Centrale Marseille

NOTICE MONOPOLY

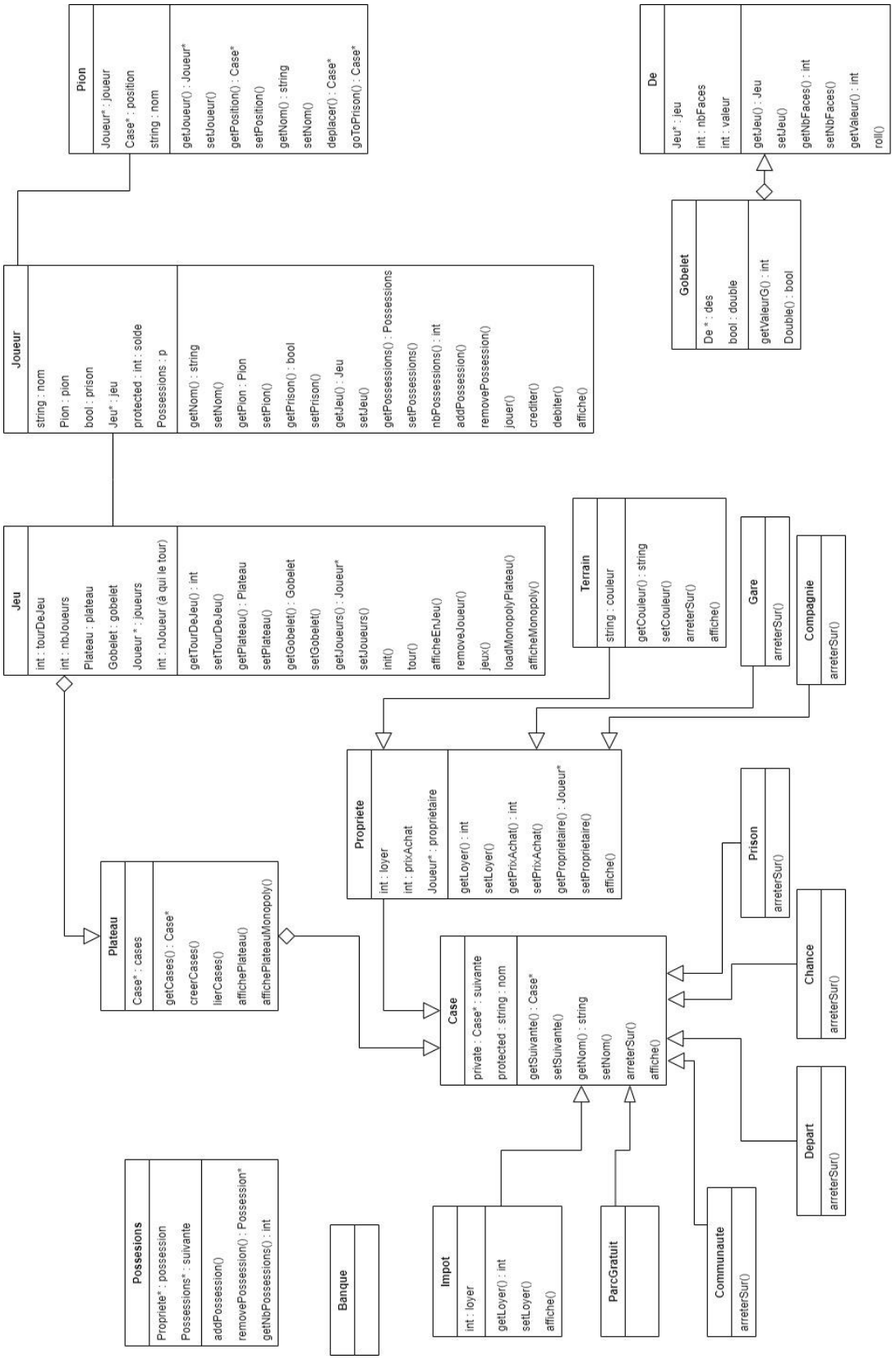
Programmation Orientée Objet - C++

Composition du groupe :
Nicolas BERT, Glenn LOUEDEC, Arthur LOURADOU, Céline MA

Notre projet consiste à l'implémentation du jeu de société **Monopoly**. Les règles officielles du jeu sont fournies depuis le site www.monopolypedia.fr.

