BARROS Matheus, **BERTOLDI** Gustavo, **CADER** Matias, **FRANCO** Carlo.

RAPPORT PROJET POKER

▪ Présentation du projet

Pour notre projet, nous avons décidé de créer un jeu de poker. Après avoir comparé nos différentes idées et leurs avantages et difficultés, nous avons finalement décidé de faire un jeu de poker entre un joueur humain et 5 « joueurs ordinateurs », avec des niveaux d'intelligence différents. Notre but était de présenter le jeu, avec une interface graphique intéressante et simpliste et avec un algorithme **original** et optimisé.

▪ Description de l'algorithme

Pour jouer au poker, il faut principalement des joueurs, un paquet de cartes, et une table. Ainsi, les classes qui composent notre code sont :

• **Carte :** Chaque carte a pour attribut sa valeur (*int*), sa couleur (*char*), un icone pour l'affichage dans l'interface graphique (*ImageIcon*) et un chemin pour récupérer l'archive contenant son icone en forme d'image (String). Cette classe implémente l'interface *Comparable* qui nous a permis de trier facilement des listes de cartes. Une méthode description retourne la description textuelle de la carte au pluriel ou singulier d'après un *boolean* fourni en paramètre.

• **Paquet :** La classe paquet crée et remplie une liste avec les 52 cartes du jeu. Son seul attribut est une *LinkedList* de cartes.

• **Hand :** La classe Hand fournit, entre autres, le calcul de la puissance de l'ensemble de cartes du joueur dans des différents moments de la tournée, bien comme cet ensemble de cartes dans des différents moments.

• **Range :** La classe Range a pour but analyser l'avancement de la puissance des mains d'un joueur lors du déroulement des tours de paris d'une même tournée. Sa puissance est donc stockée comme un caractère en fonction des cartes dont le joueur dispose.

• **Joueur :** Les joueurs possèdent les attributs qui définissent son rôle dans chaque tournée (dealer, big blind, small blind), son hand, son argent, son intelligence et des autres attributs secondaires qui assurent le déroulement du jeu. La classe contient la méthode qui fait le joueur jouer son tour, en fonction de l'intelligence dans le cas d'un ordinateur ou en fonction de l'action du joueur humain.

• **Intelligence :** Cette classe est la responsable pour la prise de décisions des joueurs ordinateurs auxquels elle appartient. Elle a un **niveau**, correspondant au niveau du joueur. Les différents niveaux de joueurs imposent des différents **limitesPlayRandomly**. Ces derniers sont d'autant plus grands que les niveaux des joueurs sont faibles. Ils sont donnés en étant des *doubles* qui seront utilisés pour décider la forme avec laquelle le joueur va agir dans chaque tournée. Cela est fait ainsi : au début de chaque tournée, on attribue des doubles pris au hasard à **typeDecisionTournée**, qui sera comparé avec **limitesPlayRandomly**, s'il est plus petit que la limite, il jouera selon **decisionAleatoire**, sinon, il joue selon **decisionNiveau2**, qui prend en compte énormément de

paramètres pour la prise de décisions. Ces décisions sont données comme des entiers, qui correspondent : -1 (Fold), 0 (Check), 1 (Call), **decision**>1 (Raise, **decision** correspond à la valeur de pari désirée). Pour déterminer combien le joueur va parier, *intelligence* dispose d'un tableau statique de **bets** possibles, qui sont mises à jour à chaque fois que le joueur est amené à jouer.

• **Jeu :** c'est la classe qui permet la gestion, l'organisation et le développement du jeu.

• **FenetreJeu :** permet de représenter le jeu sous forme d'interface graphique.

• **Ecouteur :** rend les boutons fonctionnels, pour la prise de décision du joueur humain {Raise, Call, Check) et pour passer à la tournée suivante.

• **Node :** permet de refermer la « table » dans la LinkedListCirculaire.

• **LinkedListCirculaire :** cette classe nous a beaucoup facilité le fait de créer un jeu fluide, avec l'optimisation du passage entre tournées.

Grâce à une véritable réflexion pendant l'étape de préconception, nous sommes bien partis dès le début. En effet, parmi les classes citées ci-dessus, les classes Node et LinkedListCirculaire n'étaient pas prévues comme telles dès le début, mais correspondent à la classe « Table » initialement prévue. D'une façon similaire, la classe Range, qui n'était aussi pas prévue, complémente l'Intelligence. En outre, pendant le développement du code, nous avons compris que certaines des classes envisagées au début n'étaient pas utiles : c'est le cas d'une class Table, mentionnée ci-dessus et une classe Banque, qui allait être responsable de gérer l'argent des joueurs, les paris, et le pot.

