

Devoir de Génie Logiciel

Le jeu de Monopoly

Benoît Moulin / Julien Van Den Bossche
Groupe 3.2

Sommaire

1. Capture des besoins fonctionnels.....	page 3
2. Description dynamique globale.....	page 4
2.1. Le modèle environnemental.....	page 4
2.2. Les scénarios de transactions.....	page 4
3. Le modèle statique.....	page 12
3.1. Les cas d'utilisations.....	page 12
3.2. Le diagramme des classes.....	page 13
4. Le modèle dynamique.....	page 24
4.1. Le diagramme des séquences.....	page 24
4.2. Le diagramme de collaboration.....	page 30
4.3. Le diagramme des états et des objets.....	page 31

1. Capture des besoins fonctionnels

Le Monopoly est un jeu de société qui s'adresse à un public de 7 à 77 ans. On y joue entre 2 et 4 joueurs.

On utilise le clavier et la souris.

On se déplace sur un plateau de jeu contenant 22 terrains à bâtir, 4 gares, 2 compagnies, une case prison, une case allez en prison, une case taxe de luxe, une case impôt sur le revenu, une case parc gratuit, une case départ, des cases chance et caisse de communauté.

Le but du jeu est de prendre le contrôle d'une ville en achetant le maximum de biens (terrains, hôtels, gares...).

Chaque joueur possède la même somme d'argent au départ ensuite un premier joueur lance les dés et se déplace du nombre de case du résultat du jet.

Chaque joueur s'arrêtant sur la propriété d'un autre joueur devra payer un loyer. Plus le terrain est construit et plus le loyer sera cher.

Si une propriété n'a pas de propriétaire on peut l'acquérir au prix indiqué sur la case.

Si on souhaite construire sur un terrain on doit acheter des constructions à la banque et posséder tous les terrains de la couleur du terrain où l'on souhaite construire.

Si l'on tombe sur une case chance ou caisse de communauté on tire une carte et le joueur doit faire une action (se déplacer, verser, recevoir de l'argent, aller en prison...).

Le joueur peut vendre ses biens s'il a besoin d'argent, il peut aussi hypothéquer ses terrains.

Le joueur qui possède le plus de biens s'enrichira alors que les autres joueurs lui donneront de l'argent. Le joueur qui n'a plus d'argent et de bien sera mis en faillite. Le dernier joueur restant sera le vainqueur et aura le monopole de la ville.

Actions principales :

- Lance la partie
- Se déplace
- Achète
- Vend
- Construit
- Verse de l'argent
- Reçoit de l'argent
- Echange des biens
- Hypothèque

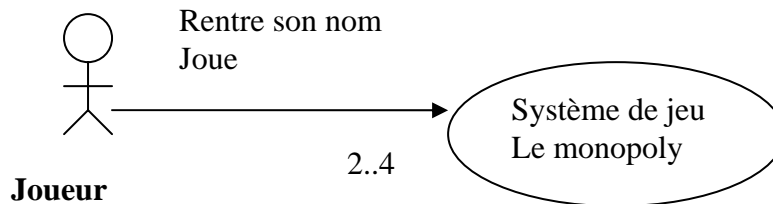
Autres actions possibles :

- Quitte la partie en cours
- Quitte le jeu
- Consulte l'aide

2. Description dynamique globale

2.1. Le modèle environnemental

Le joueur est un acteur primaire du système. Il rentre son nom, attend que les autres joueurs aient fait de même puis il joue.



2.2. Les scénarios de transactions

Titre du scénario 1 : Le joueur s'arrête sur une propriété.

Acteur primaire : Le joueur.

Acteurs secondaires : Les autres joueurs.

Description sommaire : Si le joueur tombe sur une case « propriété » et qu'elle n'appartient à personne, le joueur peut l'acheter au prix indiqué sur la case.

Si cette case appartient à un autre joueur, alors :

- s'il s'agit d'une propriété constructible, le joueur doit payer au propriétaire un loyer en fonction du nombre de constructions sur le terrain. Ces valeurs sont indiquées sur la carte de la propriété.
- si la case correspond à une gare, le joueur doit payer un loyer en fonction du nombre de gares que possède le propriétaire.
- s'il s'agit d'une compagnie, le joueur doit payer au propriétaire le montant prévu.

Si la propriété est hypothéquée le joueur ne paye pas d'amende. Si le joueur possède toutes les propriétés constructibles d'une même couleur, alors il peut construire des maisons en les achetant à la banque. Le joueur peut également construire des hôtels s'il possède au moins quatre maisons.

