Rapport de projet

Analyse et Conception de Logiciels

ROMEO Florian & WEBERT Chris

Sommaire

Objectifs

L'objectif de ce projet est d'analyser un cahier des charges puis de concevoir un programme à partir de celui-ci, en utilisant le langage Java.

L'analyse porte dans un premier temps sur les besoins de l'utilisateur, puis sur l'implémentation.

Cahier des charges

Jeu

Le client souhaite créer un jeu de cartes à un seul joueur.

Il sera basé sur les cartes de la Belote. Pour rappel, ce sont 32 cartes, réparties en quatre enseignes, contenant chacune les figures suivantes : 7, 8, 9, 10, Valet, Dame, Roi, As.

Les cartes possèdent les valeurs suivantes :

Carte	Valeur
As	11
Roi	4
Dame	3
Valet	2
10	10
9	0
8	0
7	0

Le joueur commence avec un score de 0 et doit tirer deux cartes pendant 5 tours. À chaque tour, on compare ces deux cartes :

- Si les figures et les couleurs sont identiques, le double de la somme des valeurs est soustraite au score ;
- Si les figures sont identiques mais pas les couleurs, la somme des valeurs est soustraite au score ;
- Sinon, la somme des valeurs est ajoutée au score.

Le but du jeu est d'obtenir le score le plus faible, qui sera inscrit au tableau des records à la fin de la partie.

Interface utilisateur

Lorsque le joueur lance le programme, il a le choix entre lancer une nouvelle partie, regarder le tableau des scores, ou quitter l'application.

Pour démarrer une nouvelle partie, il devra d'abord entrer son pseudonyme.

À tout moment, l'utilisateur pourra abandonner la partie et retourner au menu principal.

Choix personnels

Les cartes sont remises dans le paquet une fois le tour fini.

On ne conserve que le meilleur score de chaque joueur dans le tableau des records.

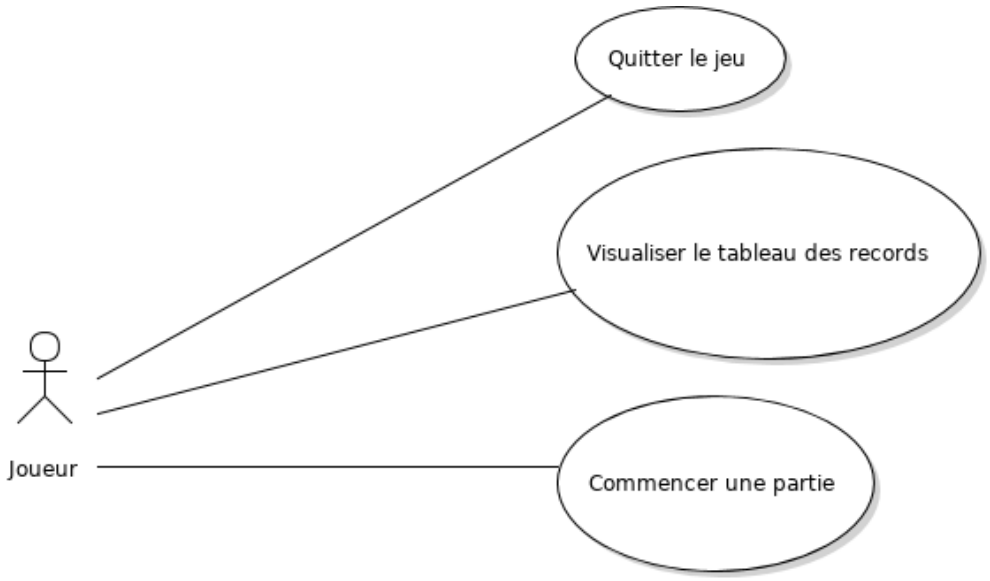
Analyse

On a tout d'abord analysé les besoins de l'utilisateur à travers les requêtes du client en élaborant les diagrammes cas-type et d'activités. On en a ainsi déduit un modèle pour les données, puis de là toute l'architecture de l'implémentation.

En utilisant le Round-Trip Engineering, on a donc pu se concentrer sur une conception orientée utilisateur pour piloter l'implémentation du projet.

Dans l'optique de rendre le programme le plus simple et accessible possible, toutes les actions se font à la souris via des boutons visibles et clairs. On a évité toutes interactions compliquées ou menus cachés (gestes, touches du clavier).

Cas-type



Cas type : Quitter le jeu
Étapes :

Actions de l'acteur	**Réponse du système**
1. Cliquer sur le bouton « Quitter »	2. La fenêtre se ferme