Cette notice a pour objectif de vous présenter les différentes classes implémentées qui permettent de générer une partie virtuelle du jeu Monopoly. Un diagramme UML a été réalisé afin de visualiser l'ensemble des différentes classes, ainsi que les relations existantes entre elles.

Diagramme UML pour le Monopoly



Détail des différentes classes

Pour la gestion du déroulement de la partie

Jeu

La classe **Jeu** permet de gérer les joueurs sous forme de liste, les lancers de dés via la classe Gobelet, la gestion des tours du jeu et le plateau où se positionneront les pions.

C'est dans cette classe que nous avons géré l'ensemble du déroulement et de l'affichage de l'état de jeu dans la console. La méthode **init** permet d'interagir avec l'utilisateur via la console afin de saisir le nombre de joueurs, leur nom ainsi que leur pion.

Gobelet / Dé

La classe **Dé** comporte plusieurs attributs, en particulier le nombre de faces qu'il comporte et la valeur qu'il affiche. Cette valeur est générée via la méthode **roll** qui affecte aléatoirement une valeur comprise entre 1 et le nombre de faces. La méthode fait appel à un **seed** afin de générer des valeurs non déterministes et donc aléatoires.

La classe **Gobelet** quant à elle hérite de la classe **Dé** et est par conséquent un dé. Ce gobelet possède un attribut **doublet** qui indique si la valeur du gobelet est égale à la valeur du Dé. La méthode **getValeurG()** du gobelet permet de récupérer ses valeurs afin de gérer les déplacements sur le plateau.

Joueur

La classe **Joueur** comporte plusieurs attributs qui sont le nom, le pion, le solde, les possessions (sous forme de liste chaînée), si le joueur est en prison ainsi que le jeu. Cette classe est associée à la classe **Jeu** via une relation d'association.

Pour la gestion du plateau de jeu

Plateau

La classe **Plateau** permet de modéliser le plateau de jeu du Monopoly. Le plateau dispose de plusieurs **Cases** qui sont de nature différente (case Départ, case Prison, les cases Propriété, les cases Chance et Communauté...).

Pour générer le plateau du jeu, on crée une **liste chaînée bouclée** à partir de la case Départ en liant chacune des cases suivantes suivant l'ordre indiqué par le jeu.

Case

La classe **Case** permet de générer les différentes cases du plateau de jeu. Autrement dit, les autres types de cases sont en fait des classes héritées de la classe **Case** qui est abstraite. Cette classe dispose de l'attribut **suivante** qui correspond à un pointeur vers une autre case. Ainsi pour créer le plateau de jeu, on lie les cases les unes à la suite des autres en faisant pointer l'attribut **suivante** vers une autre case**.**

À noter que l'action au niveau du jeu dépend du type de case sur laquelle un joueur s'arrête. De ce fait, la méthode **arreterSur()** est virtuelle et sera définie suivant la nature de la case.

Pion

Le **Pion** est la classe qui gère les déplacements d'une instance de joueur. Il se déplace de case en case en utilisant la liste chaînée qu'elles constituent. Bien entendu, il est associé à un **Joueur** unique avec une liaison One To One. En revanche, c'est la classe **Plateau** qui se charge d'afficher le pion avec le motif choisi par l'utilisateur à l'initialisation. Ce motif est stocké dans la variable **nom**.

Cette classe implémente aussi le passage par la case départ et le départ en prison, sans toucher l'argent correspondant à ce passage le cas échéant.

