JeuMultiV1 en changeant la valeur indiquant quelle équipe doit jouer.

- Sinon on appelle FinJeuMulti et la partie se termine.

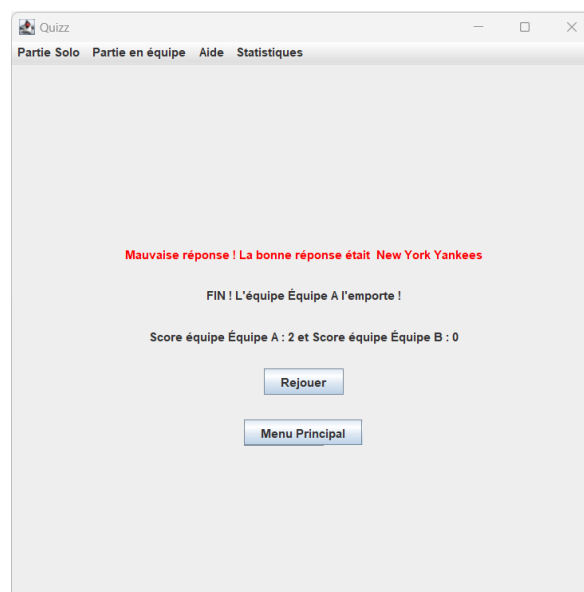


JeuMultiV2 fonctionne de façon très similaire à JeuSolo, cependant à chaque appel de JeuMultiV2 on change la valeur qui indique quelle équipe doit jouer.

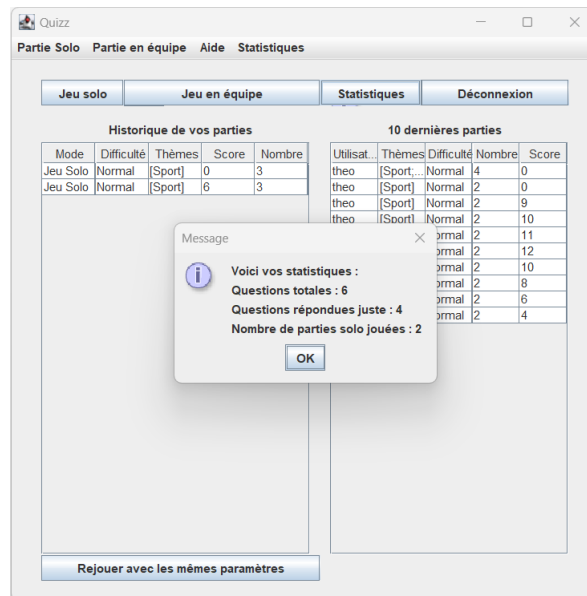
FinJeu gère l'écran de fin de partie solo et génère une musique de fin de partie. L'utilisateur peut soit relancer une partie avec les mêmes paramètres en cliquant sur Rejouer soit retourner au menu principal.



FinJeuMulti gère l'écran de fin de partie multijoueur et génère une musique de fin de partie. L'utilisateur peut voir quelle équipe a gagné et le score des 2 équipes, il peut aussi relancer une partie et retourner au menu principal.



Enfin le joueur peut accéder à ses statistiques de compte via le bouton statistiques sur le menu principal ou le bandeau déroulant statistiques en haut de la fenêtre. Cet page affiche le nombre de partie jouées, le nombre de questions auxquelles le joueur a répondu et le nombre de réponses justes.



### **III – Choix des classes et d’implémentations :**

Premièrement on implémente une interface que toutes les classes graphiques implémenterons. Cette interface contient une JFrame, un CardLayout et un Container afin de gérer les différents affichages de chaque classe et de pouvoir appeler chaque affichage dans chaque classe.

On dispose d’une classe Compte constitué de plusieurs attributs (identifiant, mot de passe, Historique de Parties, Statistiques, ...) qui nous permettent ainsi d’enregistrer chaque agissements d’un utilisateur en stérilisant le compte dans un fichier Comptes.data afin de les récupérer ou de les modifier facilement.

On dispose d’une classe Question et Partie qui sont directement utiles pour le jeu et pour les Comptes. On enregistre les parties jouées en solo et les questions dans des csv afin d’avoir un accès direct (sans passer par java) ce qui est pratique pour ajouter des questions en grande quantité.

On dispose d’une classe BoutonReponse qui regroupe un JButton et un Timer afin de permettre au bouton d’éteindre le Timer dans le jeu.

Dans le code on utilise une boucle for pour la JTable dans le menu principal car on veut exactement 10 éléments.

On utilise aussi a de nombreuses reprises une boucle while avec un itérateur pour parcourir une ArrayList des parties d'un compte et toutes les afficher dans la JTable dans le menu principal.

De même pour récupérer tous les thèmes dans les ArrayList de thèmes pour lancer le jeu et récupérer les questions correspondantes.

Dans ces deux cas nous utilisons une boucle while car nous utilisons un itérateur et nous ne connaissons pas le nombre au préalable.

#### **IV – Pistes d'amélioration**

Dans un premier temps, nous pourrions rendre le code plus lisible en isolant l'interface graphique du reste du code.

Ainsi nous pourrions mieux exploité la mécanique d'héritage afin de limité les changements entre les trois jeux existants et ne pas avoir à refaire des classes à partir de zéro.

De plus, avec plus de temps nous aurions pu implémenter les fonctionnalités bonus. Nous pourrions aussi ajouter la possibilité d'ajouter des thèmes.

#### **V – Conclusion**

Finalement, notre application de quiz est fonctionnelle et implémente toutes les fonctions demandés. Cependant certaines choses restent à améliorer telle que la qualité et la lisibilité du code.