Fatih
Bozlak
1503001522G
L2 Informatique

Programmation orienté objet

Projet final : Jeux de carte

Table des matières

Présentation du projet **2**

Contexte **3**

Organisation et fonctionnement **4**
- Classes principales 4
  - Maitre du jeu 4
  - Joueur 4
  - Carte 4
  - Paquet de carte 4
  - AccueilControlleur 4
  - FenetrePrincipaleControlleur 5
- Maquette graphique 5
- Diagramme de classe 5
- Comment jouer 6
- GitHub 7

Problèmes rencontrés **8**

Pistes d'amélioration **9**

Conclusion **10**

Présentation du projet

Ce projet a été réalisé pour valider le cours de Programmation Orientée Objet enseigné par Monsieur Fabien Poirier, dans le cadre du cursus de Licence 2 en informatique.

Le but du projet était de créer un jeu de bataille de cartes, en utilisant des cartes de type Poker. Le jeu se déroule entre deux joueurs : un utilisateur humain et un adversaire contrôlé par l'ordinateur, que nous appelons ici "IA" (pour Intelligence Artificielle). Cependant, ce terme "IA" est utilisé uniquement pour désigner l'adversaire contrôlé par l'ordinateur et ne signifie pas que l'ordinateur utilise une véritable intelligence artificielle telle que nous la connaissons aujourd'hui. En réalité, ce joueur ne dispose d'aucune capacité d'apprentissage ou d'adaptation complexe.

Le jeu de bataille est un jeu de cartes traditionnel très simple, généralement joué par deux personnes avec un jeu de 52 cartes (dans notre projet cela pourra être jouer avec moins de cartes). Une partie se déroule de la façon suivante avec 52 cartes :

- **Distribution des cartes :** Pour commencer, le maitre du jeu mélange le paquet de cartes et distribue ensuite la totalité des cartes entre les deux joueurs, faces cachées, de sorte que chaque joueur possède 26 cartes ;
- **Jeu des cartes :** Chaque joueur prend la carte du dessus de son paquet et la place face visible au centre de la table (dans notre cas nous appelons ce centre le panier) ;
- **Comparaison des cartes :** Celui qui a posé la carte de valeur la plus élevée (l'as étant la plus forte et le 2 la plus faible) récupère les deux cartes et les place sous son paquet.
- **Bataille :** Si les deux cartes ont la même valeur, il y a *bataille*. Chaque joueur pose alors une carte face cachée puis une carte face visible. Celui qui pose la carte face visible de plus haute valeur remporte toutes les cartes posées lors de la bataille. Si les deux nouvelles cartes sont à nouveau de valeur égale, la bataille continue de la même manière jusqu'à ce qu'un joueur ait une carte de plus grande valeur.
- **Fin du jeu :** Le jeu se termine lorsque l'un des joueurs a collecté toutes les cartes, ou si les joueurs conviennent de mettre fin au jeu après un certain temps ou un certain nombre de tours. Le gagnant est le joueur qui a le plus de cartes. Pour notre projet, le jeu continu jusqu'à qu'un joueur n'a plus de carte (ou les deux joueurs n'ont plus de carte).

Le jeu de bataille ne requiert aucune stratégie particulière car c'est principalement un jeu de hasard.

Contexte

Ce projet a été développé dans un environnement Windows. Toutefois, la portabilité de l'application ne devrait pas être un problème grâce à la nature cross-platform de Java. En effet, un des avantages clés de la programmation en Java est la capacité de créer des applications qui peuvent être exécutées sur divers systèmes d'exploitation.

Ainsi, il sera possible de générer un fichier JAR (Java Archive) à partir de ce projet, qui sera exécutable sur tout système d'exploitation, qu'il s'agisse de Windows, macOS, Linux ou autre. Le seul prérequis est que l'utilisateur du système dispose d'une version compatible de la machine virtuelle Java (JRE - Java Runtime Environment).

