Diagrammes de classes et évolution

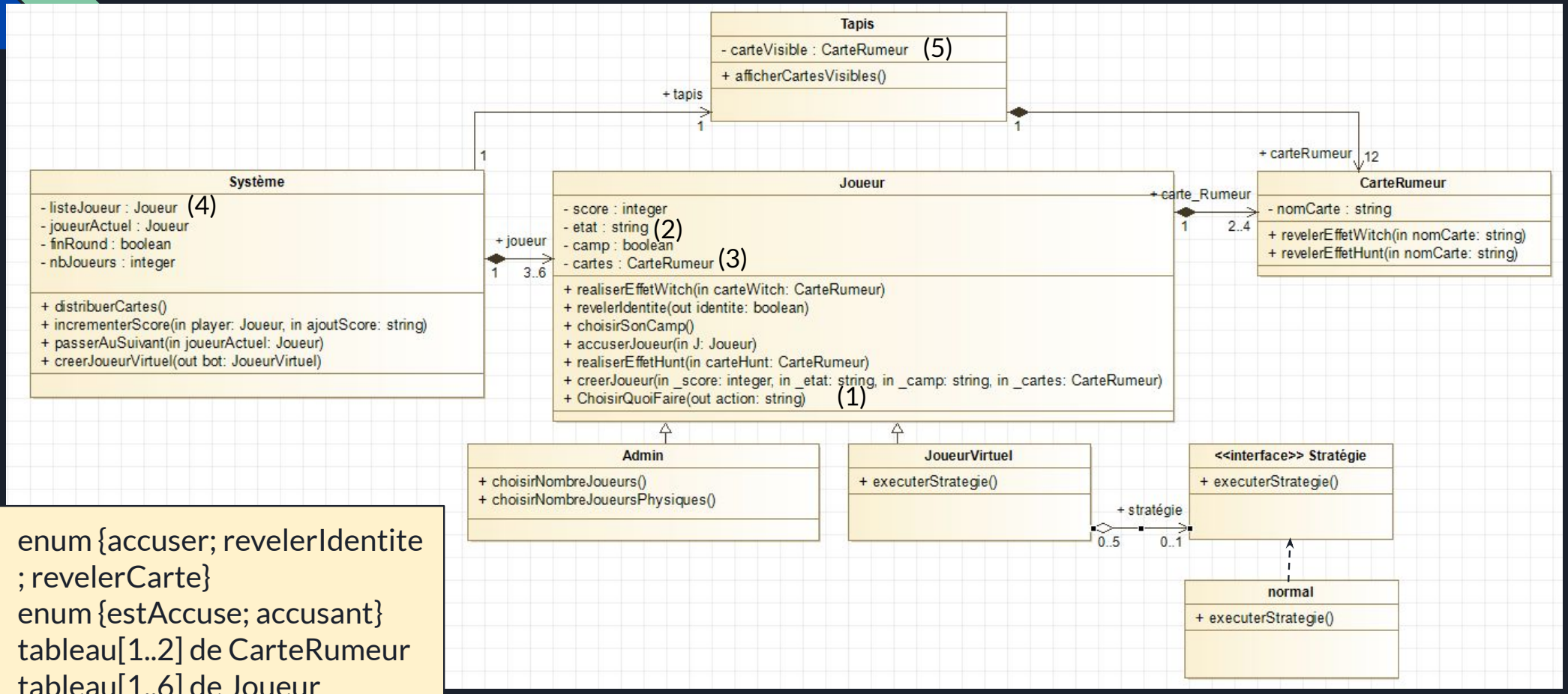
Notre diagramme de classes a beaucoup évolué au cours du développement de l'application. Nous nous sommes vites rendus compte que nous n'avions pas bien compris le patron de conception *Strategy*, que nous avons changé et précisé en deux stratégies distinctes : AccuserEtReveler, et ToujoursReveler. Finalement, la classe CarteRumeur s'est transformée en classe abstraite. Nous avons très vite compris que chaque carte avait sa propre définition de l'effet Hunt! et l'effet Witch?. C'est donc pour cela que nous avons décidé de créer une classe par carte. Ainsi nous avons pu redéfinir les méthodes realiserEffetWitch() et realiserEffetHunt() pour chaque carte. De plus, nous avons abandonné l'idée d'un joueur Admin, étant donné que la fonction de gestion de partie se fait par le système.

Enfin, l'interface graphique a été un challenge pour nous. Nous avons intégré le patron de conception Observer comme nous avons pu. Cela a également impliqué de définir des contrôleurs, et une vue graphique. La méthode main a dû changer de place pour prendre part à la vue.

D'un point de vue général, nous avons surtout ajouté des classes/méthodes/attributs. Notre version 1 du diagramme de classes était tout de même réaliste, bien qu'incomplet.

Etienne Lanternier et Nathan Vilmen

Diagramme de Classe Version 1



(1) enum {accuser; revelerIdentite ; revelerCarte}
(2) enum {estAccuse; accusant}
(3) tableau[1..2] de CarteRumeur
(4) tableau[1..6] de Joueur
(5) tableau[1..12] de CarteRumeur

Diagramme de classes : Version 2