Règles d'initialisation :

1. Le joueur a lancé les dés.

Description et règles internes au processus :

Si le joueur arrive sur une case qui correspond à une propriété, le système vérifie si elle a déjà un propriétaire. Si ce n'est pas le cas, le système demande au joueur s'il veut l'acheter. S'il accepte, le système lui débite la somme correspondante et la verse à la banque.

Si la propriété appartient déjà à quelqu'un, le système l'indique au joueur et celui-ci doit payer un loyer. Le système prend l'argent au joueur et le donne au propriétaire. Le loyer est calculé par le système selon le type de la propriété. S'il s'agit d'un terrain constructible, le loyer dépend du nombre de constructions qui se trouve sur la propriété. S'il s'agit d'une gare, le système compte le nombre de gares que possède le propriétaire et calcule le loyer en fonction de ce nombre. Enfin, s'il s'agit d'une compagnie, le loyer varie selon le résultat affiché par les dés.

Si le joueur possède tous les propriétés constructibles d'une même couleur, le système lui demande s'il veut construire. S'il accepte, le système fait le transfert d'argent du solde du joueur vers la banque.

Règles de terminaison :

1. Le joueur possède une propriété.
2. Le joueur possède des maisons ou des hôtels.
3. Les capitaux du joueur de la banque ont été modifiés.

Exception : Le joueur est en prison.

Extension : RAS.

Compléments : RAS.

Titre du scénario 2 : Le joueur s'arrête sur une case Chance ou Caisse de communauté

Acteur primaire : Le joueur.

Acteurs secondaires : Aucun.

Description sommaire : Si le joueur arrive sur une case Chance ou Caisse de communauté, il doit tirer une carte correspondante qui lui indique l'action à effectuer. Si le joueur a tiré une carte sortie de prison, il peut la conserver jusqu'à ce qu'il décide de l'utiliser.

Règles d'initialisation :

1. Le joueur lance les dés.

Description et règles internes au processus :

Si le joueur s'arrête sur une case Chance ou Caisse de communauté, le système affiche une boîte de dialogue correspondant à l'intitulé d'une carte. Le système effectue alors l'action qui est indiquée (verser ou recevoir de l'argent, allez sur une case précise,...).

Si le système affiche une carte sortie de prison, alors il exécutera l'action correspondante seulement quand le joueur le lui demandera. Une icône apparaît alors sur l'écran pour signaler au joueur qu'il possède une carte sortie de prison.

Règles de terminaison :

1. Le joueur se retrouve sur une nouvelle case
2. Les capitaux du joueur, de la banque et du Parc gratuit ont été modifiés.
3. Le joueur possède une carte sortie de prison.

Exception : Le joueur est en prison.

Extension : RAS.

Compléments : RAS.

Titre du scénario 3 : Le joueur s'arrête sur une case taxe ou impôt

Acteur primaire : Le joueur.

Acteurs secondaires : Aucun.

Description sommaire : Si le joueur tombe sur une case de type « taxe » ou « impôt », il doit payer le montant qui est indiqué sur la case. L'argent va alors dans le parc gratuit.

Règles d'initialisation :

1. Le joueur lance les dés.

Description et règles internes au processus :

Si le joueur s'arrête sur une case taxe ou impôt, le système le signale au joueur et lui indique le prix qu'il doit payer. Le système débite alors le capital du joueur de la somme correspondante. L'argent est alors placé dans le parc gratuit.

Règles de terminaison :

1. Le capital du joueur a été diminué.
2. L'argent du Parc est augmenté.

Exception : Le joueur est en prison.

Extension : RAS.

Compléments : RAS.

Titre du scénario 4 : Le joueur s'arrête sur la case Parc gratuit

Acteur primaire : Le joueur.

Acteurs secondaires : Aucun.

Description sommaire : Si le joueur s'arrête sur la case « Parc gratuit », il reçoit l'argent qui s'y trouve.

Règles d'initialisation :

1. Le joueur lance les dés.

Description et règles internes au processus :

Si le joueur arrive sur la case Parc gratuit, alors le système augmente le capital du joueur de la somme présente dans le Parc. Le système remet le montant du Parc à zéro.

Règles de terminaison :

1. Le capital du joueur a été augmenté.
2. Le Parc est vide (il n'y a plus d'argent).

Exception : Le joueur est en prison.

Extension : RAS.

Compléments : RAS.

Titre du scénario 5 : Le joueur passe sur la case Départ

Acteur primaire : Le joueur.

Acteurs secondaires : Aucun.

Description sommaire : Si le joueur passe sur la case « Départ », il reçoit une prime (20000).

Règles d'initialisation :

1. Le joueur lance les dés.

Description et règles internes au processus :

Si le joueur passe sur la case Départ (lorsqu'il boucle un tour), le système, par l'intermédiaire de la banque, augmente le capital du joueur du montant de la prime.

