Rapport Final
Conception Orientée Objet

Projet 7 Wonders

Nos diagrammes ont été réalisés avec StarUML.

Groupe PD7WA

1. Point de vue général de l'architecture

a. Glossaire

Client : membre qui communique à travers un réseau entre plusieurs réseaux. Celui qualifié de client a pour rôle d'envoyer des requêtes, des demandes.

Serveur : membre qui communique à travers un réseau entre plusieurs réseaux. Le serveur, quant à lui, attend les requêtes du client afin de pouvoir y répondre.

Lanceur : il s'agit du lanceur de notre jeu, c'est un mécanisme qui va ainsi débuter l'exécution.

Sockets : il s'agit des interfaces de connexion, qui permet suite à une connexion de recevoir et de partager à son tour des données.

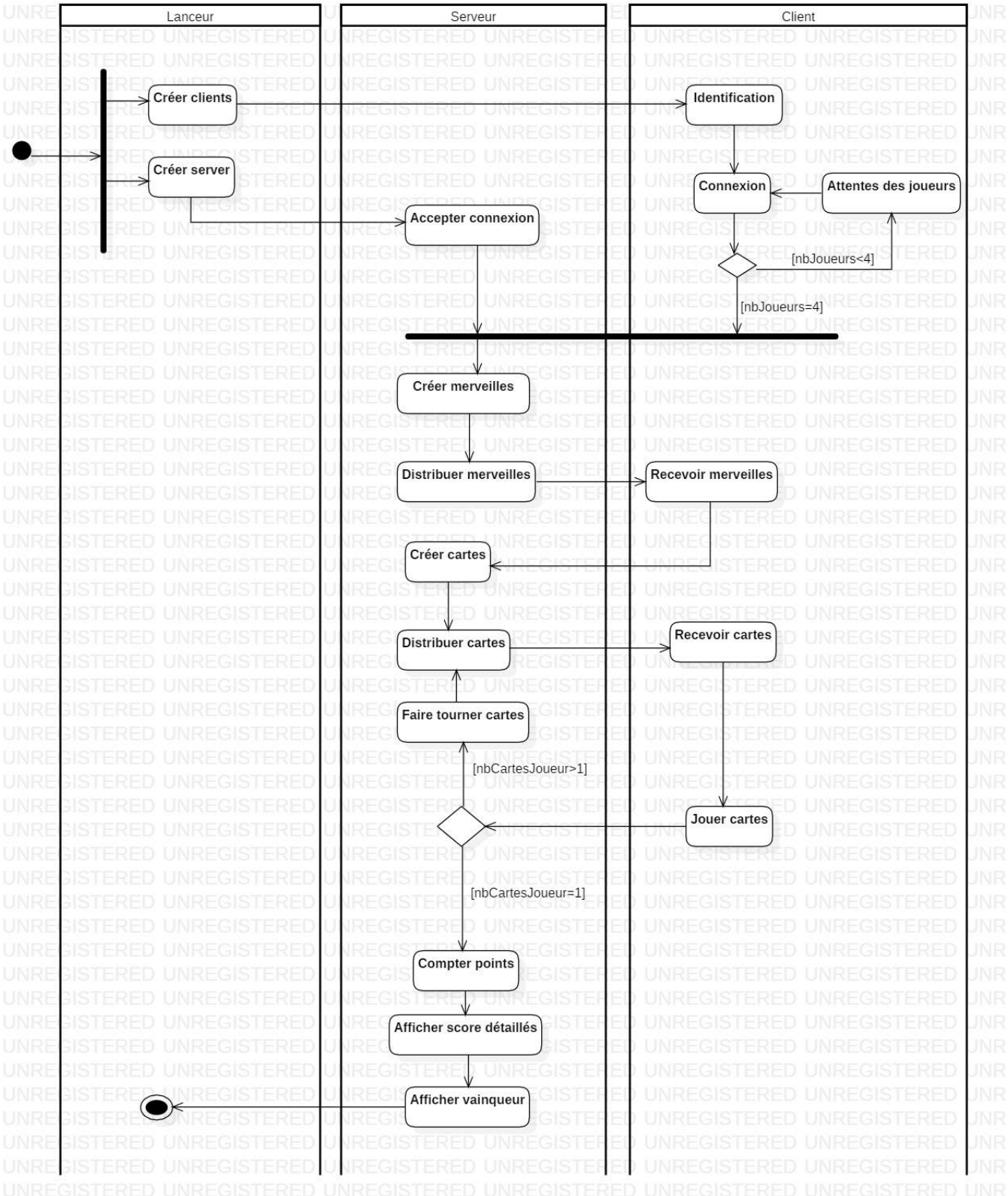
Refactoring : c'est une technique informatique qui nous permet de reprendre la structure de nos programmes et ainsi de modifier le code sans que le comportement de notre programme ne soit changé.