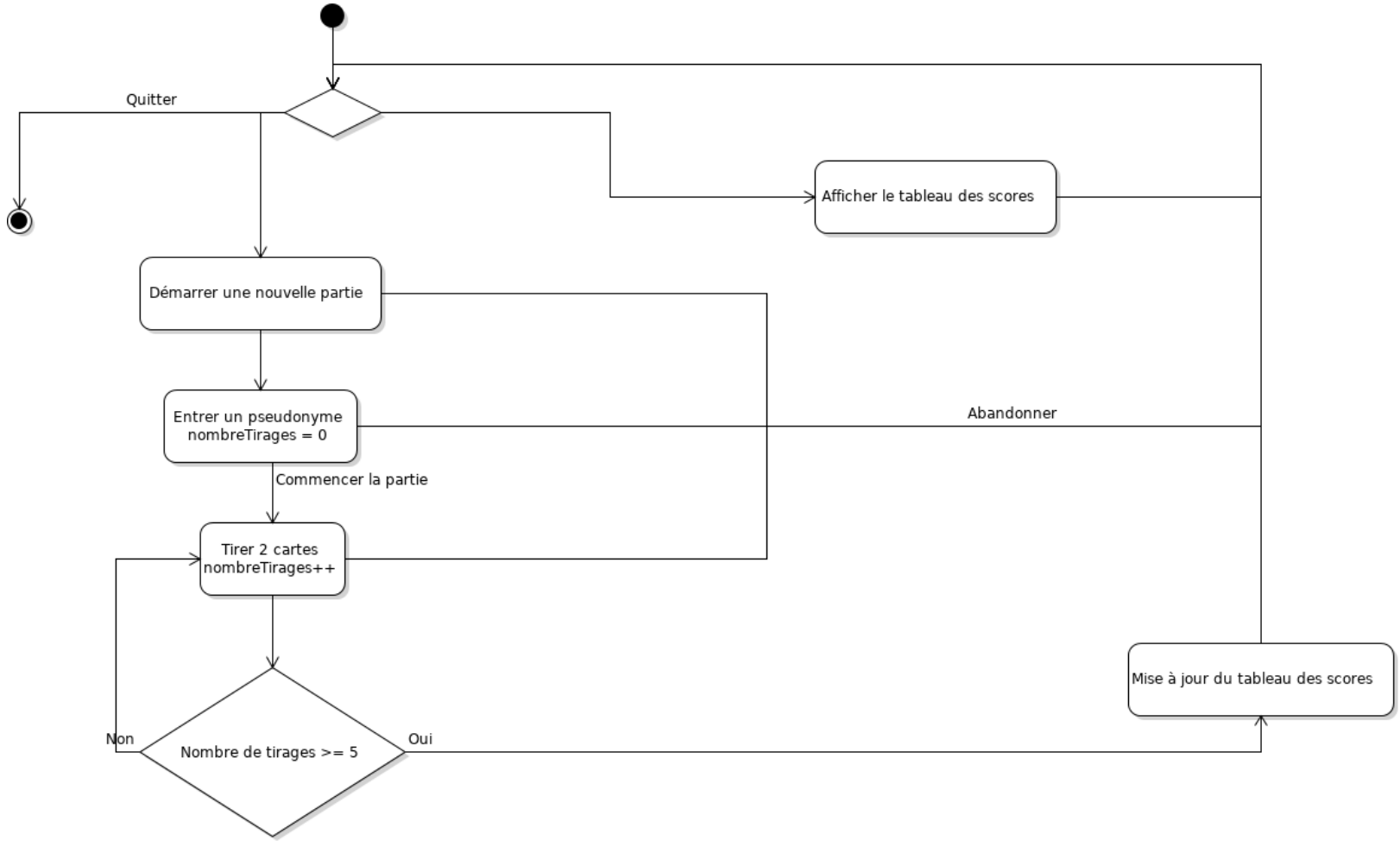
Cas type : Visualiser le tableau des records
Étapes :

Actions de l'acteur	**Réponse du système**
1. Cliquer sur le bouton « Tableau des records »	2. La fenêtre change pour afficher le tableau des records

Cas type : Commencer une partie
Étapes :

Actions de l'acteur	**Réponse du système**
1. Cliquer sur le bouton « Jouer »	
	2. La fenêtre change pour demander le pseudonyme
3. Entrer son pseudonyme	
4. Cliquer sur le bouton « Démarrer »	
	5. La fenêtre change pour afficher le plateau de jeu

Activités



Lorsque le programme se lance, le joueur a trois choix possibles :

- Démarrer une nouvelle partie :
  Le joueur peut ensuite :
  - Entrer un pseudonyme puis :
    - Tirer deux cartes puis :
      - Passer au tour suivant, ou sauvegarder si la partie est finie
      - Abandonner, et retourner au menu
    - Retourner au menu
  - Retourner au menu
- Afficher le tableau des scores
  et ensuite retourner au menu principal
- Quitter

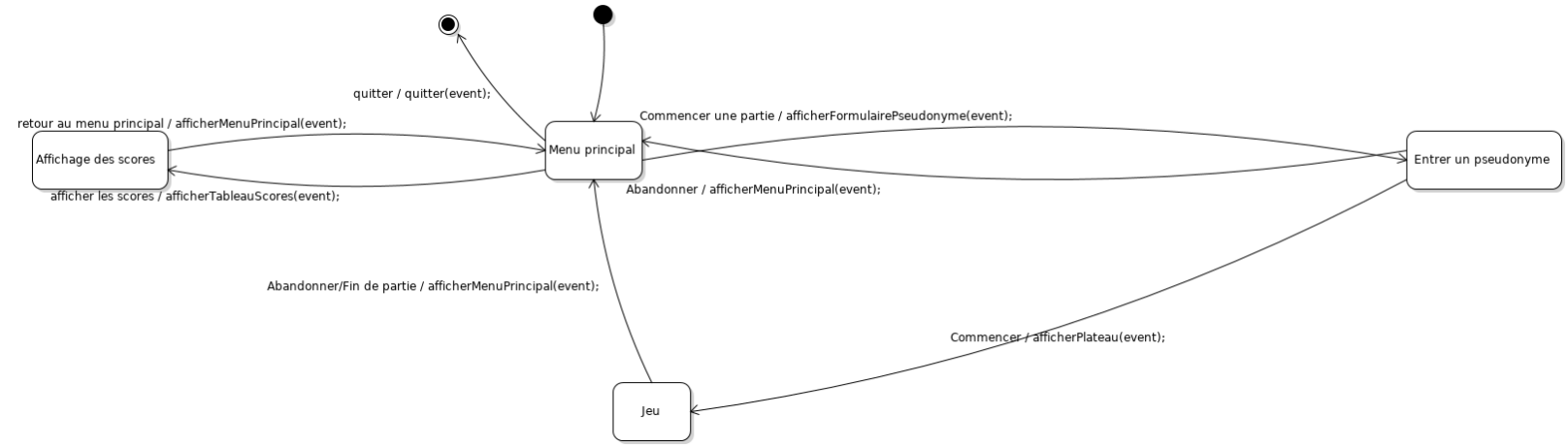
États

Le jeu commence par l'état « Menu Principal » où le joueur peut choisir :

- d'afficher les scores, pour passer le programme dans l'état « Affichage des scores »
- de commencer une partie, passant le programme dans l'état « Entrer un pseudonyme »
- de quitter, passant le programme dans son état terminal

Dans les deux premiers états, le joueur peut choisir de retourner au menu principal.