Possessions

La classe **Possessions** permet la gestion des propriétés d'un joueur en les stockant sous la forme d'une liste chaînée. Ces informations sont nécessaires pour le calcul du loyer à payer par un joueur qui s'arrête sur la propriété d'un autre.

Dans le cadre de notre projet, nous n'avons pas implémenté le calcul des loyers suivant les possessions des joueurs.

Propriété

Lorsqu'un joueur s'arrête sur une case **Propriété**, seules possibilités s'offrent à lui :

- soit il peut en devenir propriétaire dans le cas où aucun joueur ne l'a achetée
- soit il doit payer un loyer à un autre joueur devenu propriétaire avant

À noter que la possibilité de mettre une propriété aux enchères (dans le cas où un joueur refuse d'acheter ou dispose d'un solde insuffisant) n'a pas été définie dans notre projet.

La classe abstraite **Propriété** regroupe les cases de type **Terrain**, **Gare** et **Compagnie** qui peuvent toutes être achetées par un joueur.

Terrain/Gare/Compagnie

Selon les règles du Monopoly, le montant du loyer diffère suivant le type de **Propriété** et les possessions des propriétaires. Pour le détail des calculs des loyers, vous pouvez consulter le site www.monopolypedia.fr.

Pour les propriétés de type **Terrain**, elles sont regroupées par famille de couleur. Le montant du loyer est fixe jusqu'à ce que le propriétaire d'un terrain dispose de tous les autres terrains de la même famille. Dans ce cas, le loyer est doublé. La méthode **arreterSur()** fait appel à la classe **Possessions** dans le calcul du loyer.

À noter que la possibilité de construire des maisons puis hôtel permettant d'augmenter le loyer d'un **Terrain** n'a pas été implémenté dans le cadre de notre projet.

Pour les propriétés de type **Gare**, le loyer payé dépend également du nombre de gares dont dispose son propriétaire. La méthode **arreterSur()** fait également intervenir la classe **Possessions** dans le calcul du loyer.

Pour les propriétés de type **Compagnie,** le montant du loyer fait intervenir un lancer de dés. De ce fait, la méthode **arreterSur()** génère un lancer depuis la classe **Gobelet**.

Chance/Communauté

Lorsqu'un joueur s'arrête sur une case de type **Chance** ou **Communauté**, il est contraint de réaliser l'action indiqué. Soit il est déplacé dans une autre case, soit son solde est affecté par l'action réalisée (à la hausse comme à la baisse).

Dans le cadre de notre projet, les cases **Chance** et **Communauté** n'ont été pas implémentées. Les actions permises sont réduites et ne sont pas conformes aux règles du jeu. Lorsque l'on tombe sur ces cases on n'a seulement que quelques possibilités de cartes qui ne peuvent que vous faire gagner ou perdre de l'argent.

Parc Gratuit

Aucune action n'est réalisée lorsqu'un joueur s'arrête sur cette case. (Relax!)

Impôt

Un joueur qui s'arrête sur cette case est contraint de verser un montant à la banque en guise d'impôts. Le montant à verser est désigné par l'attribut **loyer**.

Départ

Il s'agit de la première case du plateau de jeu, elle se situe donc au début de la **liste chaînée bouclée** lors de la génération du jeu. D'après les règles du jeu, lorsqu'un joueur s'arrête sur la case ou la dépasse, il perçoit un montant de 200$. Pour la gestion du dépassement de la case **Départ** pendant le jeu, celle-ci est implémentée au niveau de la classe **Pion**, lors du déplacement du pion.

Prison

Le déplacement vers la case **Prison** est géré par la classe **Pion** avec la méthode **goToPrison()**. Pour sortir de prison, un joueur doit réaliser un doublet ou payer 50$, sinon il doit passer son tour. Au bout de trois tours sans doublet, le joueur paie un montant de 50$ pour sortir de prison et joue ensuite son tour en lançant les dés. La sortie de prison est gérée au sein de la classe Pion.