# Projet C++
# Compte Rendu

---

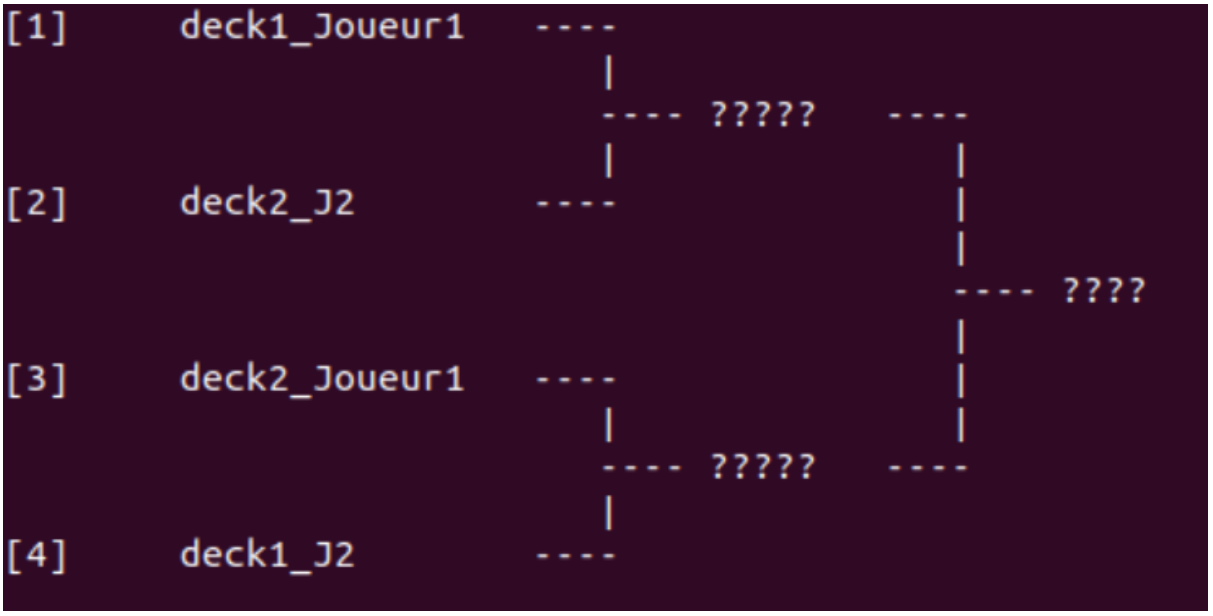
Mathieu Semenzato & Hugo Vouaux
EI-SE 4
2022-2023

**Sommaire :**

# Description de l'application

Le but de notre projet a été de réaliser un jeu de cartes à collectionner sur la coupe du monde. Le jeu que nous avons développé se joue à 2 joueurs sur un même ordinateur.

Les règles sont les suivantes : Une liste de cartes est donnée pour chaque joueur (le joueur 1 puis le joueur 2). La liste est créée aléatoirement à partir d'une base de données de cartes de chaque type (Pays, Booster, Sportif). Le but étant qu'à chaque démarrage du jeu (exécution du main), la liste donnée aux joueurs change. Par souci de réalisme, chaque carte a des caractéristiques proches de la réalité, ainsi certaines nations sont plus fortes que d'autres, les gardiens n'ont par exemple aucune influence sur les points d'attaque de votre équipe mais seront décisifs pour empêcher l'équipe adverse de marquer. A partir de cette liste, le joueur doit constituer 2 decks contenant chacun 1 carte pays parmi celles présentes dans la liste, 1 carte booster, 1 carte gardien ainsi que 4 cartes joueurs, qu'ils soient attaquant ou défenseur. Une fois que les 2 joueurs ont fait leurs 2 decks respectifs, des paris sont proposés. Au total, nous allons avoir 3 matchs comme le résume ce schéma :




Note : Pour un affichage optimal, le nom des decks ne doit pas dépasser 16 caractères.