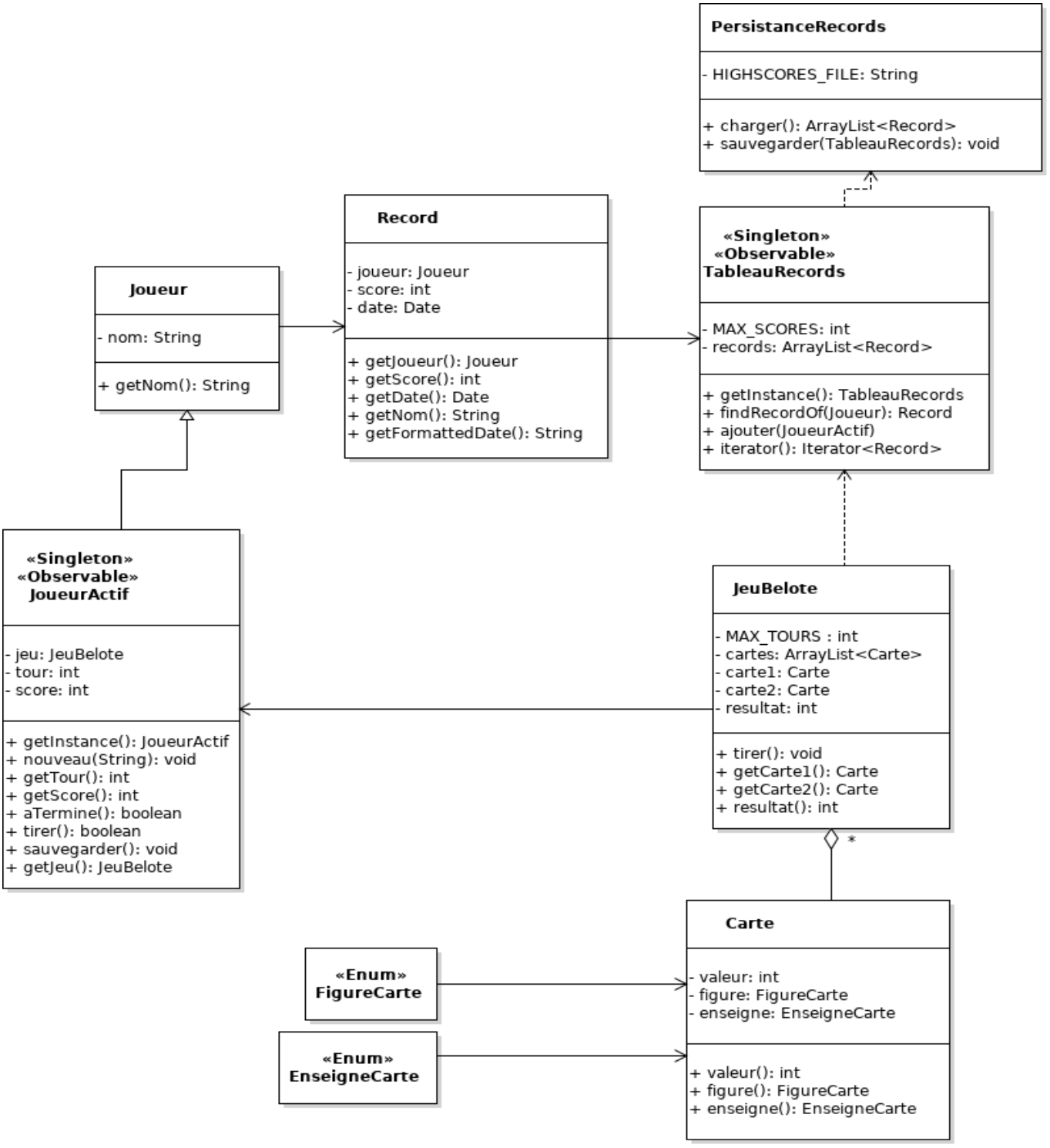
L'état « Entrer un pseudonyme » peut passer à l'état « Jeu » lorsque le joueur choisit de commencer sa partie, pour ensuite revenir à l'état « Menu Principal ».



Implémentation

Pour l'implémentation, nous avons mis en évidence une architecture MVC (Modèle-Vue-Contrôleur) avec JavaFX.

Modèle



La première étape de l’implémentation a été la conception du modèle.

Classes de données :

- Carte : représentation d’une carte de belote
- Joueur : un joueur, identifié par son nom
- Record : représentation de la meilleure partie d’un joueur enregistrée dans le tableau des records
- TableauRecords : contient le record de chaque joueur

Classes métier :

- JoueurActif : classe représentant le joueur actif, pouvant effectuer toutes les actions requises pour le jeu
  Comme le jeu est mono-joueur, cette classe est un singleton
- JeuBelote : cette classe contient les règles du jeu de Belote ainsi que les données du dernier tirage
- PersistenceRecords : permet de sauvegarder et de charger le tableau des records depuis un fichier.

Ce modèle est la base de notre application. Il gère toute la partie métier du jeu et de la gestion des records. Les interfaces permettent une gestion simple de la partie grâce aux contrôleurs et à la vue.