Règles de terminaison :

1. Le capital du joueur a été augmenté.
2. Le capital de la banque a été diminué.

Exception : Le joueur va en prison.

Extension : RAS.

Compléments : RAS.

Titre du scénario 6 : Le joueur va en prison.

Acteur primaire : Le joueur.

Acteurs secondaires : Aucun.

Description sommaire : Si le joueur arrive sur la case « allez en prison », ou s'il tire une carte du même nom, ou enfin s'il fait un double avec les dés trois fois de suite, alors le joueur se retrouve sur la case prison.

Règles d'initialisation :

1. Le joueur lance les dés.
2. Le joueur n'est pas en prison.

Description et règles internes au processus :

Si le joueur arrive sur la case « allez en prison », le système envoie le joueur (représenté par un pion) sur la case prison.

Le système agit de la même manière lorsque le joueur tire une carte « allez en prison ».

Si le joueur, après avoir fait appel à la commande de jeu pour lancer les dés, fait un double, alors le système le déplace du nombre de case correspondant et annonce au joueur qu'il doit rejouer. Celui-ci doit alors s'exécuter en faisant appel à la même commande de jeu. Si le joueur effectue trois doubles à la suite, le système l'envoie en prison.

Règles de terminaison :

1. Le joueur est en prison.

Exception : Le joueur possède une carte sortie de prison.

Extension : RAS.

Compléments : RAS.

Titre du scénario 7 : Le joueur sort de prison.

Acteur primaire : Le joueur.

Acteurs secondaires : Aucun.

Description sommaire : Si le joueur fait un double avec les dés, alors il sort de prison et avance du nombre de cases indiqué par les dés. Il peut également sortir de prison en payant une somme d'argent (5000) ou s'il possède une carte sortie de prison. Si au bout de trois tours il n'a pas fait de double, le joueur est obligé de payer 5000 pour sortir.

Règles d'initialisation :

1. Le joueur est en prison.

Description et règles internes au processus :

Le joueur fait appel à la commande de jeu pour lancer les dés. S'il n'effectue pas un double, alors il ne se passe rien et le système passe au joueur suivant. Sinon, le système déplace le joueur selon le total des deux dés.

Si le joueur possède une carte sortie de prison, il peut le signaler au système et jouer en faisant appel à la commande qui permet de lancer les dés.

Si le joueur décide de sortir en payant, le système débite le joueur de la somme prévue (5000) et la verse à la banque.

Si le joueur n'a rempli aucune de ces conditions pendant trois tours, alors le système prélève de l'argent (5000) dans le capital du joueur et le transfère vers le Parc gratuit. Le joueur sort alors de prison.

Règles de terminaison :

1. Le joueur n'est pas en prison.
2. Le capital du joueur a été modifié.
3. L'argent du Parc gratuit est augmenté.

Exception : RAS.

Extension : RAS.

Compléments : RAS.

Titre du scénario 8 : Le joueur vend ou achète des biens à un autre joueur.

Acteur primaire : Le joueur.

Acteurs secondaires : Les autres joueurs.

Description sommaire : Le joueur peut vendre ses propriétés (si elles sont exemptes de construction) à un autre joueur, ou acheter des propriétés d'un autre joueur, à un prix convenu. Il peut également hypothéquer un terrain (toujours sans construction). Dans ce cas, la banque lui donne la somme prévue. Mais, s'il veut le racheter, alors il doit rendre à la banque l'argent qu'elle lui a versé, augmentée de 10%. Le joueur peut aussi vendre ou acheter une carte de sortie de prison à un autre joueur

Règles d'initialisation :

1. Chaque joueur concerné accepte la vente.

Description et règles internes au processus :

Si le joueur décide de vendre une de ses propriétés, ou d'en acheter une à un autre joueur, le système vérifie qu'il n'y a pas de construction. Ensuite, le système effectue l'échange d'argent en augmentant le capital du vendeur et en diminuant celui de l'acheteur de la valeur de la vente.

Si le joueur décide d'hypothéquer un de ces terrains, le système vérifie que cet endroit est vierge de construction. Puis, le système fait, de la même manière que précédemment, le transfert d'argent en remplaçant l'acheteur par la banque.

Si le joueur veut racheter une propriété qu'il a hypothéqué, le système lui prend une somme d'argent correspondant à celle que lui a donné la banque, lors de l'hypothèque, augmentée de 10%, et la verse à la banque.

Si le joueur décide de vendre, ou d'acheter, une carte sortie de prison à un autre joueur, le système met à jour les soldes des joueurs concernés par la vente de la même façon que dans le premier cas (augmentation ou diminution).