▪ Diagramme UML

Ainsi, veuillez trouver le diagramme UML de notre code final, avec les classes listées ci-dessus, dans le lien suivant :

https://www.draw.io/#G14SdmhrzN6Rsq8raPzCghsmzjNtUSiAvg

Pour y accéder, veuillez utiliser l'identifiant suivant.

Adresse mail : pokermaster.amerinsa@gmail.com

Mot de passe : Amerinsa

▪ Performance du groupe

- Eléments récupérés

Dès le début, on a envisagé écrire un code le plus original possible, donc majorité du code est totalement original, sauf quelques petits points qu'on détaillera. Effectivement, pour réussir à coder notre jeu de poker, il nous a fallu repousser nos limites et acquérir de nouvelles connaissances. Mais aussi, à certains moments, nous avons eu besoin d'une fonction spécifique, que nous avons récupéré sur internet. Ainsi, les éléments récupérés tous faits sont :

- La méthode pour redimensionner l'image des cartes pour créer l'*ImageIcon*.

- L'idée de la *LinkedListCirculaire* a aussi été presque totalement inspirée d'un code trouvé sur internet, mais on l'a adapté pour qu'il atteigne nos besoins spécifiques.

- **Principaux problèmes résolus**

Sans doute, nous avons rencontré de diverses difficultés pendant la réalisation d'un algorithme si complexe. Les problèmes les plus intéressants que nous avons résolus sont les suivants.

L'idée de créer les classes *LinkedListCirculaire* et *Node*, qui traduisent la complexité à laquelle on s'est confronté lorsqu'on codait le jeu. Dans un jeu de poker, les joueurs s'assoient autour d'une table circulaire et, à chaque tournée, les rôles des joueurs évoluent dans le sens de l'horloge. Donc, une *LinkedList* (telle qu'on l'a connaissait avant) rendrait le code très complexe dû aux plusieurs conditions qu'on aurait à penser lors du parcours de la liste. Ainsi, la classe *LinkedListeCirculaire* non seulement a résolu tous ces problèmes, mais a aussi rendu le code beaucoup plus lisible et simple. L'idée derrière cette liste est la même derrière une *LinkedList*, sauf que le premier et dernier *node* sont toujours liés. Lors du codage de cette classe, on a pu créer des méthodes spécifiques pour notre jeu, comme un *getter* pour un joueur dealer, par exemple.

Un autre problème qu'on a rencontré c'était la synchronisation des tâches à exécuter, ce qu'on a pu résoudre en utilisant les *Threads.* On a cherché le fonctionnement de cet outil Java et on l'a bien implémenté dans notre code. C'est aussi intéressant de mentionner qu'on a cherché à maîtriser les expressions lambda en Java, ce qui nous énormément facilité le travail avec les plusieurs listes qu'on a dû gérer.

Enfin, lors de la création de l'interface graphique, on s'est rendu compte qu'il nous fallait d'autres *layouts* pour arriver au résultat qu'on envisageait. On a choisi le *GridBagLayout*, qui nous donne une flexibilité beaucoup plus grande que les *layouts* qu'on connaissait et nous a permis d'arriver à un résultat idéal.

- **Organisation**

Même avant le début du confinement, nous avions mis en place un système de travail collaboratif grâce à l'utilisation de la plateforme Github. En effet, deux d'entre nous connaissaient déjà ce mode de travail et donc le confinement n'a pas vraiment affecté notre progrès dans ce sens-là, même si le fait de ne pas discuter en personne est moins avantageux. Avec le cours du temps, les autres deux membres du groupe se sont habitués au fonctionnement de Github. Également, même en dehors des heures de cours, nous nous communiquions de manière régulière, principalement avec des messages à travers l'application WhatsApp, mais aussi avec des appels grâce à Discord quand il nous fallait faire le point. En moyenne, on discutait du projet chaque 2 ou 3 jours : nous étions tous les quatre passionnés et engagés pour créer le meilleur programme possible.

