

Ce document représente les diagrammes de classes intiales et finales, les changements que nous avons adapté dans la dernière phrase et une description sur l’état actuel de l’application

Le Vu Long TRAN

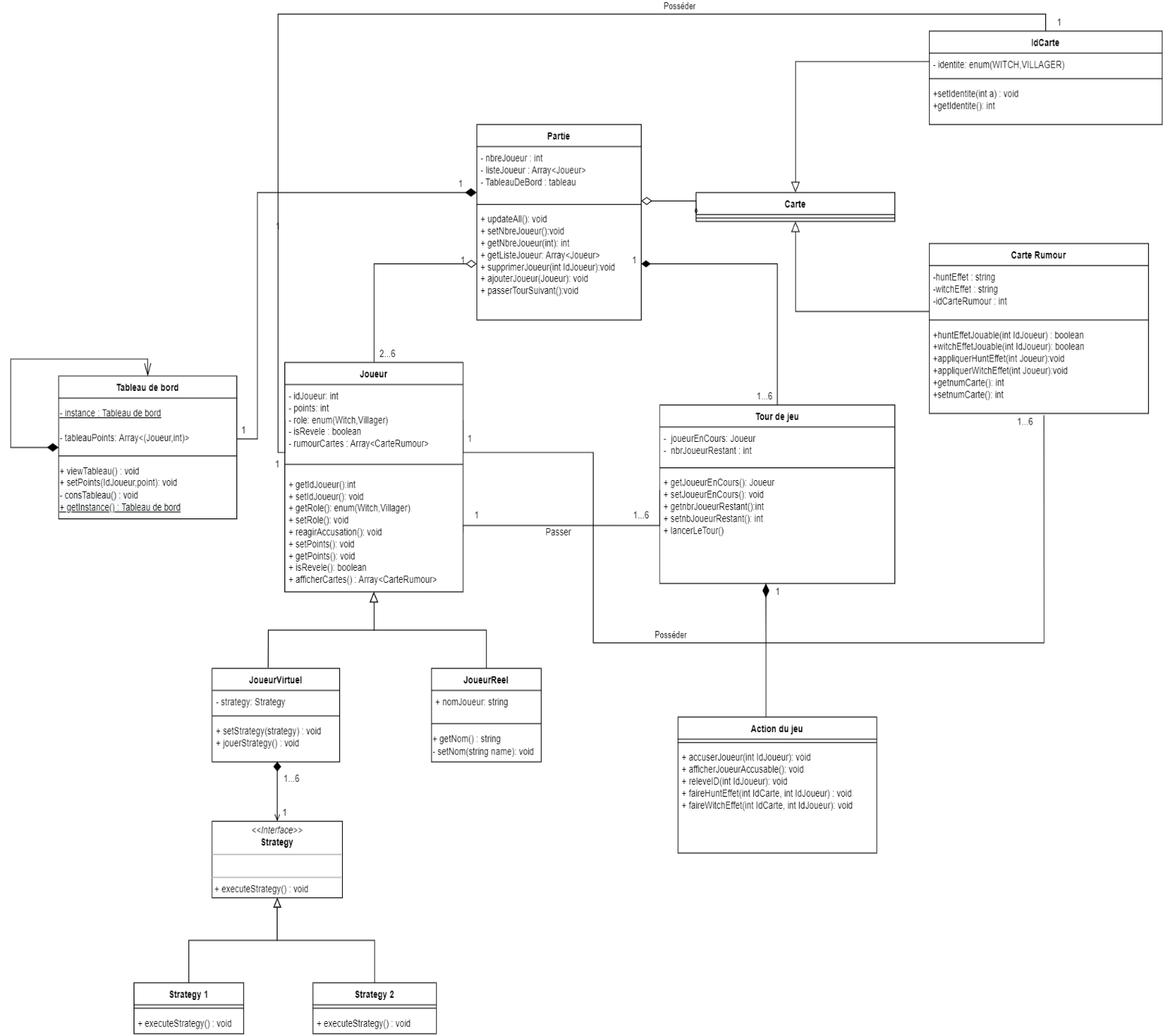
Mehmet SOYLEMEZ

LO02 - Livrable3

# Modélisation UML initiale

Nous allons tout d’abord représenter le diagramme de classe initiale que nous avons fait au début du projet, et ensuite nous décrivons les développements que nous avons apporté de là.