Règles de terminaison :

1. Le capital de chaque joueur concerné est modifié.
2. Le capital de la banque est modifié (lors d'une hypothèque).
3. Le nombre de propriétés de chaque joueur concerné est modifié.
4. La propriété change propriétaire.
5. La carte de sortie de prison change de propriétaire.

Exception : Le joueur est en prison.

Extension : RAS

Compléments : RAS

Titre du scénario 9 : Le joueur échange ses propriétés.

Acteur primaire : Le joueur.

Acteurs secondaires : Un autre joueur.

Description sommaire : Le joueur peut échanger une propriété exempte de construction avec une propriété d'un autre joueur si elles ont la même valeur.

Règles d'initialisation :

1. Chaque joueur concerné possède au moins une propriété.

Description et règles internes au processus :

Si le joueur décide d'échanger une de ses propriétés avec celle d'un autre joueur, il le signale au système. Celui-ci vérifie qu'elles sont exemptes de construction et qu'elles ont la même valeur. Si ces conditions sont respectées alors le système met à jour le nom des propriétaires.

Règles de terminaison :

1. Le nombre de propriétés de chaque joueur reste inchangé.

2. Les propriétés échangées ont un nouveau propriétaire.

Exception : RAS.

Extension : RAS

Compléments : RAS

Titre du scénario 10 : Le joueur fait faillite.

Acteur primaire : Le joueur.

Acteurs secondaires : Aucun.

Description sommaire : Le joueur fait faillite quand il n'a plus d'argent pour payer ces dettes et qu'il ne peut plus emprunter. Il doit se retirer du jeu.

Règles d'initialisation :

1. Le joueur ne peut plus rien payer.

Description et règles internes au processus :

Si le capital du joueur est à zéro, le système vérifie s'il peut encore emprunter. S'il ne peut pas, le système signale alors au joueur qu'il a fait faillite et qu'il est éliminé.

Règles de terminaison :

1. Le joueur n'a plus d'argent.

2. Le joueur est éliminé.

Exception : RAS

Extension : RAS

Compléments : RAS

Titre du scénario 11 : Fin de la partie.

Acteur primaire : Le joueur.

Acteurs secondaires : Un autre joueur.

Description sommaire : Le dernier joueur qui n'a pas fait faillite est le gagnant.

Règles d'initialisation :

1. Il ne reste que deux joueurs.

Description et règles internes au processus :

Si le joueur est le dernier à qui il reste de l'argent, alors le système le déclare vainqueur et termine la partie.

Règles de terminaison :

1. Il y a un vainqueur.
2. La partie est finie.

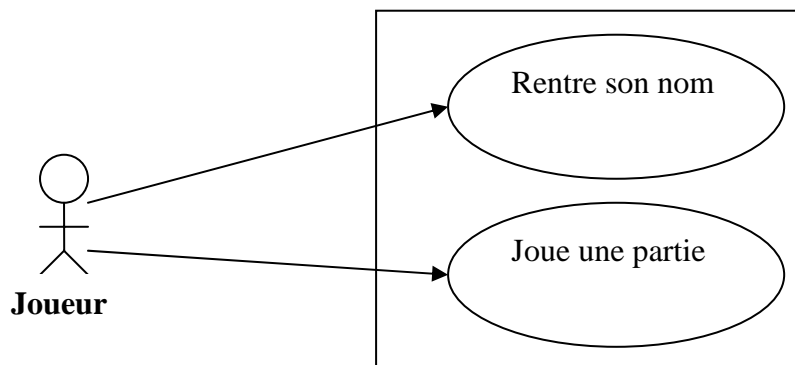
Exception : La partie a été interrompue volontairement par les joueurs. Dans ce cas, il n'y a pas de vainqueur.

