UNIVERSITÉ DE PARIS
Université Paris Diderot

RAPPORT DU PROJET LOA (aka c++)

de 1^{e} année de Master en Informatique
parcours IMPAIRS + DATA
soutenu par

Djamel ALI

et

Mohand Yacine OUNNOUGHENE

le 13 Janvier 2021

(Projet fait en 1^{e}*semestre* de M1)

Réalisation d'un framework autour des jeux de cartes

L'équipe enseignante :
Cours :
M. Jean-Baptiste YUNÈS
Travaux Dirigés :
Mme. Anne MICHELI
M. Yan JURSKI
M. Simon FOREST

Année universitaire 2020-2021

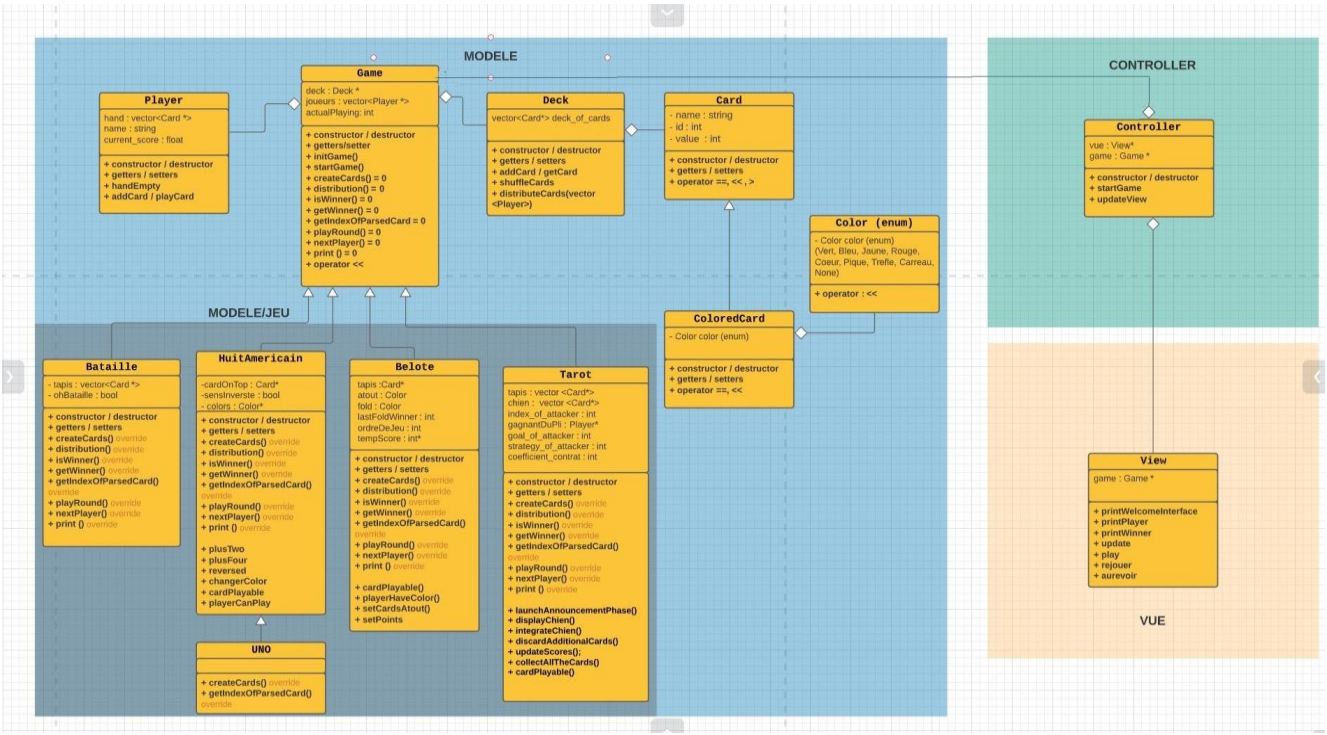
SOMMAIRE:

1. Introduction:

Le but concrêt est de réaliser des applications permettant de jouer à plusieurs jeux de cartes.

En analysant l'ensemble des jeux proposés, on a remarqué qu'il est possible de généraliser plusieurs concepts (et donc de factoriser beaucoup de code).

2. Diagramme UML :



3. Quleques explications :

3.1 La classe Game:

La classe Game généralise quasiment tout jeu de cartes qu'on pourrait imaginer car si on regarde bien les différents champs (attributs et méthodes) qu'elle contient, on remarque qu'effectivement un jeu de cartes (peu importe lequel) a toujours un paquet de cartes (*Deck*) qui contient au départ toutes les cartes qu'on va utiliser pour jouer, une liste de joueurs (*joueurs*) (au minimum 2, et un joueur peut être soit une machine soit un humain), et à tout moment durant la partie un joueur qui est en train de jouer (*actualPlaying*) et tous les autres attendent leurs tours.

La classe *Game* est la classe principale de ce framework, elle contient des méthodes virtuelles pures (qu'il faut implémenter dans les sous-classes de Game), elle se compose (par des relations d'agrégation) d'autres classes (*Player*, Deck).

Tout nouveau jeu de carte dans ce framework doit hériter de cette classe Game.

Nous avons choisi d'implémenter les jeux suivants :

1. La Bataille.
2. Le Huit Americain et Uno (Uno hérite de 8 Americain).
3. La Belote.
4. Le Tarot.

3.2 Les classes Player, Deck, Card:

La classe game possède des instances de Player et Deck,

Ces 2 derniers représentent un joueur (avec une liste de cartes qui représente sa main) et le paquet de cartes respectivement (la pioche).

La classe Card représente les cartes, nous avons 2 types, les classes normales (dites simples) et les cartes avec couleur (utilisation de l'énumération Color).

4. Le Pattern utilisé :

Nous avons utilisé le pattern MVC pour l'implémentation de notre framework:

Une classe Vue:

Cette classe interagit avec l'utilisateur (e.g : afficher des messages, récupération de l'index d'une carte à jouer.

En plus de l'affichage des informations d'une partie de jeu.

Une classe Controller:

Cette classe agit comme lien entre la vue et le Model, elle met à jour l'affichage de la Vue (Les cartes qui sont sur le tapis de jeu, les mains des joueurs, leurs scores, …) et ainsi met à jour le Model en jouant une carte par exemple grace à son indice (index).

Une classe Model:

Cette classe représente une partie de jeu de cartes (avec ses joueurs, son tapis de cartes …).

5. Compilation et exécution:

Un fichier Readme est disponible expliquant toutes les étapes pour compiler et exécuter le projet utilisant le fichier Makefile.

Il vous suffit de :

- Exécuter la commande **'make** ' dans le repertoire courant pour compiler le code.
- Taper la commande '**./main**' afin de lancer le jeu.
- Il y a aussi la commande '**make clean**' qui permet de nettoyer les fichiers intermediaires (*.o) et le fichier d'exécution.

6. **Informations Générales:**

- Lien au dépôt git contenant la version finale du projet :

https://gaufre.informatique.univ-paris-diderot.fr/alid/card-games-framework