Projet JEST - LO02 - A25

Arthur Dodin, Pauline Dubois

Janvier 2026

1 Évolution du diagramme de classe

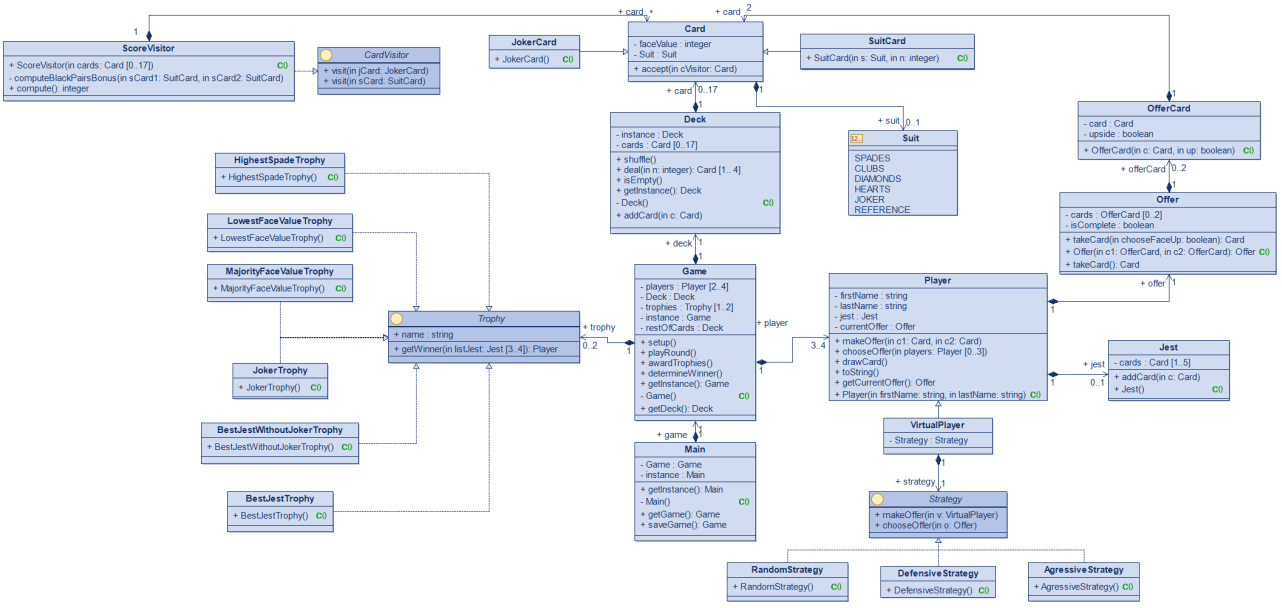

Figure 1: Diagramme UML initial

Le diagramme initial ne comprenait que les classes directement liées à la partie de Jest, sans gestion d'interface graphique notamment. Son architecture a été en grande partie conservée pour la partie *Model* du projet.