Extension : RAS

Compléments : RAS

3. Le modèle statique

3.1. Les cas d'utilisations



3.2. le diagramme de classe

Grille d'extraction d'objets

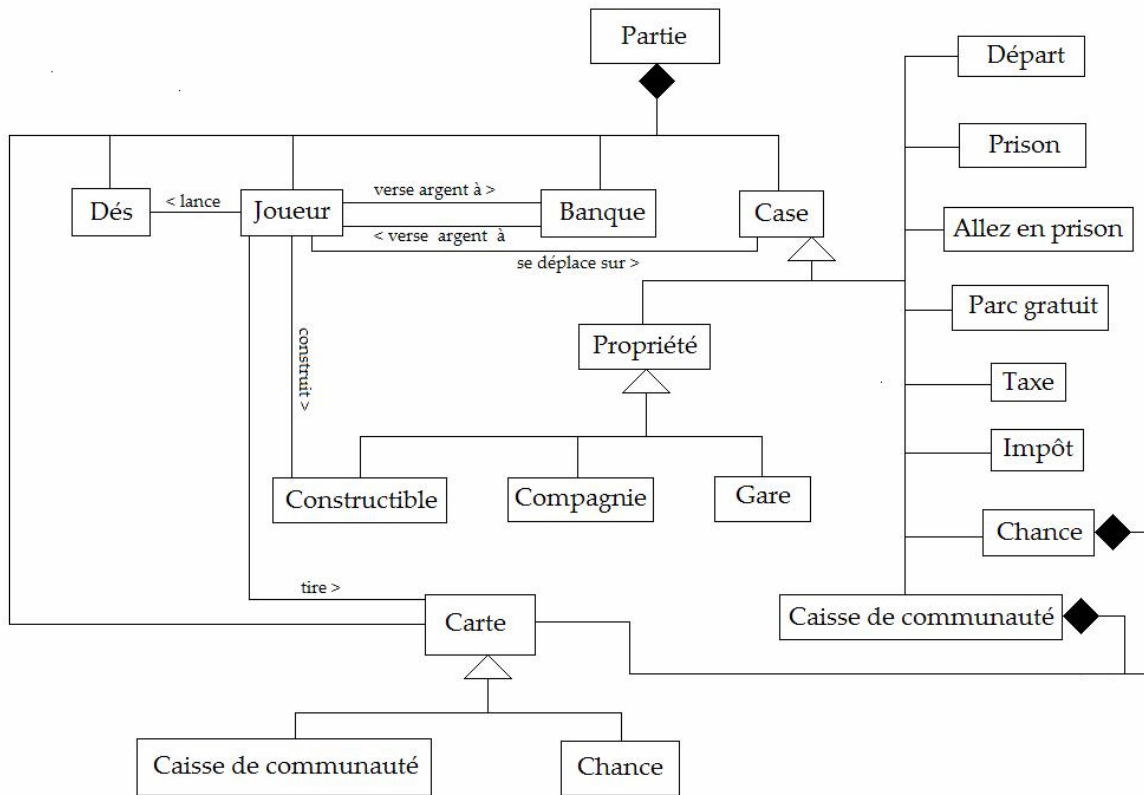
Objets	Attributs	états	opérations
<u>Joueur</u>	Nom	prisonnier	entre son nom
	Solde	libre	se déplace
	Position	riche	débite
	Propriétés	fauché	crédite
	Nombre de doubles	Éliminé	échange
	Nombre cartes de libération		construire
			vend
			achète
<u>Case</u>	type		renvoie son type
	nom		
	position		
<u>Case Départ</u>	somme		verse une somme
<u>Case communauté</u>			tirer une carte
<u>Case Chance</u>			tirer une carte
<u>Prison</u>	prisonniers		
<u>Allez en prison</u>	destination		envoiEnPrison
<u>Taxe de luxe</u>	somme		débite
<u>Impôt</u>	somme		débite
<u>Parc gratuit</u>	somme		verse reçoit
<u>Propriété</u>	type	hypothéqué	renvoyer nom propriétaire
	valeur	occupé	fait payer un loyer
	propriétaire	libre	
	prix loyer		
<u>Constructible</u>	prix construction	vierge	
-	nombre maisons	construit	
	couleur		
<u>Gare</u>			
<u>Compagnie</u>			
<u>Carte</u>	contenu numéro		piocher

Devoir de Génie logiciel : le jeu de Monopoly

Objets	Attributs	états	opérations
<u>Carte chance</u>			
<u>Carte caisse de communauté</u>			
<u>Banque</u>	solde		verse reçoit
<u>Dé</u>	valeur dé n°1 valeur dé n°2		lance dé isDouble
<u>Partie</u>	nombre de joueurs tableau de joueurs banque joueur courant tas de cartes chance tas de cartes de communauté parc gratuit positon joueur courant cases	initial en cours finie interrompue finie gagnée	gerePosition changeJoueur gerePrison gereGare gereConstructible gereCompagnie gereVente gereEchange

NB: On retrouve dans cette grille d'extraction des objets sans attributs et sans opérations mais cela est normal car ces objets héritent d'autres objets qui possèdent de attributs et opérations.

Diagramme de classes



Répertoire de classes

Identité : Joueur

Définition : cet objet permettra de modéliser un joueur du jeu. Ce joueur possédera plusieurs attributs qui le qualifieront.