Modules : les modules permettent d'ajouter de nouvelles fonctionnalités au programme de départ et ainsi l'enrichir au fur et à mesure du code établi.

Dépendances : les dépendances permettent quant à elle de créer certaines relations entre différents packages par exemple.

Thread : c'est un fil d'exécution ou bien, plus simplement une tâche. Plusieurs threads peuvent s'exécuter en parallèle.

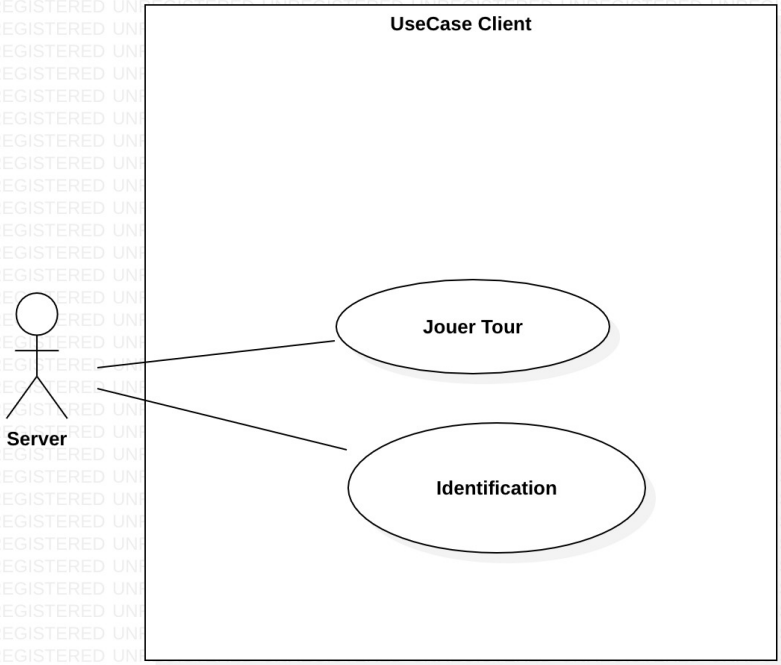
b. Représentation générale (diagramme d'activité)



Sur le diagramme d'activité nous avons rajouté l'action d'identification après la création des clients et avant la phase de connexion. De plus nous avons modifié la condition 'if' pour voir si le nombre de cartes du joueur est égal à 1 et passer directement à la fin de partie et ainsi au comptage des points. Le 'if' après la phase de connexion permet d'attendre les N clients, dans notre cas il s'agit des 4 joueurs afin de lancer la partie.

2. Client

a. Analyse des besoins (cas d'utilisation)