Sauvegarde

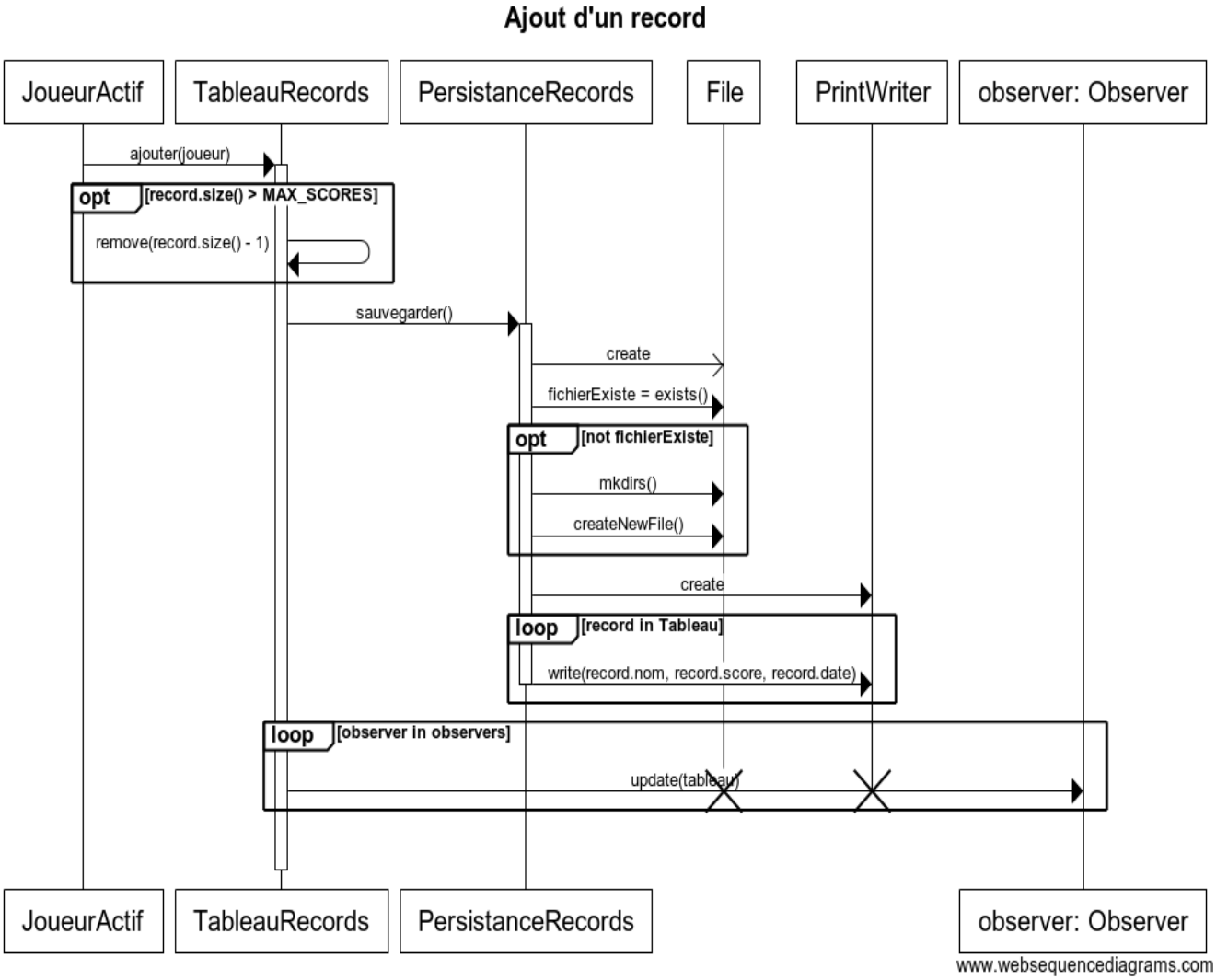
La sauvegarde s'effectue lorsqu'un joueur termine une partie, s'il n'est pas encore présent dans le tableau des records ou s'il améliore son précédent record.

Ce processus est déclenché par le joueur, qui peut ajouter son score aux tableau des meilleurs scores (classe TableauRecords). La méthode d'ajout vérifie que le joueur a bien amélioré son score et le met a jour le cas échéant. Si c'est sa première partie, le nom du joueur est ajouté au tableau des records. Sinon, aucune modification n'est faite.

Si le nombre de records maximal (10) est déjà atteint, on enlève le plus mauvais record du tableau.

La seconde partie du processus consiste à enregistrer de manière permanente ce tableau des records, afin qu'il ne soit pas perdu lors du redémarrage du jeu.