La machine virtuelle Java est une composante essentielle qui interprète le code byte Java (contenu dans les fichiers JAR) pour le système d'exploitation hôte. Elle sert d'interface entre le code Java et le système d'exploitation, ce qui permet à une application Java d'être indépendante de la plateforme sur laquelle elle est exécutée. Cette indépendance par rapport à la plateforme est l'un des piliers du langage Java, qui est souvent résumé par le slogan "Ecrire une fois, exécuter partout" (Write Once, Run Anywhere).

La version minimal de la JRE à utiliser et la version 17.

Pour faciliter le développement de ce programme, j'ai opté pour l'utilisation d'un environnement de développement intégré (IDE) afin d'organiser plus efficacement la structure du projet et de bénéficier de fonctionnalités avancées. Dans ce cas, j'ai choisi d'utiliser IntelliJ, un IDE populaire pour le développement Java.

Dans l'ensemble, l'utilisation d'IntelliJ a contribué à améliorer mon efficacité et ma productivité lors du développement de ce projet. Il m'a permis de me concentrer davantage sur la logique du programme plutôt que sur les tâches de gestion de projet, ce qui a conduit à un développement plus fluide et une meilleure qualité du code.

Pour garder une trace de mes différentes versions du projet, y compris les obstacles rencontrés et les retours en arrière effectués, j'ai utilisé l'outil de contrôle de version Git.

Organisation et fonctionnement

Classes principales

Maitre du jeu

Le programme du jeu repose entièrement sur une entité centrale que j'ai appelée le `MaîtreDuJeu`. Cette entité est représentée par une classe Java et elle joue le rôle de l'autorité suprême dans le déroulement du jeu. Le Maître du Jeu est responsable de la gestion de toutes les règles et de la logique du jeu.

Le Maître du Jeu assume plusieurs responsabilités clés dans le jeu. Tout d'abord, il est chargé de la distribution des cartes aux joueurs, il est responsable des cartes jouées par les joueurs. Lorsque les joueurs posent leurs cartes, le Maître du Jeu détermine quelle carte est la plus forte. De plus, le Maître du Jeu gère également le panier, qui est l'endroit où les cartes sont accumulées lors des batailles.

Joueur

La classe `Joueur` représente un joueur dans un jeu de cartes. Elle contient les informations telles que le pseudo du joueur, son identifiant unique et la liste des cartes qu'il détient. Elle propose des méthodes pour jouer une carte, recevoir une carte.

Carte

La classe `Carte` représente une carte dans un jeu de cartes. Chaque carte a une valeur et une couleur. Elle hérite de la classe ImageView de JavaFX pour permettre l'affichage graphique de la carte.

Paquet de carte

La classe `PaquetDeCarte` représente un paquet de cartes dans un jeu de cartes. Elle permet de créer, manipuler et obtenir des informations sur le paquet de cartes, telles que retirer une carte, mélanger les cartes et accéder à la liste des cartes.

AccueilControlleur

La classe `AccueilControlleur` est un contrôleur JavaFX responsable de la gestion de la fenêtre d'accueil du jeu. Elle gère les interactions de l'utilisateur, valide les entrées saisies et permet de passer à la fenêtre principale du jeu.

FenetrePrincipaleController

La classe `FenetrePrincipaleControlleur` est un contrôleur JavaFX responsable de la logique de l'interface graphique de la fenêtre principale du jeu de bataille. Elle contrôle les actions des joueurs, la distribution des cartes, les animations et les événements du jeu.

Maquette graphique

Pour obtenir une perspective plus globale, j'ai conçu un modèle réduit qui illustre le design du jeu.