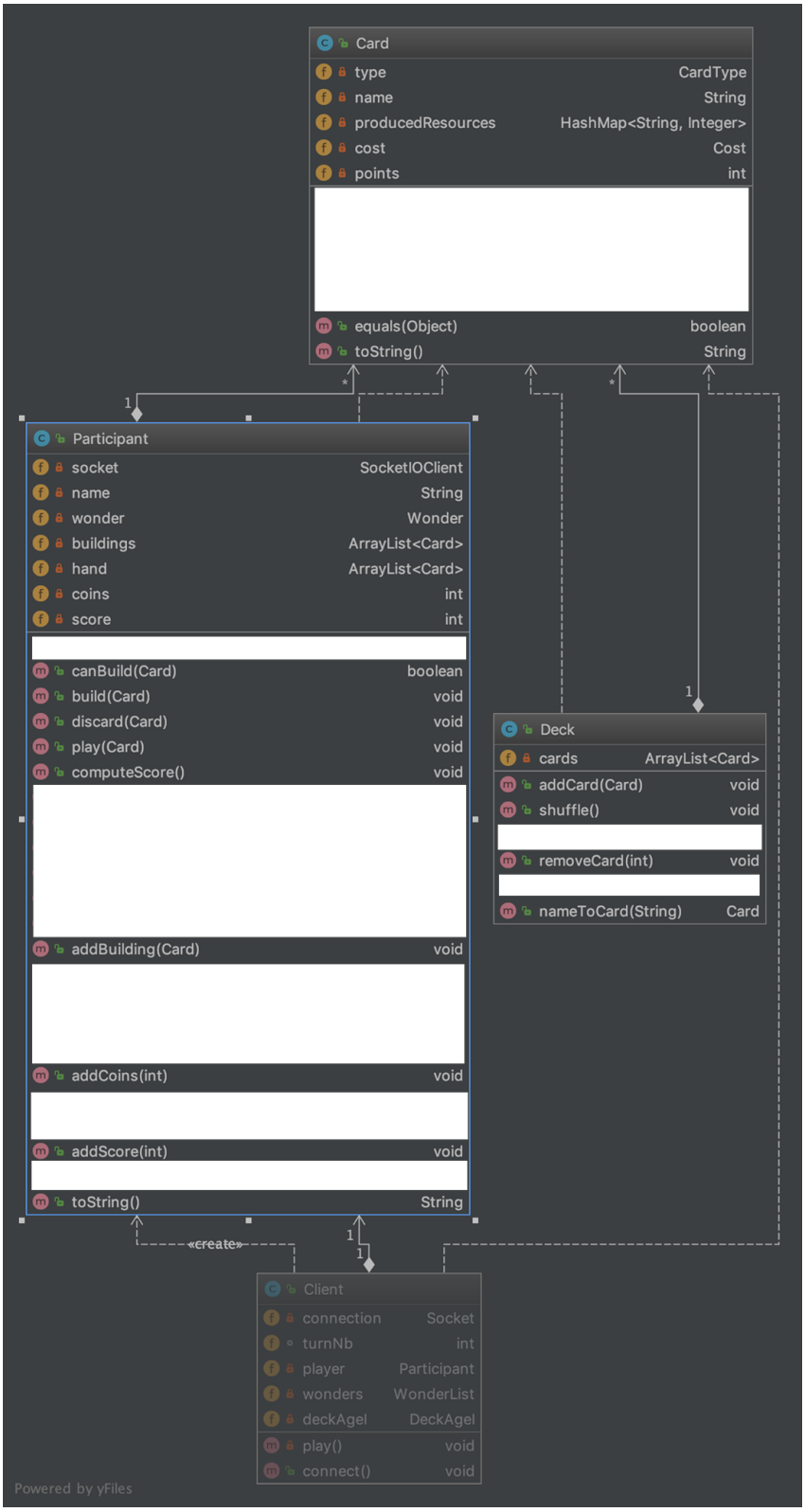


Sur le use case client, nous pouvons voir l'acteur serveur qui peut réaliser plusieurs actions :

- Jouer tour : qui englobe jouer une carte et défausser une carte
- Identification