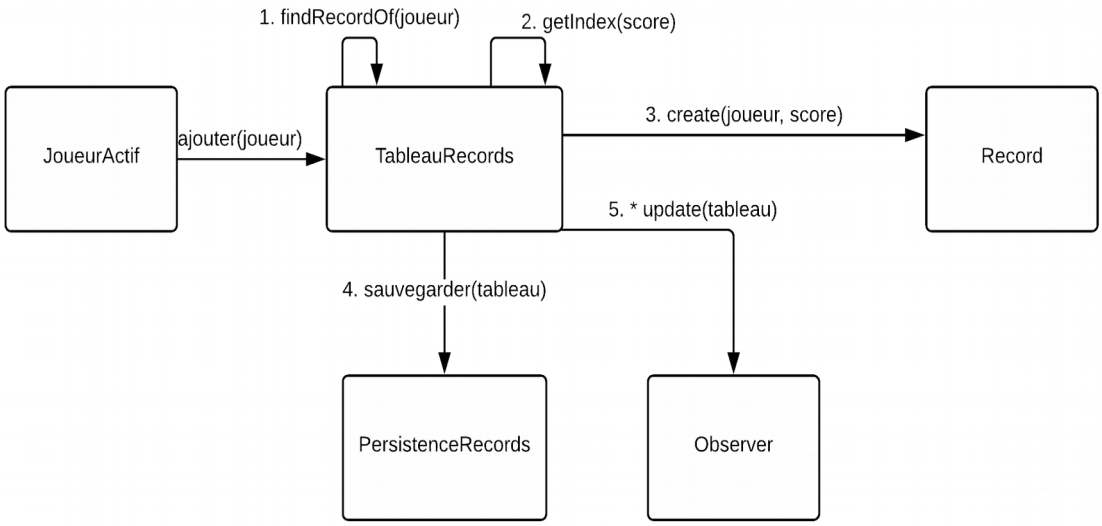
Nous déléguons la persistance de ce tableau à la classe PersistanceRecords afin de pouvoir changer facilement de méthode de stockage (par exemple passer d'un fichier texte à un serveur distant pour avoir un classement international).



Afin de garder une architecture correcte, la méthode de sauvegarde est déclenchée par le JoueurActif, pour que l'interface utilisateur ne s'occupe que de son lancement.

La vue sera un observeur de la classe TableauRecords, qui la notifiera de chaque mise à jour du tableau.

La persistance n'est utilisée que par la classe TableauRecords, afin de limiter la dépendance au système de la machine exécutant le programme.



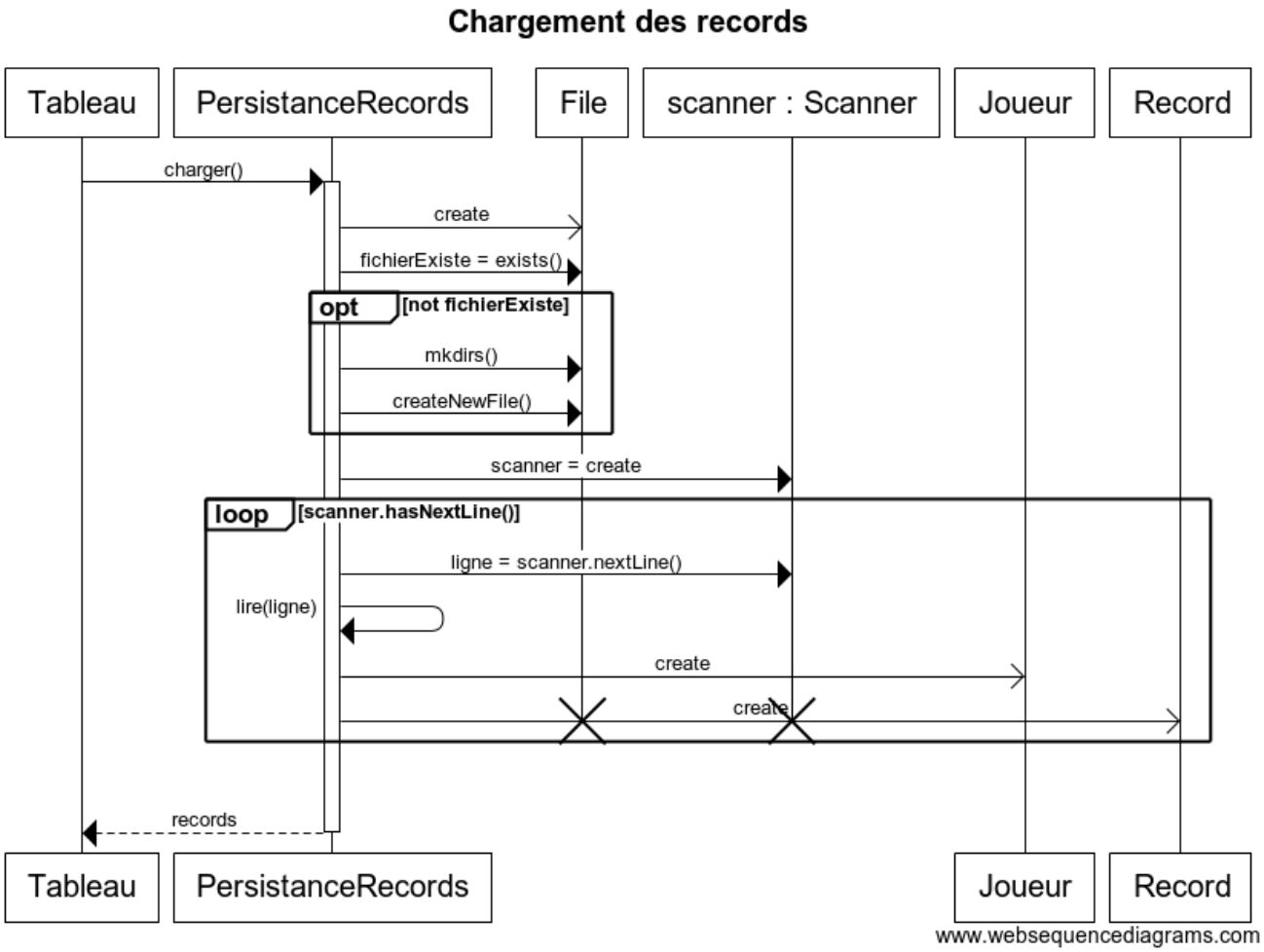
Chargement

Le processus de chargement des records s'effectue au lancement du programme.