En effet, la création d'une maquette m'a permis de visualiser les différentes composantes du jeu, leur agencement, mais aussi l'expérience utilisateur que je souhaite offrir. Cette maquette, matérialisant le jeu dans ses grandes lignes, s'est révélée être un outil précieux dans la phase de développement, me permettant d'identifier rapidement les améliorations à apporter ou les problèmes potentiels à résoudre. C'est donc un véritable atout qui favorise une meilleure compréhension de la structure du jeu et une vision claire de l'objectif final que je vise.

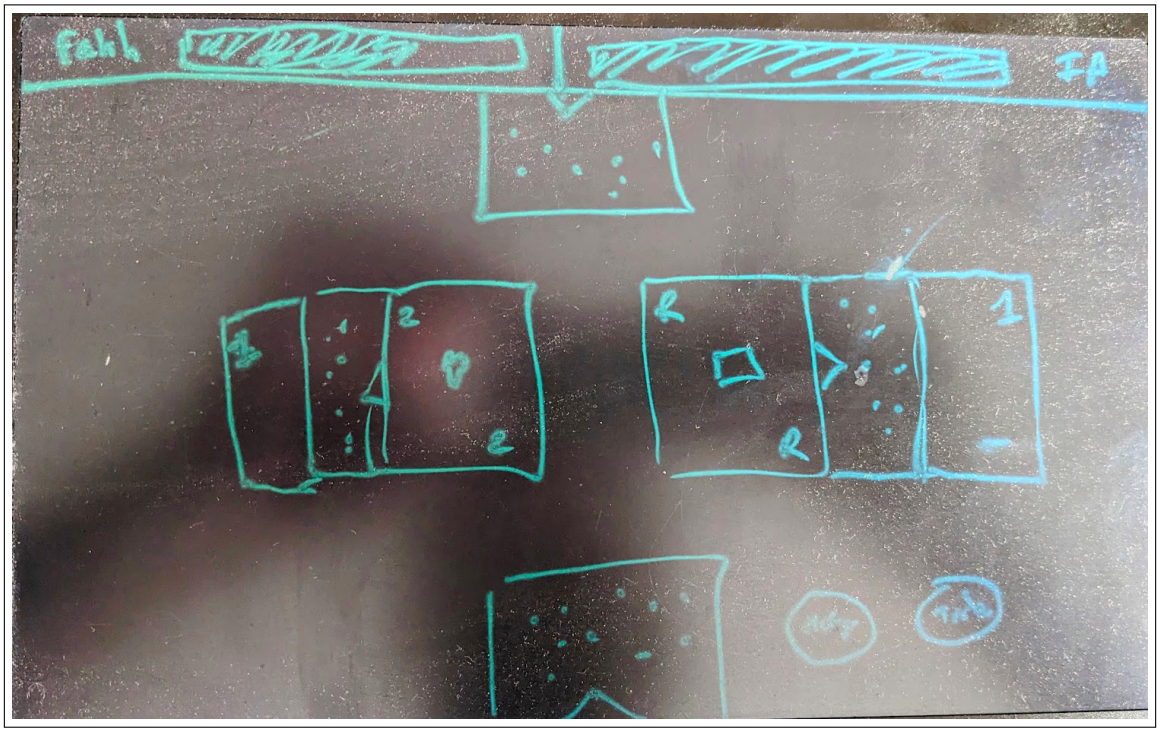
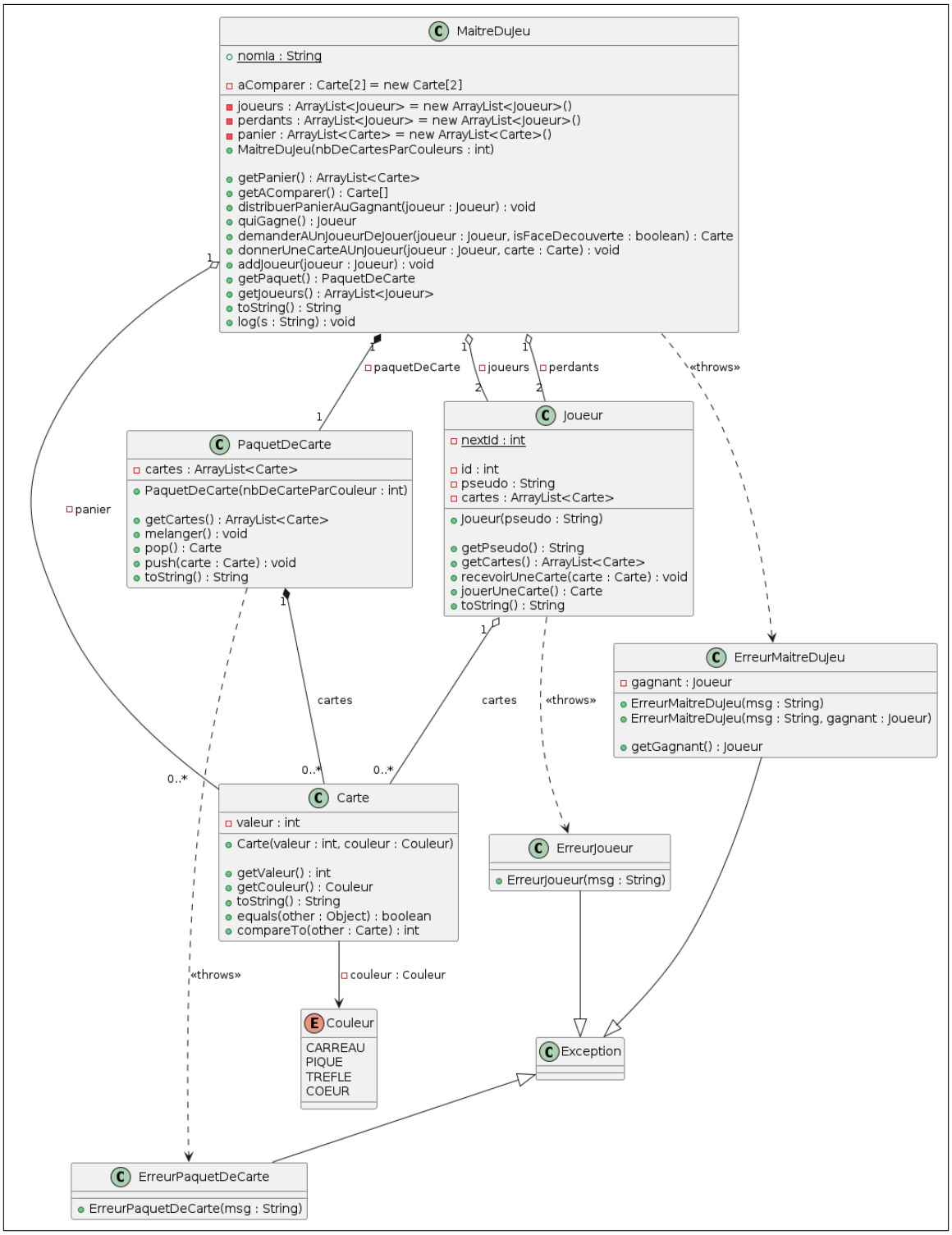