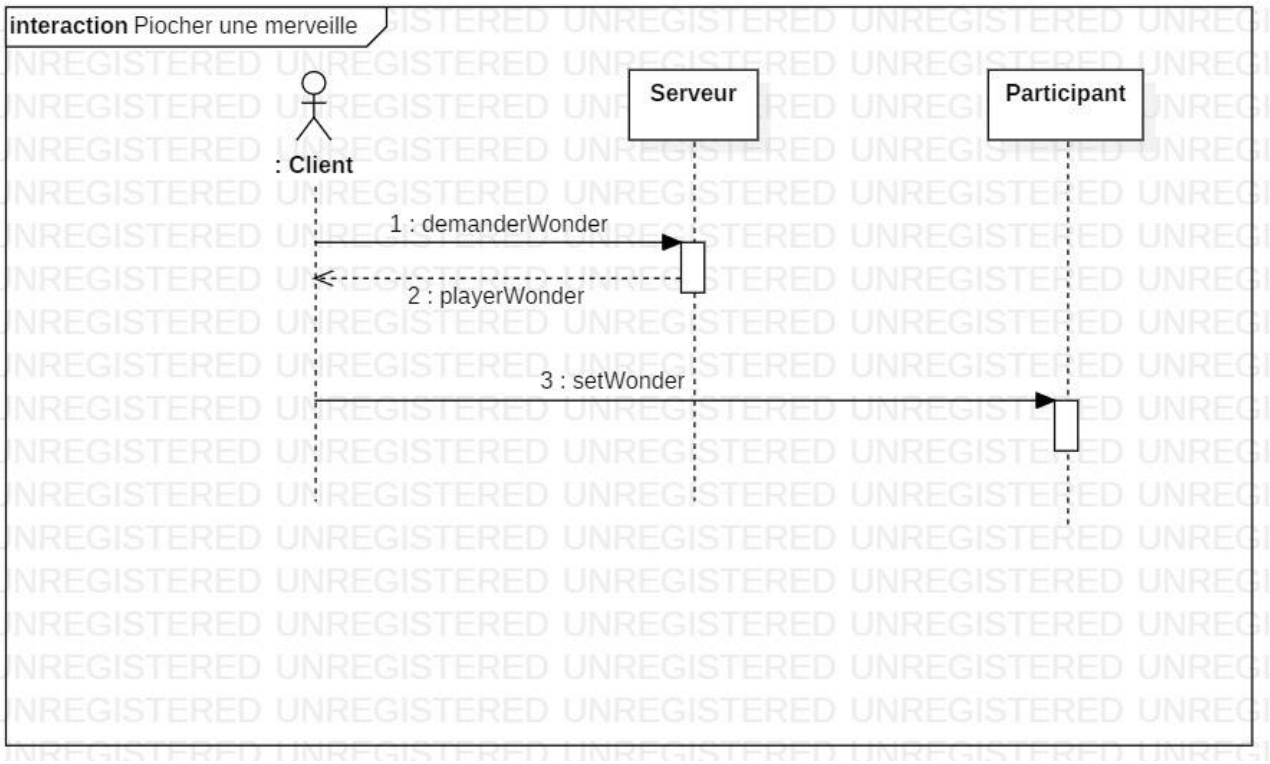
b. Conception logicielle

i. Point de vue statique (diagramme de classes)



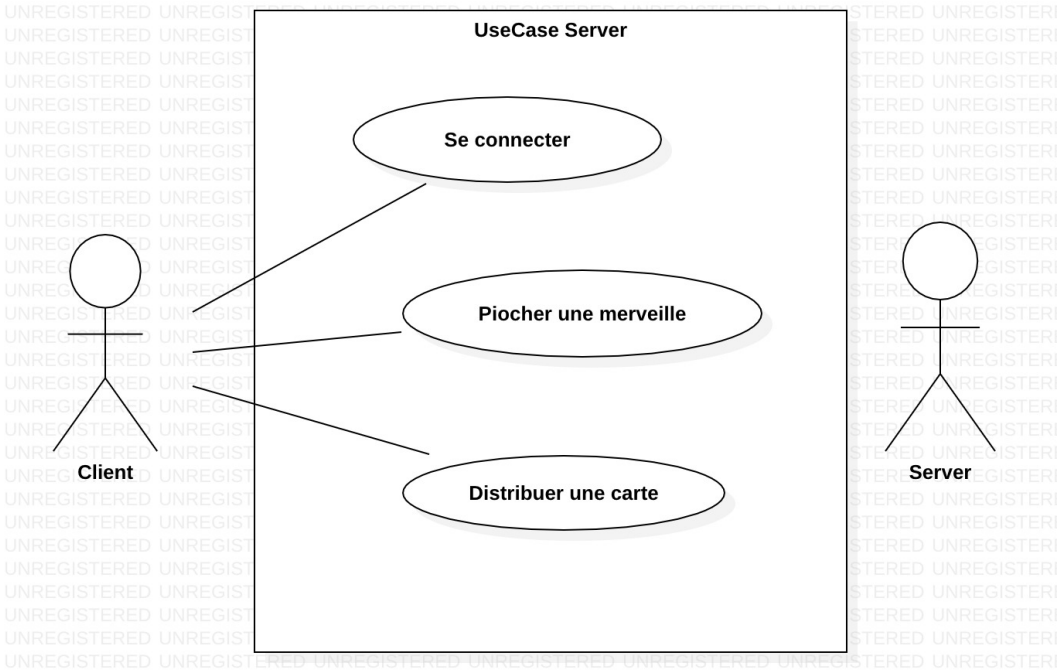
Voici le diagramme de classes client. On y retrouve ainsi les classes Card, Deck, Participant et Client qui y sont raccordées.