Les 2 joueurs vont donc devoir parier sur une équipe pour chacun des matchs sans avoir vu les decks de l'adversaire, le but étant de deviner si notre deck est plus fort que celui de l'adversaire et de parier adéquatement (il est donc possible de parier contre son équipe si vous n'avez pas confiance en vous).

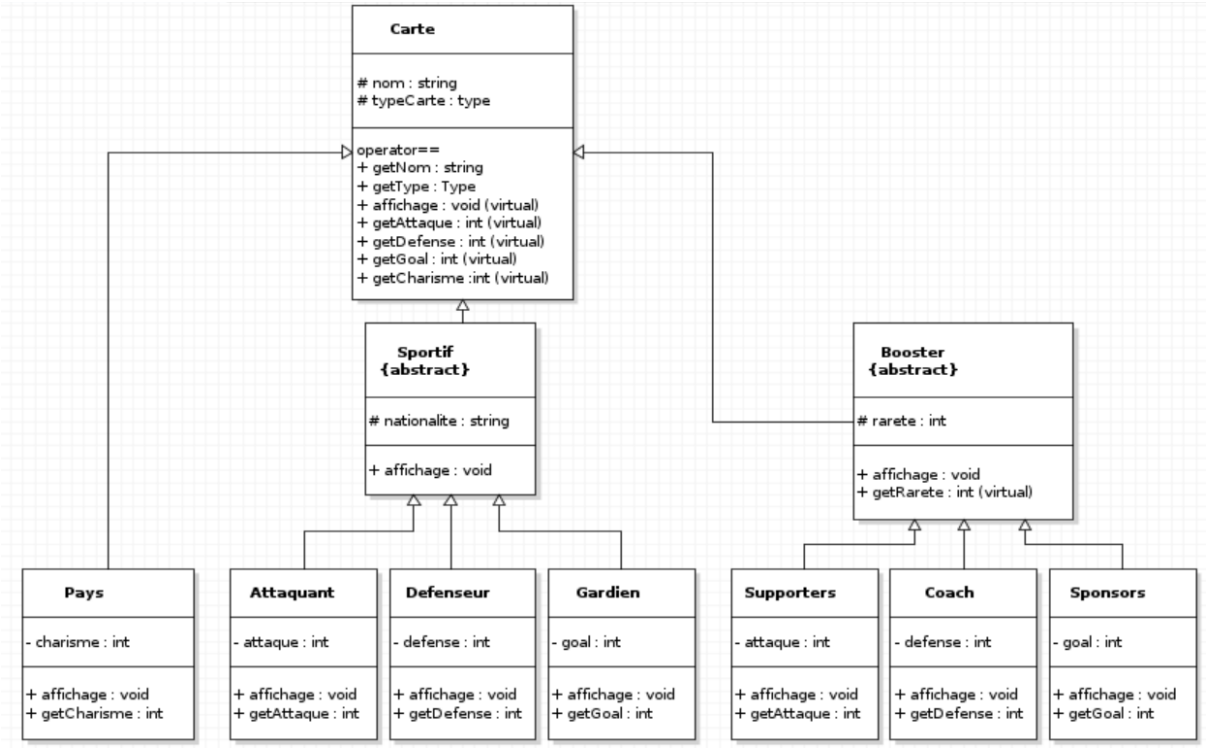
Pour chaque match, le deck gagnant (donc le joueur gagnant) est annoncé ainsi que les paris corrects. Chaque pari correct ou match gagné rapporte 1 point et 1 carte (pouvant être un pays, un sportif ou un booster) qui sera distribuée à la fin du championnat. Le nombre de buts marqués par chaque équipe est calculé en fonction de ses points d'attaques, ainsi que des points de défense et de goal de l'équipe adverse. Le vainqueur est alors celui qui a marqué le plus de buts.