Diagramme de classe

Voici le diagramme de classe que j'ai utilisé pour développer ce programme. Il faut noter, cependant, que ce diagramme de classe ne prend pas en considération l'usage de la bibliothèque JavaFX.



Comment jouer

Pour entrer dans le jeu, le processus est simple : vous devez d'abord indiquer un pseudonyme et spécifier le nombre total de cartes avec lesquelles vous souhaitez jouer. Cela donne un certain contrôle sur la complexité et la durée de la partie.

Une fois ces informations fournies, le maître du jeu se charge de la distribution

des cartes, assurant ainsi un tirage équilibré et aléatoire. Chaque joueur reçoit donc son lot de cartes, prêt à se lancer dans le jeu.

Pour jouer votre carte, il suffit simplement d'appuyer sur la touche espace. Cela rend le jeu intuitif et facile à comprendre, tout en gardant un certain rythme. Les joueurs n'ont pas à se préoccuper de commandes compliquées, ce qui permet de se concentrer pleinement sur la stratégie et le plaisir du jeu.

GitHub

`https://github.com/bobylatruffe/jeuDeCarte`

Problèmes rencontrés

J'aimerais évoquer certains défis majeurs que j'ai rencontrés lors de la réalisation de ce projet, en particulier la programmation et la gestion des animations. Ce problème a requis de ma part un investissement considérable pour apprendre à concevoir mes propres événements. C'est cette phase d'apprentissage qui m'a permis d'appréhender de manière plus approfondie la programmation événementielle, une technique de programmation qui est devenue centrale dans mon projet.

Par ailleurs, un autre obstacle significatif fut l'initiation à la programmation d'interfaces graphiques. Cela a nécessité de comprendre les bases de la conception et de l'interaction des interfaces utilisateur, ce qui n'a pas été une tâche aisée. Cependant, je suis maintenant capable de manœuvrer avec assurance dans le monde de JavaFX, un outil précieux pour le développement d'interfaces graphiques.

Aujourd'hui, je suis particulièrement fier d'avoir surmonté ces défis. Acquérir ces nouvelles compétences m'a renforcé dans la conviction que la création d'applications graphiques ne représente plus un obstacle insurmontable pour moi. Au contraire, je suis désormais confiant et prêt à aborder des projets de plus grande envergure dans ce domaine.

Pistes d'amélioration

Bien que j'aie mis beaucoup d'efforts dans la création de ce jeu, il reste des aspects à améliorer. À l'origine, j'avais prévu d'intégrer une fonctionnalité appelée "Tricheur". Cette option aurait permis à un joueur de voir ses cartes et de les jouer, ajoutant ainsi une nouvelle dimension stratégique au jeu.

Un autre point qui nécessite des ajustements concerne le "Gameplay". Dans l'état actuel du jeu, il est indispensable d'attendre la fin des animations avant de pouvoir jouer une carte. Sinon, un bug survient, ce qui peut perturber le déroulement de la partie. La fluidité du gameplay est un facteur essentiel pour assurer l'immersion du joueur et son plaisir de jeu.

De plus, il serait très intéressant d'ajouter une fonctionnalité de jeu en ligne. Cela permettrait aux joueurs de s'affronter à distance, augmentant ainsi le niveau de compétition et l'interactivité du jeu. C'est un ajout qui pourrait véritablement enrichir l'expérience de jeu et élargir la portée de ce programme.

Conclusion

Je tiens à exprimer ma fierté d'avoir mené à bien ce projet et le cours associé. Il est vrai que l'expérience acquise en réalisant des travaux pratiques diffère significativement de celle que l'on obtient en participant activement à la création d'une application concrète.

Au cours de ce projet, j'ai eu l'opportunité de mettre en pratique les connaissances que j'ai acquises lors des cours. Plus encore, j'ai appris à me débrouiller par moi-même, ce qui m'a permis d'acquérir de nouvelles compétences et d'explorer de nouvelles techniques. Cela a été particulièrement vrai dans les domaines de la programmation événementielle et graphique, où j'ai dû dépasser ce que j'avais appris initialement pour mener à bien ce projet.

Bien que le jeu que j'ai développé puisse paraître basique et simpliste, je crois fermement que la manière dont il a été codé sert de fondation solide pour des améliorations futures. De plus, ces améliorations pourront être apportées sans nécessiter de modifications majeures du code existant. N'est-ce pas là l'essence même de la programmation orientée objet ? C'est-à-dire, créer du code qui est non seulement fonctionnel, mais également facilement réutilisable et adaptable à de nouvelles situations ? Cette expérience a vraiment renforcé mon appréciation pour cette approche de la programmation.