Voici le diagramme de classe initiale :



**Figure 1 : Diagramme de classe initiale**

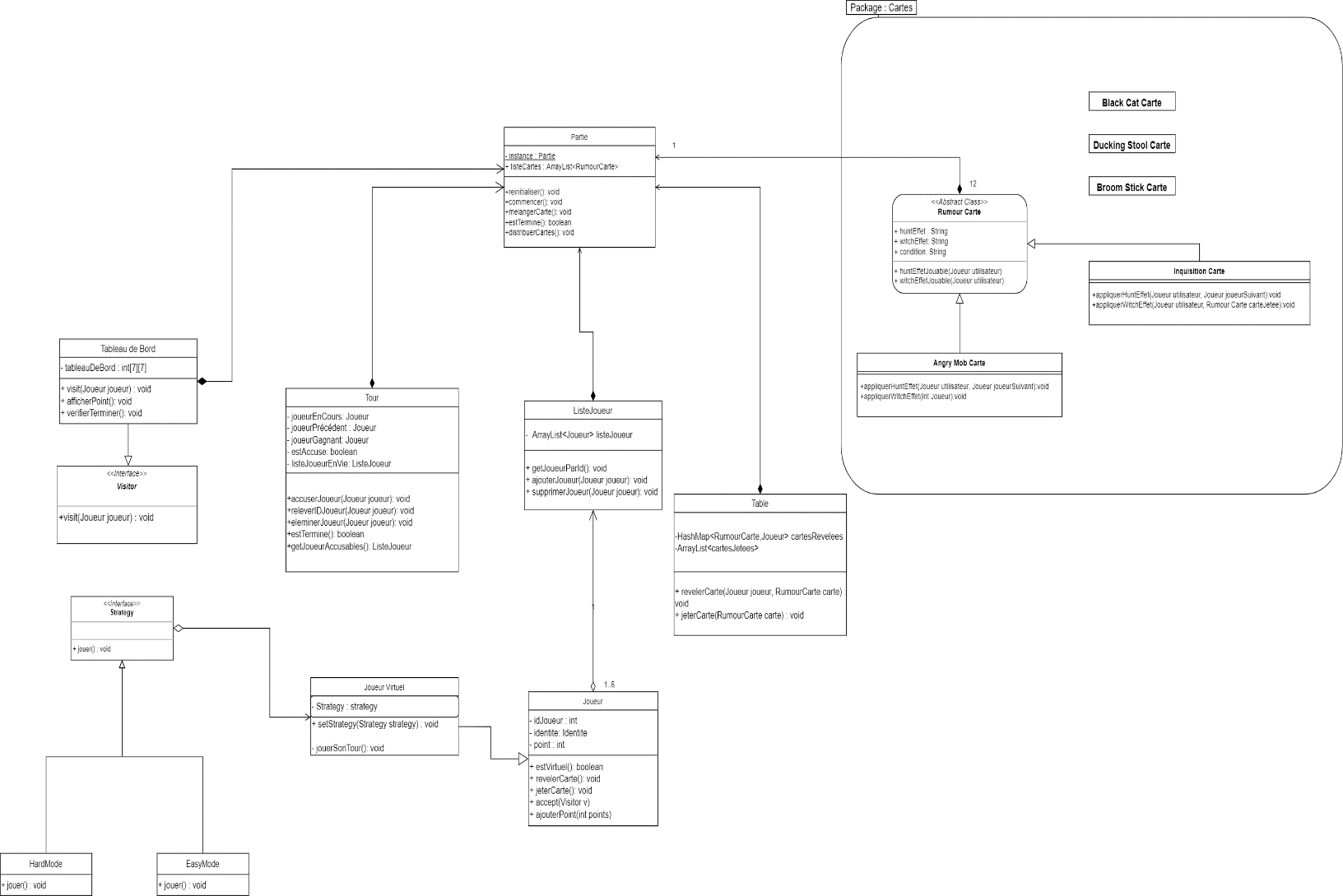
# Modélisation UML finale

## Changement dans le package model – le cœur d’application

Nous avons décidé d’essayer au maximum de modéliser toutes les classes qui représentent des entités du jeu, d’après nous cela rendre les classes plus lisibles, et permet de séparer les méthodes de manière plus sémantique, c’est – à – dire une classe ne contient que les méthodes qui lui concerne.

Voici les changements principals que on a adapté depuis le diagramme de classe initiale

* Dans le modèle initial, pour répresenter la liste de joueur, on a utilisé toujour un **ArrayList<Joueur>**, dans le modèle final, on a décidé de créer notre propre classe **ListeJoueur** qui possède un attribut de type **ArrayList<Joueur>**, et on a pu créé notre propres méthodes pour manipuler les joueurs,
* On a créé une classe **Main** pour représenter la main d’un joueur au lieu d’utiliser **ArrayList<RumourCarte>** comme dans le modèle initial.
* Création de la classe **Table** pour représenter **les cartes révélées** des joueurs et **les cartes jetées.** (Dans le modèle initial on a utilisé les collections pour les stocker, ce qui rend le code illisible !)
* Adoptation de l’interface **Visitor** pour le comptage de point
* La mise en packages des classes.



