Etat fonctionnel de l’application

Nous allons nous intéresser à l’état fonctionnel en regard du présent cahier des charges.

L’application permet de jouer à deux ou trois joueurs. Ces joueurs peuvent être des joueurs physiques ou des joueurs virtuels. Pour ce qui est des joueurs virtuels, le jeu leur attribue une stratégie de jeu simple. L’IA place sa carte en fonction de ce qui va lui rapporter le plus de point à l’instant t : l’alignement des formes, des couleurs ou du remplissage. Cette stratégie lui permet de déterminer les actions à mener à chaque instant. De plus, nous avons respectée le nombre de round demandé pour finir une partie de Shape Up, ils sont au nombre de 4.

Le jeu propose deux variantes comme initialement demandé dans le cahier des charges en plus d’un mode de jeu classique. Le joueur choisit en début de partie le choix de la variante :

1. La première variante est un mode de jeu classique suivant les règles du jeu : l’application permet de jouer à deux ou trois joueurs, la Victory Card est donnée au joueur en début de partie et il pioche et pose directement sa carte sur le plateau.
2. La deuxième variante consiste à proposer au joueur en début de partie de choisir sa Victory Card.
3. La troisième variante est proposée par les règles du jeu et consiste à donner une main de 3 cartes à chaque joueur en début de partie. A son tour, il doit jouer une carte puis en piocher une nouvelle. Lorsqu’il n’y a plus de cartes à la pioche, il dépose deux de ses cartes sur le plateau et conserve la troisième qui est sa Victory Card.

Pour ce qui et du choix de la forme du tapis, nous sommes restés sur une forme de 5 cartes par 3. Cependant le joueur a la possibilité de déplacer l’ensemble des cartes du plateau en plaçant sa carte en dehors de celui-ci. Cela n’est pas choisi en début de partie, il peut choisir durant la partie de modifier le placement des cartes sur le plateau.

L’ensemble de l’application est intégré dans une interface graphique. Il est également possible de jouer dans une seconde interface, concurrente à la première et qui fonctionne en ligne de commandes.

Enfin, l’architecture retenue respecte les règles de la conception orientée objet : nous avons rassemblé dans une même entité appelée objet les données et les traitements qui s’y appliquent et respectée la syntaxe de la déclaration d’une classe.

Différence entre le diagramme de classes initial et le diagramme de classes final

Lors de la production du cœur de l’application et de l’interface en ligne de commandes, nous avons effectué quelques changements. Ainsi, on observe des modifications quant à la modélisation UML finale.

Nous avons apporté des modifications au patron de conception Strategy en ajoutant deux classes implémentant Strategy : RandomStrategy et ObstructStrategy permettant aux IAs davantage de choix pour le placement de leurs cartes.

La classe Position a été supprimée et ses méthodes ajoutées à la classe Plateau. En effet, nous nous sommes rendus compte que ses méthodes avaient plus de sens à être utilisé dans la classe Plateau notamment la méthode isPosAlreadyTaken( ) qui vérifie si il y a déjà une carte posée sur le plateau.

Ensuite, nous avons ajouté une énumération Mode qui comporte 3 instances : *Classique, Avancé, Personnalisé.* Elle permet au joueur de choisir son mode de jeu.

L’ensemble des autres classes ont été gardés. Cependant, nombreuses sont les méthodes ajoutées dans les différentes classes pour le bon déroulement du jeu :

1. start( ), checkAndGame( ), tourDeJeu( ), comptagePoints( ) pour la classe Jeu permettant de lancer une partie de jeu, de s’assurer que chaque joueur prenne la main à tour de rôle et d’annoncer un gagnant lorsqu’il ne reste plus aucune carte à la pioche.
2. chooseCardToPlay( ) pour la classe Joueur. Nous nous sommes rendu compte que nous en aurions une grande nécessité pour une variante du jeu.
3. moovePositions( ) pour la classe Plateau permettant de déplacer les cartes sur le plateau.

Nous n’avons pas pu prévoir l’ensemble des situations du jeu au début du projet. Les modifications ont été apportées au fur et à mesure de l’implémentation.