ii. Point de vue dynamique (diagramme de séquence)



3. Serveur

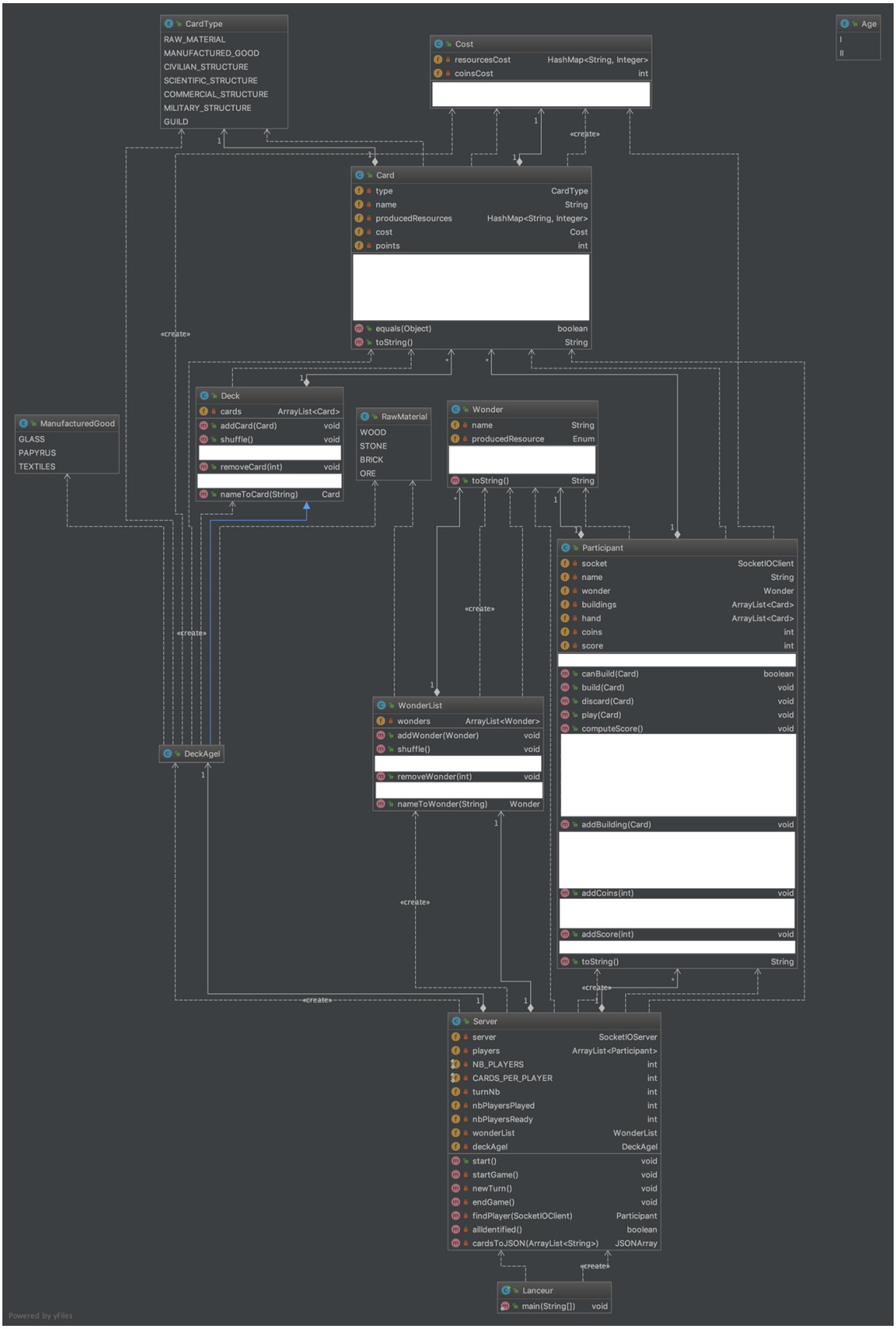
a. Analyse des besoins (cas d'utilisation)



Sur le use case serveur, nous pouvons voir l'acteur client qui peut réaliser de se connecter. Ensuite pour le jeu, l'acteur client peut ainsi piocher une merveille ou encore distribuer une carte.