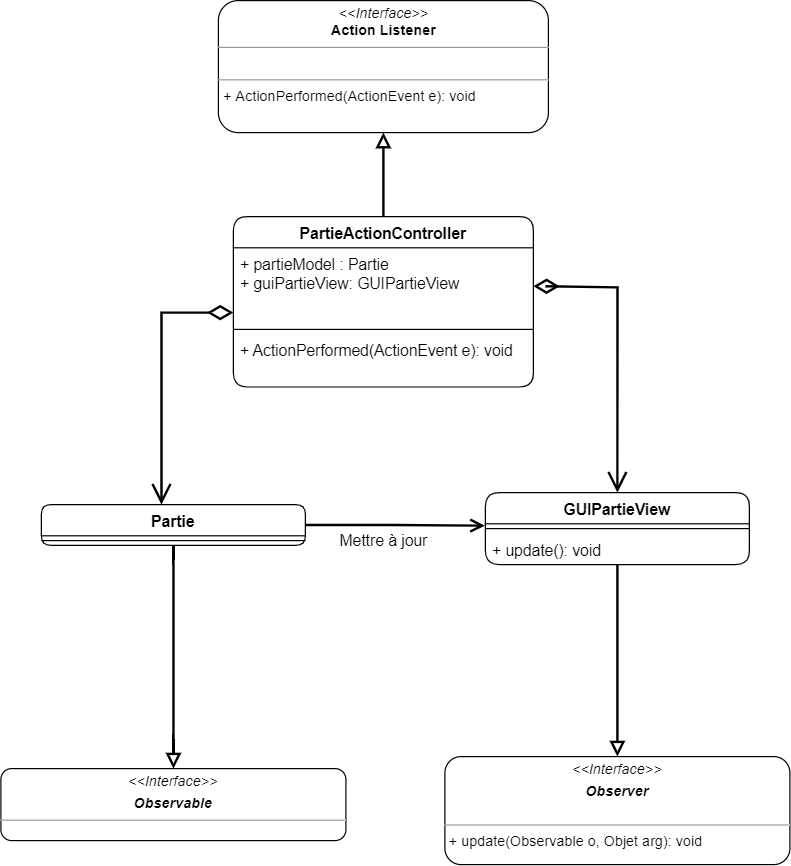
**Figure 2 : Diagramme de classe finale – package modèle**