Lorsque le tournoi est terminé, le gagnant est annoncé. Il est alors possible de remporter la victoire en gagnant tous ses matchs ou en réalisant les meilleurs pronostics. Vous pouvez ensuite décider de terminer la partie ou de continuer de jouer. Si vous continuez de jouer, les cartes viendront s'ajouter à la liste de départ, tandis que si l'on quitte le jeu les cartes gagnées seront perdues. Comme tout jeu de collection vous pouvez obtenir des cartes plus ou moins rares, le but est d'en avoir le plus possible.

Note : Tous les joueurs et entraîneurs de toutes les équipes n'ont pas été mis dans le jeu (à cause de leur nombre), si vous souhaitez ajouter des personnes particulières, il vous suffit d'ouvrir le document.txt correspondant et de l'ajouter. La carte sera automatiquement créée lors de l'exécution du programme. Le jeu possède déjà 191 cartes différentes, chacune étant unique.

# Présentation et description de l'UML

Voici dans un premier temps la partie de l'UML qui caractérise les différentes cartes :



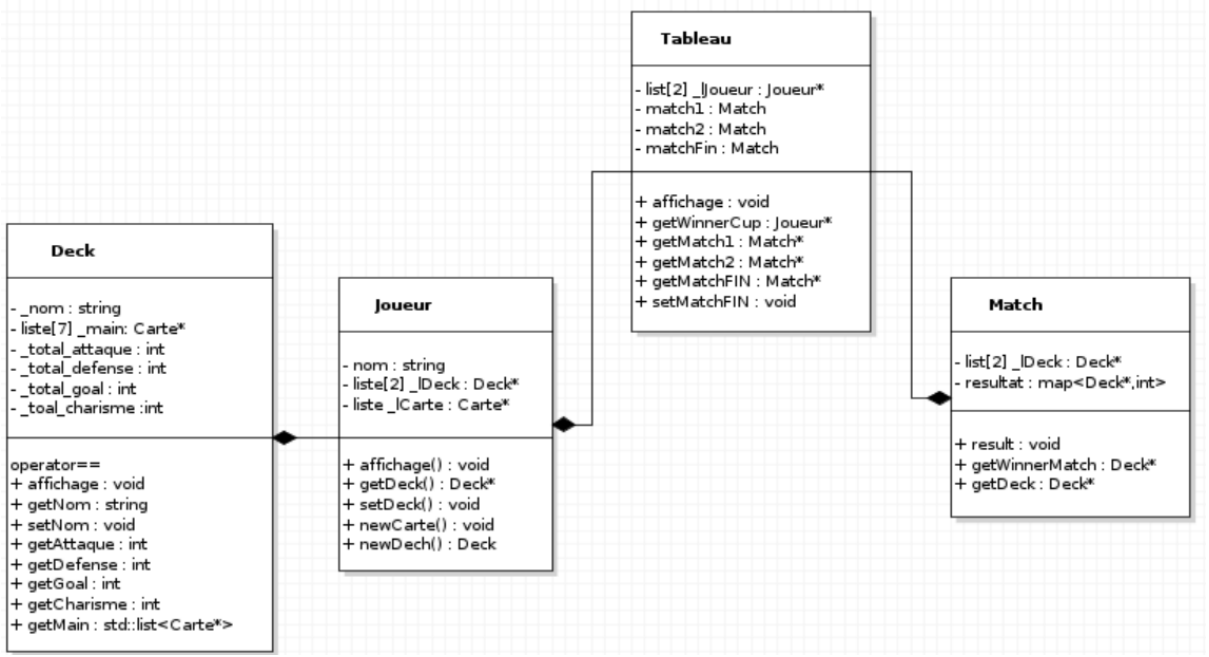

Les cartes sont deux 3 types : Pays, Sportif (attaquant, défenseur ou goal) ou Booster (supporteurs, coach ou sponsors). Chaque pays possède un attribut charisme, correspondant à leur classement FIFA actuel. Chaque sportif possède soit une qualité d'attaque, soit de défense, soit de goal, ainsi qu'une nationalité.De même pour les booster qui ont un indice de rareté à la place de la nationalité.