Attributs :

- nom : Le nom du joueur, attribut qui permet de l'identifier
- position : sa position sur la grille de jeu. Elle est représentée par un chiffre compris entre 1 et 40.
- nbDoubles : Le nombre de jets consécutifs où le joueur a obtenu un double. Au maximum trois, ensuite ce compteur se réinitialise à zéro.
- statut : permet de savoir si le joueur est encore dans la partie. Cet attribut peut changer dans le cas d'une faillite ou d'un abandon de partie.
- nbTourPrison : permet de connaître le nombre de tour que le joueur a passé en prison (au bout de trois tours, il est libéré).
- Solde : la somme d'argent dont le joueur dispose.
- liste_Propriétés : la liste des propriétés que le joueur possède.
- nbCarteLiberation : le nombre de carte « sortie de prison » que le joueur possède.
- enPrison : permet de savoir si le joueur est emprisonné.

Méthodes :

Des méthodes d'accès aux attributs du joueur :

- getNom : Renvoie le nom du joueur.
- getSolde : Renvoie le solde du joueur.
- getNbCarteLiberation : qui permet de connaître le nombre de cartes de libérations du joueur.
- enPrison : permet de savoir si le joueur est emprisonné.
- getListe_terrain : renvoie les propriétés du joueur.
- getNbTourPrison : permet de connaître le nombre de tour que le joueur a passé en prison.
- getNbDouble : renvoie le nombre de doubles consécutifs que le joueur a obtenu.
- getNbPropriétés : permet de connaître le nombre de constructions que le joueur possède. Cette méthode sera utilisée dans le cas où le joueur reçoit une amende proportionnelle aux nombres de propriétés qu'il possède.
- isFaillite : permet de savoir si le joueur fait faillite.

Des méthodes de modifications des attributs du joueur.

- changeSolde : modifie le solde du joueur.
- setNbCarteLibération : modifie le nombre de cartes de sortie de prison du joueur.
- setEnPrison : permet de libérer ou d'emprisonner le joueur.
- setNbDouble : permet de changer le nombre de doubles consécutifs que le joueur a obtenu.
- setNbPropriétés : permet de modifier le nombre de propriétés que le joueur possède.
- deplace : permet de changer la position du joueur sur la grille de jeu.
- ajoutePropriete : permet d'ajouter une propriété dans la liste de propriétés possédées par le joueur.
- supprimePropriete : supprime une propriété dans la liste des propriétés du joueur.
- acheteTerrain : permet au joueur d'acheter un terrain. On regardera si le solde du joueur est suffisant pour l'acheter et on regardera si cette propriété n'a pas déjà un propriétaire.
- vendConstructible : permet de vendre une propriété constructible. On prendra soin de regarder s'il n'y a pas de construction sur ce dernier. Si c'est le cas on proposera au joueur de vendre ses propriétés. On enlèvera ce terrain de la liste des propriétés de ce joueur. On attribuera à la propriété un nouveau propriétaire : soit un autre joueur, soit la banque.
- construitMaison : permet de faire construire une maison sur une propriété constructible. On regardera si le solde du joueur est suffisant. On regardera aussi si la propriété appartient au joueur.
- vendMaison : permet de vendre une maison sur une propriété constructible à un certain prix. On change le solde du joueur, le solde de la banque, le nombre de maisons disponibles dans le jeu.
- echangeConstructible : permet d'échanger une propriété constructible avec un autre joueur. Les deux propriétés constructibles doivent être vierges de constructions.
- compteConstructibleMemeCouleur : permet de savoir combien de propriétés d'une même couleur le joueur possède.
- possedeGroupe : permet de savoir si le joueur possède l'ensemble des propriétés constructibles de la couleur de la propriété où il se trouve dans le cas d'une demande de construction, de vente de propriété constructible. Utilise

compteConstructibleMemeCouleur et compare le résultat avec l'attribut nbConstructibleMemeCouleur de la classe Constructible.

Identité : Case

Elle définit une case du plateau de jeu. Cette case pourra être une propriété constructible, une gare, un service, une case chance ou caisse de communauté, une prison, un parking, une taxe de luxe, une case départ, une simple visite en prison. Cet objet case sera donc la classe mère des types de cases citées.

Attributs :

- type : cet attribut définit le type de case : départ, chance, prison, communauté, constructible, parc, allez en prison, taxe.
- Position : il définit la position de la case sur la grille de jeu.
- Nom : le nom de la case (ex : rue de la paix, caisse communauté...).

Méthodes :

On définit des méthodes d'accès et de modifications des variables.

- getType : renvoie le type de case.
- getNom : renvoie l'intitulé de la case.
- getPosition : renvoie la position de la case sur la grille de jeu.

Identité : Case Départ

Elle définit la case départ et hérite de la classe Case.

Attributs :

- Somme : la somme qu'un joueur touche lorsqu'il passe par cette case

Méthodes :

- getSomme : permet de connaître le montant touché par un joueur lorsqu'il passe par cette case.

Identité : Case Communauté

Elle définit la case caisse de communauté et hérite de la classe case.