- **Semainier**

Semaine	Travail
09/03/2020	Travail de préconception. Réfléxion sur les classes nécessaires et sur les interfaces graphiques souhaitées. Distribution des tâches. Rédaction du premier rapport.
16/03/2020	Début de l'écriture des classes basiques: Carte, Joueur, Jeu, Paquet, Fenetre, Jeu. Mise en place d'un affichage de base: première interface graphique.
23/03/2020	Approfondissement sur la classe Hand et la manière de comparer les Hand entre elles. Création d'une LinkedListCirculaire pour faciliter la participation des joueurs lors des tournées.
30/03/2020	Approfondissement sur l'interface graphique, avec un nouveau layout, les images des cartes, et les boutons définissant les actions du joueur humain. Mise en place de la distribution des cartes.
06/04/2020	Amélioration des codes de gestion du jeu, avec l'implémentation de nouvelles méthodes.
13/04/2020	Création des fenêtres qui composent le Menu du jeu. Amélioration des niveaux d'intelligence des ordinateurs.
Vacances	Beaucoup de tests du jeu, pour résoudre les éventuels bugs, en optimisant certaines méthodes. Rédaction du rapport et création du diagramme UML.
04/05/2020	Semaine Bonus, utilisée pour améliorer des détails de l'interface graphique et modifications sur l'intelligence.
12/05/2020	Rendu final et soutenance.

- **Forces et faiblesses**

➔ Sans doute, le principal atout de notre projet c'est le fait que le jeu de poker est totalement fonctionnel. Nous avons bien progressé chaque semaine, ce qui nous a permis d'améliorer la performance du code et d'éliminer les bugs rencontrés à chaque étape. En effet, il s'agit d'un jeu complexe, difficile à coder, et dont le codage dépassait nos connaissances immédiates. Néanmoins, notre engagement vis-à-vis du projet nous a permis de repousser nos limites et d'atteindre le résultat souhaité. Ainsi, nous avons tous les quatre acquis de nouvelles connaissances en programmation pendant ce projet.

➔ Nous aurons voulu travailler plus l'interface graphique. En effet, notre version est la plus efficace et non pas la plus attractive visuellement. De plus, même si nous sommes satisfaits avec les niveaux d'intelligence obtenus, nous aurons aimé d'aller encore plus loin dans cet aspect.

Notre chef de projet, Matheus BARROS, a très bien organisé et coordonné le travail à réaliser. Il nous a incité et motivé à surmonter les obstacles rencontrés. En plus d'être un excellent programmeur, il a été un excellent leader.

Taux d'implication : BARROS Matheus (32.5%) ; BERTOLDI Gustavo (32.5%) ; CADER Matias (20%); FRANCO Carlo (15%).

➔ Ainsi, par tout ce qui précède, et surtout conscients des aspects qui pourraient être améliorés, nous considérons que nous méritons une note de 17,5/20.

▪ Bibliographie

➔ **LinkedListCirculaire** adaptée de : https://www.baeldung.com/java-circular-linked-list

ANNEXE

Fiche d'auto-évaluation des projets informatique 19-20

Identification du groupe

Nom du groupe **GROUPE POKER**

Valorisation des compétences des différents membres

- *Nom 1 :* **BARROS (32.5%)**
- *Nom 2 :* **BERTOLDI (32.5%)**
- *Nom 3 :* **CADER (20%)**
- *Nom 4 :* **FRANCO (15%)**

Évaluation du chef d'équipe

➔ Notre chef de projet, Matheus BARROS, a très bien organisé et coordonné le travail à réaliser. Il nous a incité et motivé à surmonter les obstacles rencontrés. En plus d'être un excellent programmeur, il a été un excellent leader.

Auto-notation du projet

Note auto-attribuée : 17,5 /20

Justification courte (forces et faiblesses) : (à part ce qu'on a écrit dans le rapport)

➔ Notre production finale est un jeu de poker complètement fonctionnel. Il s'agit d'un code complexe, et pourtant le jouer est facile et dynamique.

➔ Tous les quatre nous avons acquis de nouvelles connaissances en programmation.

➔ En ce qui concerne l'interface graphique, nous aurons voulu ajouter les jetons et une distribution des cartes plus agréable visuellement.

Commentaires libres sur votre projet

Le codage du PokerInsa nous a posé beaucoup de difficultés. Nous ne pensions pas que le jeu serait si complexe, mais nous nous sommes rendu compte de sa complexité le long du projet. De ce fait, il est extrêmement gratifiant de pouvoir rendre le projet fonctionnel face à des obstacles rencontrés dans le codage et dans la vie externe, dans un moment de pandémie avec ses conséquences.