La classe Carte possède de nombreuses méthodes, dont getAttaque(), getDefense(), getGoal() et getCharisme() qui sont des méthodes virtuelles renvoyant la valeur du champ si

il existe, 0 sinon. Cela nous permet plus tard, lors de la création des Decks de ne pas se soucier du type de carte dont il s'agit. La fonction affichage() est elle aussi virtuelle est permet d'afficher les caractéristiques de chaque carte.

Enfin, nous avons redéfini l'opérateur == pour connaître lorsque deux cartes sont identiques, cela est utile pour éviter les doublons dans les deck et connaître si le joueur possède déjà la carte reçu en fin de partie pour pouvoir se moquer de lui.

Nous allons maintenant passer à la deuxième partie, qui comprends les joueurs participant à la compétition ainsi que leur Decks respectifs :




La classe Joueur possède un nom, une liste de deux Deck* (car les tableaux posaient problème lors de l'initialisation) ainsi que la liste de Carte* sur laquelle nous appliquons une règle de tri pour que les cartes apparaissent dans l'ordre suivant : Pays, Booster, Gardien, Attaquant puis Défenseur. Nous avons choisi cet ordre car il s'agissait du plus cohérent lors du choix des cartes de chaque Deck. La fonction affichage() permet de connaître toutes les cartes (triées) que possède le joueur. La fonction getDeck() renvoie un pointeur vers le deck choisi, newDeck() permet de créer les Deck un à un. Nous disposons d'ailleurs d'une surcharge de cette fonction pour s'assurer que le deuxième Deck ne contient pas de Carte du premier. Enfin la fonction newCarte() permet d'ajouter en fin de partie les cartes gagnées par le joueur à sa liste de Carte*.

La classe Deck possède les attributs suivants : nom, le liste de 7 Carte*, et la somme des attaques, défense, goal et charisme de ses cartes. La classe possède des accesseurs pour tous ses champs privés, ainsi qu'un setter pour le nom. Nous pouvons aussi afficher le Deck grâce à la fonction affichage(). Enfin, la classe redéfinit l'opérateur == facilitant ainsi la partie des paris.

La classe Match possède une liste des deux Deck qui s'opposent (nous n'utilisons pas de tableau à cause de problèmes lors de l'initialisation), ainsi que d'une map possédant comme clé des pointeurs vers les Deck, et en valeur le nombre de but marqué par chacun lors du match. La méthode result() permet de remplir la map en fonction des caractéristiques

de chaque Deck, getDeck() est un accesseur et getWinnermatch() renvoie un pointeur vers le Deck qui a marqué le plus de buts.

Enfin la classe Tableau regroupe les classes précédentes et dicte l'organisation du tournoi (ici en demi-finale puis finale).

# Fonction Main

La fonction du main.c est triple. Dans un premier temps (lignes 6 à 228), nous créons les bases de données des cartes en lisant les différents fichiers .txt. Si vous souhaitez d'ailleurs ajouter des cartes manquantes qui sont chères à vos yeux dans le jeu (ou vous même), il suffit d'ouvrir le .txt correspondant et d'y ajouter votre carte personnalisée. Nous créons ensuite les deux joueurs (en prenant plus ou moins en compte leur avis pour le choix du nom) en leur attribuant au hasard un certain nombre de cartes. Ils choisissent alors leur deux Deck qui s'affrontent lors du tournoi. Enfin sur chacun des Match, le main.c assure le traitement des paris et l'appel des bonnes fonctions d'affichage.

Par ailleurs, le fichier main.cpp contient les tests unitaires réalisés afin de vérifier le bon fonctionnement des classes créées. Pour l'utiliser, il faut décommenter la ligne 1 du main ainsi que la partie TESTCASE (ligne 8 à 118) et commenter le main du jeu (ligne 121 à 580).