Attributs : ceux de la classe Case

Méthodes :

- `tireCarteCommunaute` : permet de tirer une carte dans le tas de cartes de la caisse de communauté. Une fois tirée la carte sera remise en dessous du paquet. Cette méthode va donc appeler un objet de type `carte` et plus particulièrement un objet qui hérite de la classe `carte` : `caisse de communauté`.

Identité : Case Chance

Elle définit la case chance et hérite de la classe `case`.

Attributs : Ceux de la classe `Case`

Méthodes :

- `tireCarteChance` : de la même façon que la case communauté, on pourra tirer une carte chance avec cette méthode. On appellera un objet de type `chance`.

Identité : Prison

Elle définit la case prison, c'est une sous classe de la classe `case`.

Attributs :

- `Prisonniers` : c'est un tableau de joueur. On peut donc connaître le nombre de joueurs en prisons ainsi que leurs caractéristiques.

Méthodes :

- `getPrisonniers` : renvoie la liste des prisonniers, qui sont des objets `joueur`.

Identité AllerEnPrison

Cette case, si elle est atteinte par joueur, envoie directement ce dernier en prison sans passer par la case départ et donc sans toucher la somme de la case départ.

Attributs :

- `destination` : indique la destination de la case prison, elle représentée sous forme d'un entier qui est la position de la case sur la grille de jeu.

Méthodes :

- `getDestination` : permet de connaître l'indice de la case prison pour y envoyer le joueur qui s'est arrêté sur cette case « aller en prison ».

Identité Taxe

Elle définit la case taxe de luxe et permet de faire payer le joueur qui s'arrête sur cette dernière. Elle hérite de la classe Case.

Attributs :

- somme : la somme de la taxe à payer.

Méthodes :

- getSomme : permet de connaître le montant de la taxe.

Identité Impôt

Elle définit la case impôt sur le revenu et permet de faire payer le joueur qui s'arrête sur cette dernière. Elle hérite de la classe Case.

Attributs :

- somme : la somme de l'impôt à payer.

Méthodes :

- getSomme : permet de connaître le montant de l'impôt.

Identité Parc Gratuit

Elle définit la case parc gratuit. Si un joueur s'arrête sur cette case on lui verse le montant du parc gratuit. Le montant provient des amendes et taxes payées par les joueurs. Elle hérite de la classe Case.

Attributs :

- somme : la somme qui est sur le parc.

Méthodes :

- getSomme : permet de connaître le montant du parc gratuit
- setSomme : permet de modifier le montant du parc. Cette méthode est utilisée quand le joueur doit payer une taxe ou une amende.

Identité Propriété

Elle définit les cases du terrain qui sont acquérables. Elle hérite de la classe Case

Attributs (elle hérite des attributs de la classe Case):

Le type de propriété : gare, constructible ou compagnie.
Un booléen pour savoir si la propriété est hypothéquée.
Sa valeur d'achat.
La valeur de l'hypothèque.
Un attribut de type joueur pour connaître le propriétaire.
Le nombre de propriétés de la même couleur

Méthodes :

Des méthodes d'accès aux attributs de la classe :
getColor qui renvoie la couleur de la propriété
getValeur qui renvoie la valeur d'achat de la propriété
getNbPropriétéMêmeCouleur qui renvoie le nombre de propriétés de même couleur
getValeurHypothèque qui renvoie la valeur de la propriété lors d'une hypothèque
getHypothèque qui permet de savoir si la propriété est hypothéquée
getPropriétaire qui permet de connaître le propriétaire
getType qui permet de connaître le propriétaire

Identité Constructible

Elle hérite de la classe propriété. Elle définit les terrains qui sont constructibles.

Attributs :

Un tableau d'entiers regroupant les différents loyers
Le prix d'une construction
Le nombre de constructions
Sa couleur.

Méthodes :

Des méthodes d'accès aux attributs de la classe

getLoyer qui renvoie le loyer en fonction du nombre de constructions
getNbConstructions qui renvoie le nombre de constructions
getPrixConstruction qui renvoie le prix d'une construction
setNbMaisonconstruite qui modifie le nombre de maisons construites

valeurAmende qui permet de connaître le montant du loyer en fonction du nombre de construction
getAmende qui donne l'amende au joueur visiteur

Identité Gare

Elle hérite de la classe Case et permet de définir une Gare.

Attributs : ceux de la classe Case

Un tableau d'entier regroupant les différents loyers

Méthodes :

getLoyer qui permet de connaître le montant du loyer à payer

valeurAmende qui permet de connaître le montant du loyer pour le joueur visiteur

getAmende qui permet de faire payer le loyer au joueur visiteur

Identité Compagnie

Elle hérite de la classe Case et permet de définir la compagnie des eaux et de l'électricité.