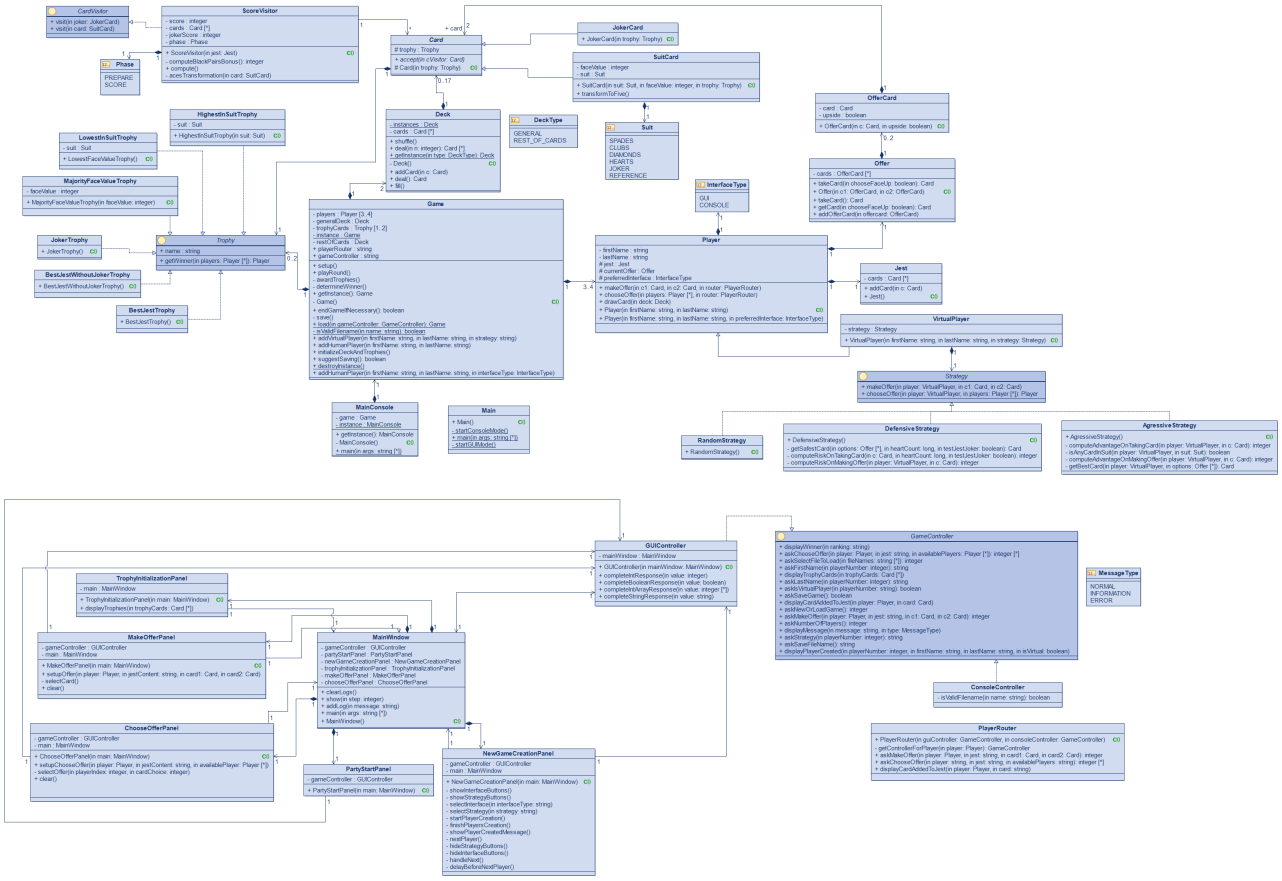

Figure 2: Diagramme UML définitif

On a choisi de scinder notre digramme de classe en trois parties:

- La partie *Model* en haut
- La partie *Vue* en bas à gauche
- La partie *Controller* en bas à droite

De nombreuses classes ont été ajoutées entre les versions initiales et finales du diagramme UML.

Ces différences s'expliquent tout d'abord par l'utilisation de patrons de conception (Multi-ton, Strategy ou Visitor) qui n'avaient pas été pris en compte, ou de façon erronée. Avec l'implémentation des différentes méthodes, nous nous sommes rendus compte que le schéma initial ne correspondait pas aux solutions les plus simples, propres ou efficaces pour certaines étapes de la partie de JEST. Nous avons donc fait le choix d'adapter notre architecture de classes pour composer au mieux avec les contraintes.

L'implémentation entre les livrables 2 et 3 de l'interface graphique a par ailleurs été l'occasion d'importants ajouts de près d'une dizaine de classes, que ce soit sur les parties Controller ou Vue. Les interactions consoles, auparavant directement dans le code modèle, ont été déplacées dans une vue Console spécifique, afin de permettre un fonctionnement en harmonie avec l'interface graphique.

2 Etat actuel de l'application

L'application permet actuellement d'effectuer les tâches suivantes du cahier des charges:

- Jouer à plusieurs, qu'il s'agisse de joueurs physiques et/ou virtuels
  - Lors de la création d'un joueur virtuel, on dispose d'un choix entre 3 stratégies: Aléatoire, Risquée ou Défensive. Le tout via l'utilisation du Patron Strategy, qui permet au joueur virtuel de déterminer l'action à mener à chaque instant
- L'ensemble de l'application est intégrée dans une interface graphique, utilisant les images fournies dans le sujet.

- L'ensemble de l'application est intégrée dans une interface rudimentaire en ligne de commande.
- L'architecture du code a été pensée avec la volonté de respecter au mieux les règles de la *Conception Orientée Objet*. Cela donne donc au projet modularité et extensibilité (notamment au niveau des cartes, des stratégies de joueur virtuel, des trophées).
- Il est possible de sauvegarder autant de parties que souhaité pour la reprendre plus tard, ou bien même jouer plusieurs fois une même partie pour tester différentes combinaisons.

L'ensemble des fonctionnalités implémentées sont fonctionnelles (à l'exception évidement de potentielles erreurs non détectées).

2.1 Interface graphique

L'ensemble de l'application est intégrée dans une interface graphique permettant, pour chaque joueur, de choisir s'il souhaite effectuer ses actions en console ou graphiquement. Il n'est actuellement pas possible de modifier cette préférence en cours de partie car nous n'avons pas trouvé le moment idéal pour demander aux joueurs s'ils souhaitaient changer. Néanmoins, le projet a été conçu pour, et les méthodes permettant d'effectuer ce changement ont été implémentées.

Cette interface graphique utilise les images de cartes fournies, et permet d'effectuer si souhaité l'ensemble du déroulement d'une partie, de la création des joueurs aux résultat, en passant par la sauvegarde et le chargement de sauvegarde.

2.2 Restes à implémenter

Les fonctionnalités suivantes n'ont pas été implémentées, faute de temps:

- Notion d'extension qui correspondent à des nouvelles cartes, qu'il vous faudra concevoir et implémenter. En début de partie, il sera demandé au joueur s'il souhaite intégrer ces cartes d'extension ou pas à la partie qui va débuter.
- Notion de variantes dans les règles du jeu. En début de partie, le choix d'une variante sera proposé au joueur et c'est la variante choisie qui imposera les règles d'une partie. Il est demandé de concevoir et implémenter deux variantes en plus des règles de base.

L'ajout de nouvelles cartes serait facile à mettre en place, le code étant été prévu pour faciliter des ajouts dans le deck de départ. Il ne s'agit ici aucunement d'une difficulté technique.

L'ajout de variantes de jeux pourrait s'avérer plus complexe, en fonction des moments de la partie impactés. S'il s'agit du comptage des points par exemple, le patron *Visitor* utilisé permet une implémentation simple et rapide. D'un autre coté, s'il s'agit de modifier le déroulé de la partie, l'implémentation de variantes s'avérerait plus fastidieux.