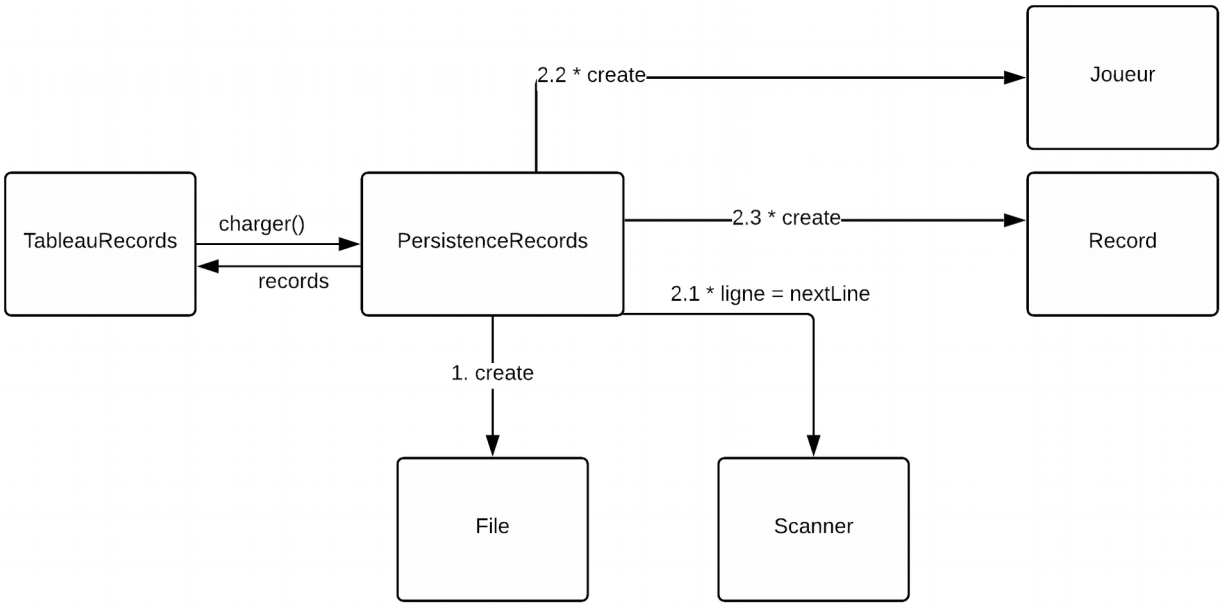
La classe TableauRecords est un singleton, qui lance le chargement dès sa création.

Le fichier des records, contenant le nom, le score et la date, sera ouvert par la classe PersistanceRecords, et crée s'il n'existe pas encore.

Chaque ligne sera lue, puis convertie en instance de l'objet Record et renvoyée au tableau des records.