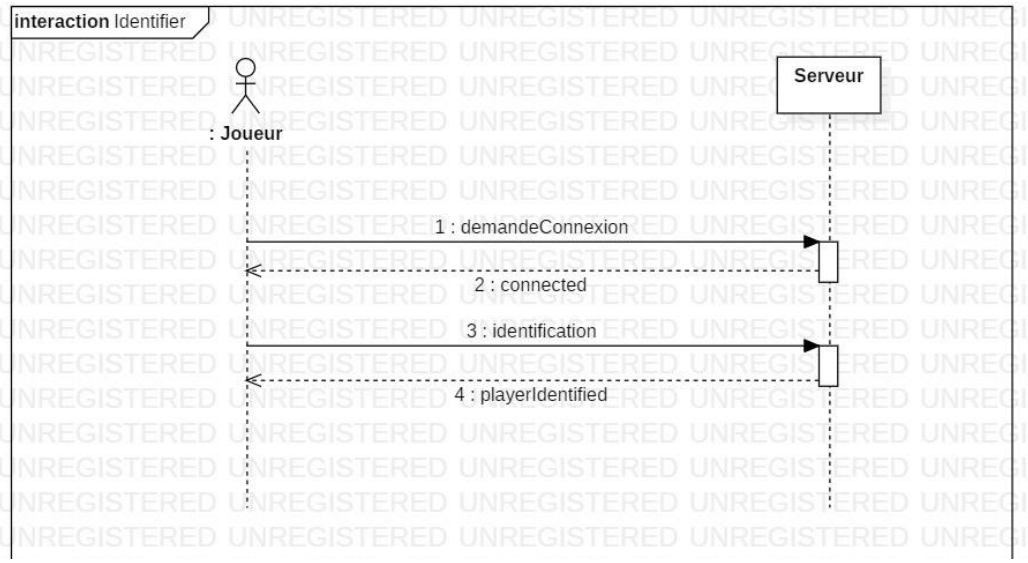
b. Conception logicielle

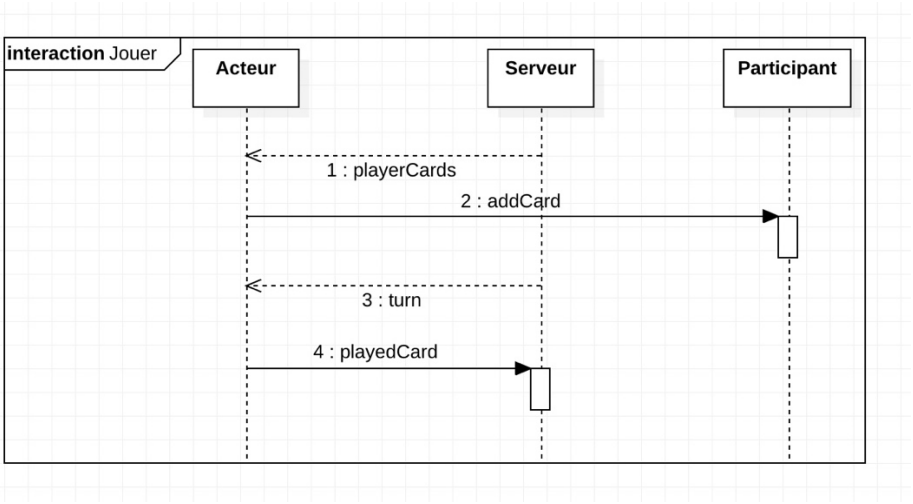
i. Point de vue statique (diagramme de classes)



Voici le diagramme de classes serveur.

ii. Point de vue dynamique (diagramme de séquence)





4. Conclusion

a. Analyse de notre solution (points forts et points faible

Points Forts	Points Faibles
Nous persévérons en prenant en compte les remarques qui nous sont faites afin d'améliorer au maximum notre projet. Notre jeu fonctionne. Le jeu est testé.	Nous avons rencontré des difficultés qui restent d'itérations en itérations et qui nous ralentissent dans notre développement. Toutes les fonctionnalités du jeu ne sont pas implémentées.

b. Évolution prévue

Notre code fonctionne à présent pour le premier âge. Dans le futur, il faudrait ajouter les 2 nouveaux âges ainsi que les fonctionnalités manquantes. De plus, on pourrait développer une interface pour joueurs afin d'avoir une meilleur vision du jeu.