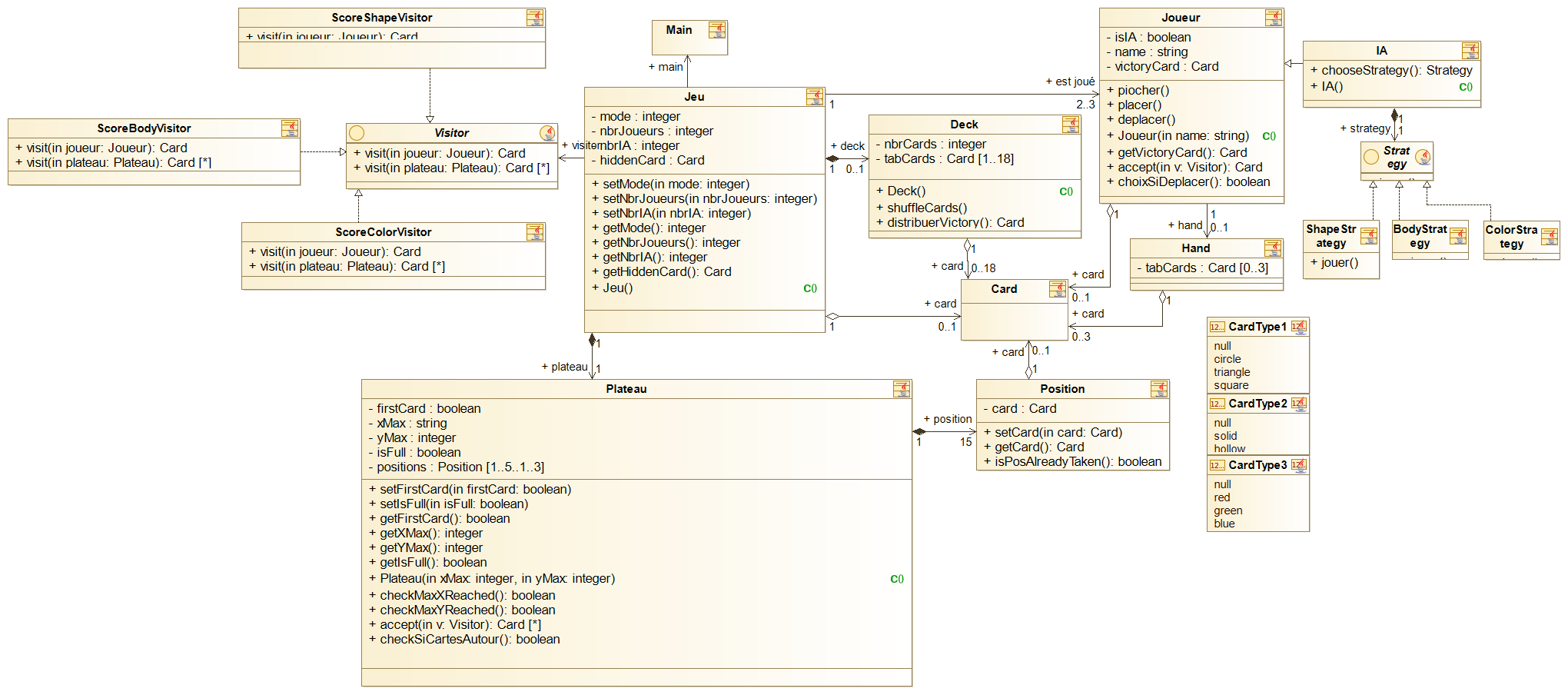


Figure 1. Diagramme de classes initial

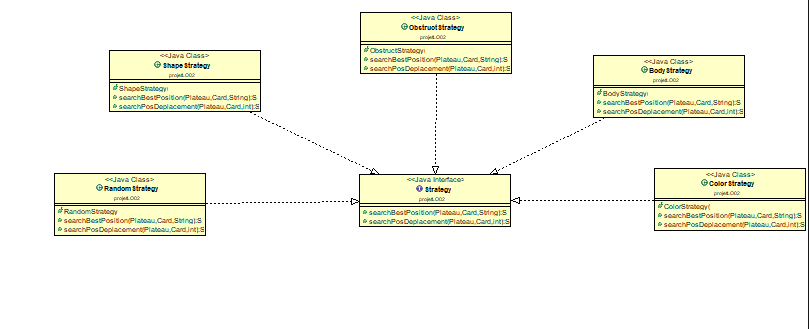


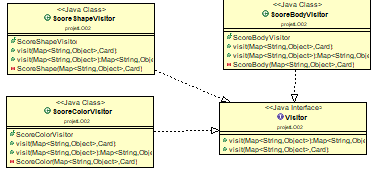
Figure . Diagramme de classes final - patron de conception Strategy

Figure . Diagramme de classes final - patron de conception Visitor

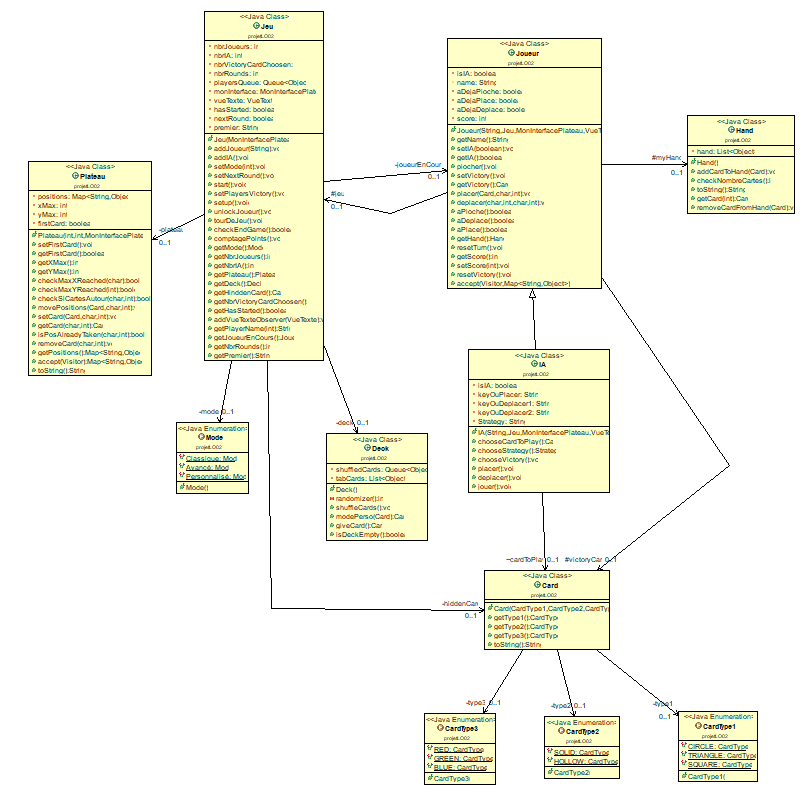


Figure . Diagramme de classes final - moteur du jeu