## Intégration de le patron de conception MVC

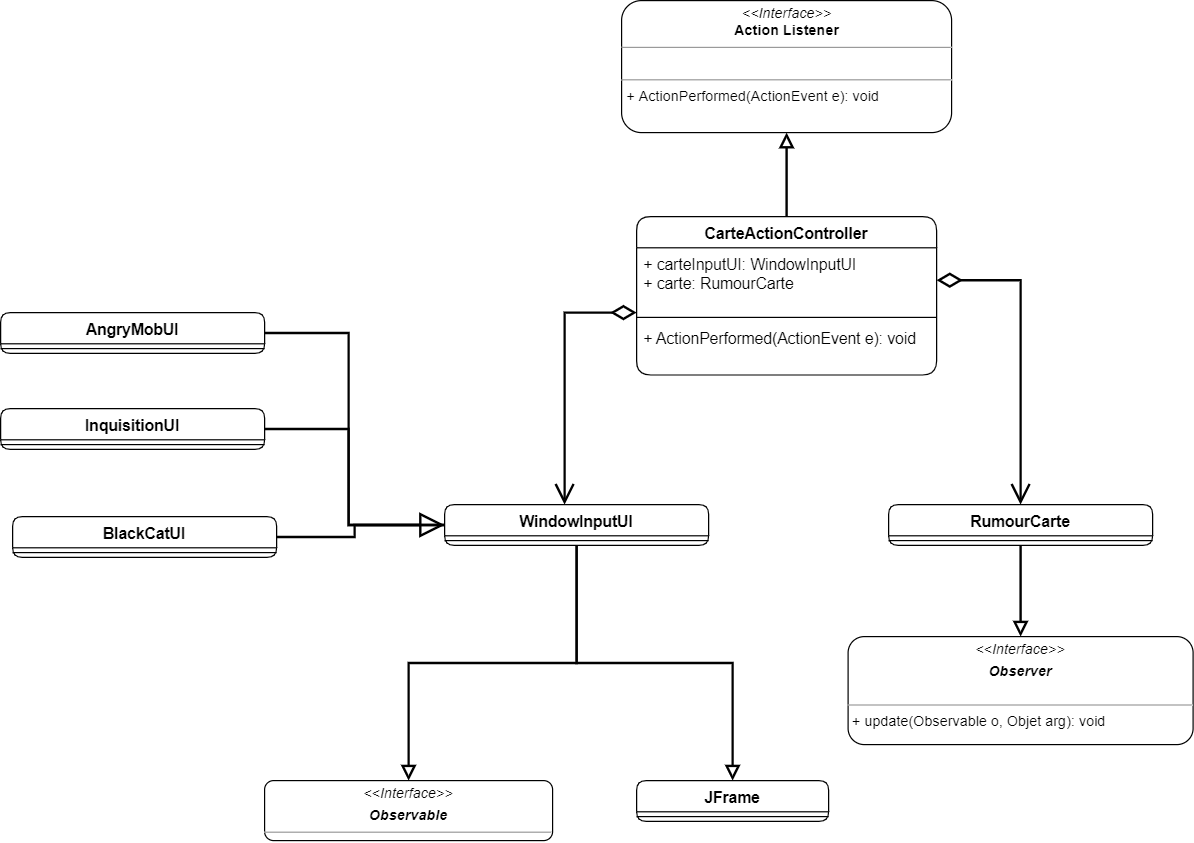
Pour intégrer le patron de conception MVC dans notre application, nous avons réfléchi beaucoup sur comment on va organiser les classes dans notre code, dans un premier temps nous n’avons aucune idées qu’est ce qui va aller dans le controller, le modèle ou la vue, comme c’est une application très complexe qui peut avoir plusieurs vue en même temps(La vue sur le joueur, sur la partie…) , et beaucoup d’actions à gérer pour le controller. Nous croyons c’est la partie qui nous demande le plus de temps pour réfléchir, finalement, nous avons décidé de créer plusieurs vues et plusieurs controller, chaque vue et controller est responsable pour la gestion de différentes tâches.

En gros, on a décidé de créer deux controllers pour différentes tâches :

* La première controller : **PartieActionController** est responsable pour la gestion d’action de joueur pendant le Jeu, les actions sont par exemple (Accuser, ReleverID, Faire Hunt, Faire Witch, Voir Points …)
* La deuxième controller : **CarteActionController** est créé particulièrement pour gérer les traitements de la carte jouée par le joueur, elle a l’objectif de collecter les inputs de la carte jouée par le joueur, et ensuite effectuer le traitement correspondant,
  + Par exemple : si un joueur jouer la carte Angry Mob avec l’effet Hunt qui consiste à révéler un autre joueur, la **CarteActionController** va générer une nouvelle fenêtre pour demander l’utilisateur de choisi un autre joueur, ensuite il recoit ce joueur via la fenêtre (AngryMobUI) et effectuer le traitement en appelant le modèle (AngryMob Carte).



**Figure 3 : Diagramme de classe MVC – PartieActionController**



**Figure 4 : Diagramme de classe MVC - CarteActionController**

# L’état actuel de l’application.

Dans cette partie, nous présentons l’état actuel de l’application, ceux qui ont été implantés, ceux qui ne le sont pas, et quelques améliorations possibles.

Ceux qui ont été implantés :

* La plupart des classes représentant les entités du jeu (**partie – joueur – joueurVirtuel …**) ont été implantés avec un fonctionnement presque correct, tous ceux qui sont présents dans le jeu, sont présents aussi dans nos classes, il faut encore plus de temps pour assurer aucune erreur peut survenir pendant le déroulement du jeu.
* La mise en place de modèle MVC avec les tâches et les rôles qui sont assez bien distribués entre modèle-controleur-vue.
* L’intégration de l’interface graphique, et la possibilité de joueur à la fois en mode console et GUI

Ceux qui ne sont pas encore implantés :

* Pour l’instant le tableau de bord ne permet que d’afficher les points sur la console, il faut dans l’avenir créer une graphique pour qu’on puisse voir les points en mode GUI.
* Nous avons constaté que nos strategié sont encore assez simple, on peut l’encore améliorer pour rendre les BOTS plus intelligents !
* On peut encore améliorer l’interface graphique, qui permet peut – être de sauvergarder l’état du jeu, d’ajouter les sons pendant qu’on joue