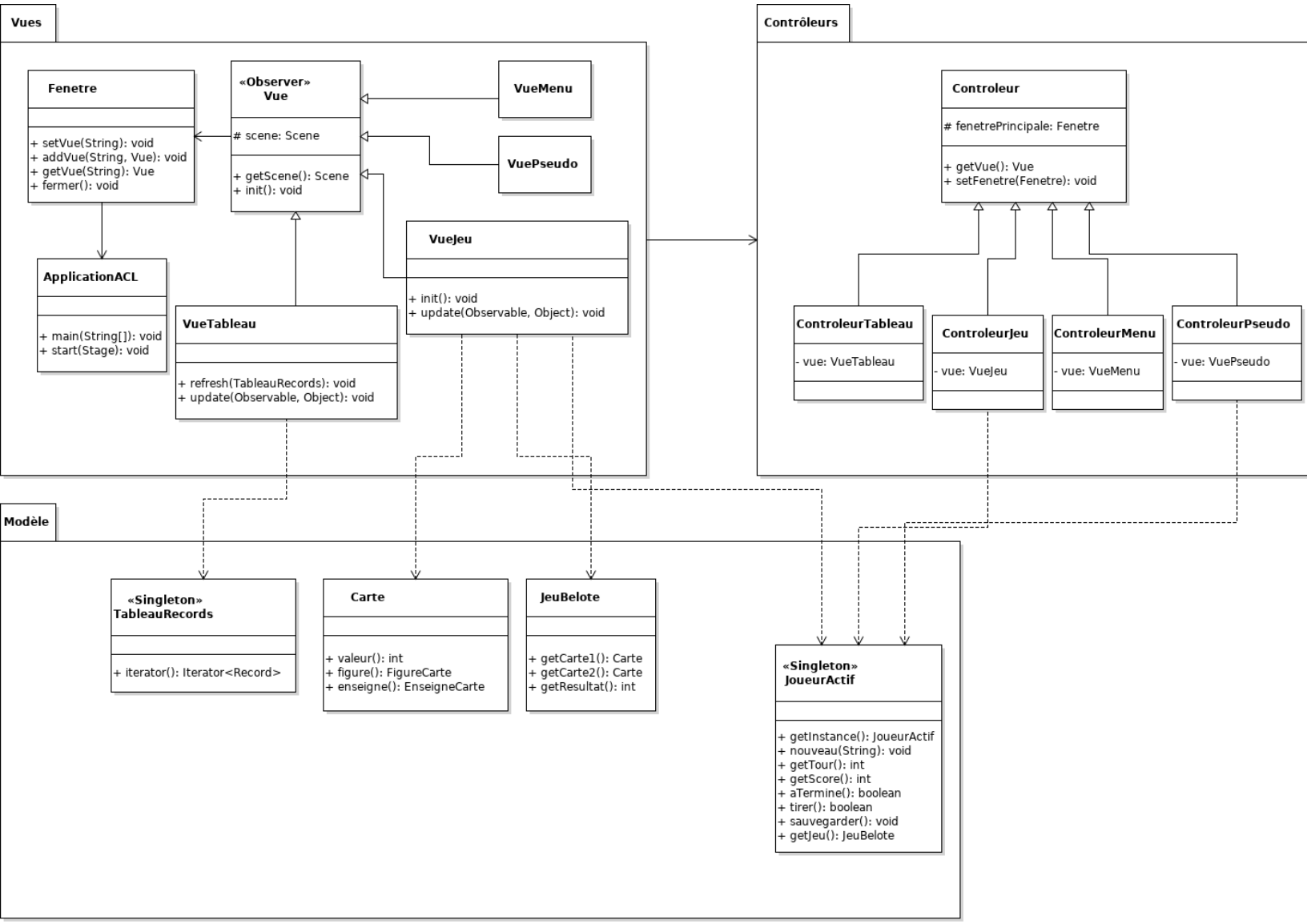


L'architecture de la procédure de chargement est très similaire à celle de sauvegarde. La classe PersistenceRecords s'occupe de lire le fichier, crée les objets du modèle (Joueur et Record) pour les ajouter au tableau des records.



Vue et Contrôleurs

Les vues et les contrôleurs sont basés sur JavaFX.



La classe principale, ApplicationACL, permet d'initialiser la bibliothèque JavaFX et de créer les vues ainsi que la Fenetre.

Chaque vue est représentée par une classe, contenant tous les éléments de l'interface, leur disposition ainsi que les éléments de mise à jour. Ces derniers sont déclenchés soit directement par une action du joueur, soit indirectement par le design pattern Observer-Observable (cas de VueJeu et VueTableau).

Les contrôleurs effectuent les actions suite aux événements déclenchés par l'utilisateur.

Nous disposons de quatre vues :

- Le menu principal, VueMenu, associé au contrôleur ControleurMenu;
- Le tableau des records, VueTableau, associé à ControleurTableau;
- Le formulaire d’entrée du pseudonyme du joueur, VuePseudo, associé à ControleurPseudo;
- Le jeu, VueJeu, associé à PlateauController.

Pour simplifier les interactions du joueur, toutes les actions possibles ont été placées dans la classe JoueurActif, qui est un singleton, car le jeu ne se joue qu’à un seul joueur.

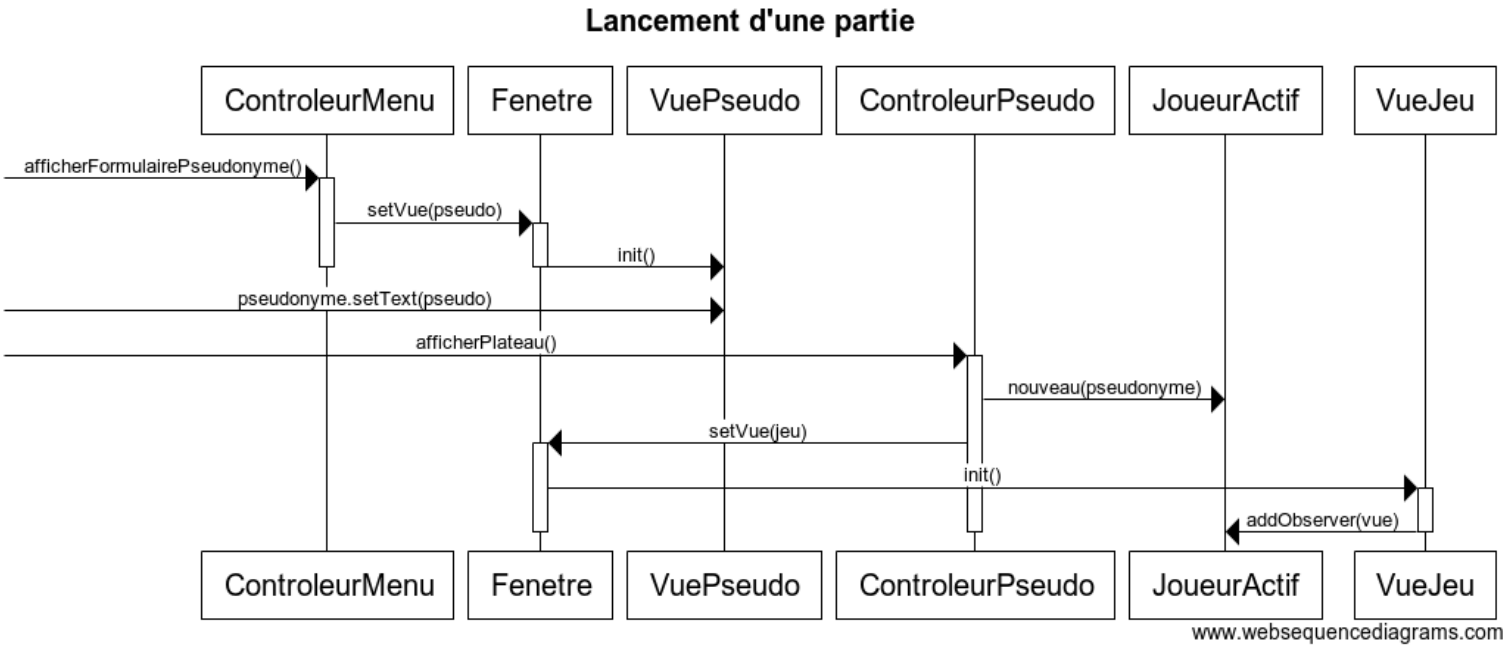
Lancement d'une partie

Comme on peut le voir dans le diagramme d'activités, pour lancer le jeu, le joueur doit d'abord entrer son pseudonyme.

Une fois dans le menu principal, le joueur peut cliquer sur le bouton « Jouer », qui change la vue pour le formulaire du pseudonyme et l'initialise.