# Mise en valeur des contraintes

Concernant les contraintes du projet, les 8 classes minimales et 3 niveaux de hiérarchie nous ont permis de bien séparer les différentes fonctions de nos cartes. Nous avons ainsi pu créer une classe spécifique pour chaque type et sous-type de cartes en définissant adéquatement l'affichage de ces cartes en prenant en compte leur différentes particularités : l'utilisation d'une fonction virtuelle pour l'affichage a ici trouvé son utilité, cette fonction étant constamment redéfini et n'étant pas utilisé par la classe Carte.

Le fait d'avoir plusieurs niveaux de hiérarchie nous a également été très utile pour la création du deck de cartes. En effet, la classe Carte étant la classe mère, il a été possible de créer dans la classe Deck un vecteur de Carte prenant des cartes Joueurs, Pays ou Boosters. Nous avons par ailleurs choisi de prendre une liste de Carte* pour la composition du Deck car il nous permet d'effectuer une fonction de tri rapide. Ce choix nous semblait le plus judicieux puisque nous n'avions pas besoin d'accéder aux cartes elles-mêmes par indice. Nous avons par ailleurs redéfini l'opérateur == pour les Carte et les Deck permettant ainsi d'empêcher les doublons et de faciliter la partie de traitement des paris.

Le code devant également fonctionner sur n'importe quel ordinateur, le fait de vérifier qu'il n'y ait aucune erreur avec Valgrind nous permet d'éviter certaines erreurs de compilation qui sont susceptibles de bloquer sur d'autres machines. L'utilisation de tests unitaires permet aussi de s'assurer étape par étape que les méthodes des classes et les classes dans leur globalité sont correctement codés. Le fait que les fonctions ne doivent

faire plus de 30 lignes permet aussi de repenser certaines fonctions afin qu'elles soient le plus simple possible. Or plus une fonction est simple et plus les risques d'erreurs sur le code diminuent.

Enfin concernant l'exécution du code, l'utilisation de build automatique est très appréciée pour compiler rapidement le code. Cela est autant utile dans la phase de développement du code que pour le résultat final. La règle clean est également utile lors du partage de document entre les membres du binôme afin d'éviter de partager les fichiers .o.

# Fiertés sur le projet

Ce projet a nécessité de nombreuses heures de travail, mais le rendu final rejoint les attentes que nous nous sommes fixées. Particulièrement, l'utilisation de fichiers afin de remplir les listes de cartes Pays, Joueurs, Booster comme fait pour le TP sur les molécules a été une façon astucieuse de remplir notre base de données de cartes de manière simple et efficace, tout en assurant un agrandissement simple de la base de donnée si on le souhaite. Celle-ci possède déjà les 32 pays participant à la coupe du monde 2022, 20 coachs, les 7 sponsors officiels, 75 attaquants (tous buteurs lors de la compétition sauf Giroud et Ronaldo), les 30 meilleurs défenseurs, 18 gardiens et 9 types de supporters. L'arborescence des matchs permet également un affichage esthétique des différents matchs à n'importe quelle étape de la compétition. Les différents affichages des cartes selon leur type est également très satisfaisant, car il permet d'apporter pour chaque type de carte ses spécificités et ainsi rendre le jeu plus complet. Cette partie a été également pratique pour se rendre compte de la puissance de la programmation objet comme la redéfinition et la surcharge de certaines méthodes. Nous avons ainsi pu implémenter la quasi-totalité des notions abordées en cours (sauf template) ce qui nous a aidé à mieux assimiler celles-ci.

# Procédure d'installation et exécution du code

Pour assurer la portabilité du code et à la suite de problème logiciel, l'affichage du déroulement du programme se fait dans le terminal. La procédure à suivre est donc très simple :

make (pour compiler)
./carte (pour lancer le programme)
make clean (pour supprimer les .o et l'exécutable)

Concernant la façon de jouer au jeu, la procédure est détaillée et extrêmement guidée en lançant l'exécutable et ne nécessite donc pas d'y revenir dans notre compte-rendu.

*Bon jeux à vous !* 😀