Le joueur entre son pseudonyme, puis clique sur le bouton « Commencer la partie », qui lance la création d'un nouveau joueur portant son nom, et change la vue par celle du plateau de jeu avant de l'initialiser.

La vue du jeu va devenir un observateur de JoueurActif, et se mettra à jour à chacun de ses évènements (lorsqu'un tour vient d'être joué).

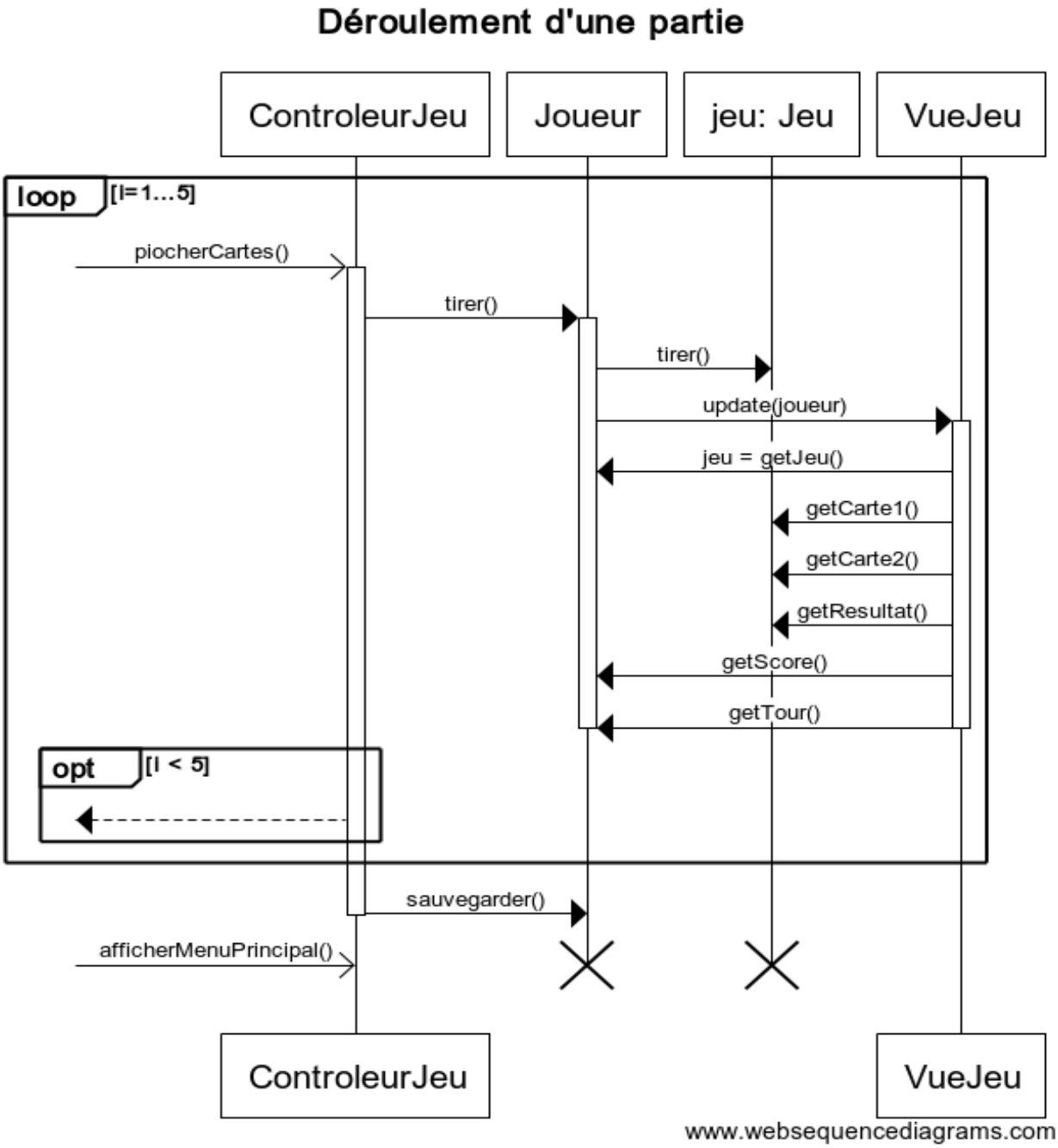


Déroulement d'une partie

Pendant sa partie, le joueur a le droit de piocher cinq fois deux cartes en appuyant sur le bouton « Tour suivant ». Cela lance l'action de tirage, traitée par le jeu, créant les valeurs du tirage (les deux cartes et le résultat). Le Joueur met alors à jour son score et incrémente son nombre de tours.

Le jeu va indiquer à tous ses observateurs qu'un nouveau tirage a eu lieu. La vue du jeu va alors se mettre à jour, en récupérant les nouvelles informations à afficher.

Au cinquième tour, le contrôleur empêche le joueur de rejouer en supprimant le bouton « Tour suivant » et demande une sauvegarde du record. Le joueur n'a plus que la possibilité de retourner au menu principal.



Conclusion

Ce projet nous a permis d’appliquer les méthodes d’analyse et de modélisation logicielles (UML) vues en cours.

Grâce à ces techniques, nous avons pu nous consacrer davantage sur la conception plutôt que sur le code, rendant le logiciel plus fiable et nous économisant du temps en évitant certaines erreurs.

Bien que le cahier des charges soit entièrement satisfait, nous pouvons encore améliorer certains points notamment :

- l’accessibilité, en rendant le jeu accessible aux non-voyants par exemple ;
- la performance, qui n’a pas pu être étudiée à cause de contraintes de temps. Comme le jeu ne nécessite pas beaucoup de ressources et qu’elles sont gérées par Java, le jeu est fluide.
- La compétition, en facilitant la comparaison des scores avec par exemple un classement des joueurs en ligne.