Les attributs :

Un tableau d'entiers regroupant les différents entiers

Méthodes :

getLoyer qui permet de connaître le montant du loyer à payer

valeurAmende qui permet de connaître le montant du loyer pour le joueur visiteur

getAmende qui permet de faire payer le loyer au joueur visiteur

Identité Caisse de communauté

Elle hérite des attributs de la classe Case et définit la case caisse de communauté.

Méthodes :

Tirecarte qui permet de tirer une carte dans le jeu de carte

Identité Chance

Elle hérite des attributs de la classe Case et définit la case chance.

Méthodes :

Tirecarte qui permet de tirer une carte dans le jeu de carte

Identité Carte

Elle définit une carte, qui peut être une carte caisse de communauté ou carte chance.

Attributs :

Son type : si c'est une amende simple, une amende en fonction du nombre de constructions, un déplacement à un endroit précis.

Le texte de la carte.

Un nombre : soit la valeur du gain, soit la valeur de l'amende, soit le déplacement.

Le prix par maison dans le cas d'une amende en fonction des propriétés qu'il y a sur le terrain.

Le prix par hôtel dans le cas d'une amende en fonction des propriétés sur le terrain.

Le rendez vous dans le cas d'un déplacement à un endroit précis.

Méthodes :

GetType qui renvoie le type de la carte.

getTexte qui renvoie l'intitulé de la carte.

getNombre qui renvoie soit le montant de l'amende, du gain ou encore le déplacement.

getParMaison qui renvoie le prix de l'amende dans le cas d'une amende proportionnelle au nombre de maisons du propriétaire.

getParhotel qui renvoie le prix de l'amende dans le cas d'une amende proportionnelle au nombre d'hôtels du propriétaire.

Identité Carte Chance

Les attributs sont ceux de la classe Carte et définit une carte chance.

Méthode :

getAction qui donne l'action à faire selon le type de carte : débit, crédit, déplacement.

Identité Carte caisse de communauté

Les attributs sont ceux de la classe Carte et définit une carte caisse de communauté.

Méthode :

getAction qui donne l'action à faire selon le type de carte : débit, crédit, déplacement.

Identité banque

Elle définit la banque.

Attributs :

Solde : le solde de la banque.

nbhotelDispo : le nombre d'hôtel que la banque possède.

nbMaisonDispo : le nombre de maisons que la banque possède.

Méthodes :

Verse : qui permet de verser une somme d'argent à un joueur.

Recoit : qui permet de percevoir une somme d'argent de la part d'un joueur.

getnbMaisonDispo : renvoie le nombre de maisons disponibles.

getnbHotelDispo : renvoie le nombre d'hôtels disponibles.

setNbHotelDispo : change le nombre d'hôtels disponibles.

setnbMaisonDispo : change le nombre de maisons disponibles.

Identité Dé

Elle permet de définir des dés et de les lancer.

Attributs :

Le nombre de faces des dés.

La valeur du dé numéro 1.

La valeur du dé numéro 2.

Les méthodes :

lanceDé qui permet au joueur de lancer les dés.

getDe qui permet de connaître la valeur du dé numéro 1 ou numéro 2.

isDouble qui permet de savoir si le joueur à fait un double.

Identité Partie

Cette classe permet de gérer le jeu et utilise tous les objets cités précédemment.

Attributs :

- Le nombre de joueur.
- Un tableau de joueur.
- La banque.
- Un tableau de cartes communauté.
- Un tableau de cartes chance.
- Qui est le joueur courant.
- Le nombre de joueurs restants.
- Un parc gratuit.
- L'indice du joueur courant.

Les méthodes :

- Méthode partie pour gérer une partie de jeu normal, elle fera appelle au méthode ci dessous.
- Méthode changeJoueur qui augmente de 1 l'indice du joueur courant pour passer au joueur suivant (si indice=nbJoueurs l'indice est remis à 0). On va regarder dans cette méthode si le joueur suivant a fait faillite, le nombre de joueurs restant pour déclarer un vainqueur quand il reste plus qu'un joueur (et bloquer les boutons dans les interfaces graphiques).
- Méthode gerePrison pour gérer un joueur en prison
- Méthode gerePosition pour savoir quoi faire en fonction du type de la case. Elle fera appelle au méthode suivante :

- Méthode actionCarte pour permet de savoir quelle action faire en fonction de la carte tirée quand on est tombé sur une case chance ou communauté.
- Méthode gereConstructible qui permettra de gérer les différentes actions possibles du joueur lorsqu'il tombe sur une propriété de type constructible(achat, loyer, vente, construction...).
- Méthode gereGare qui permettra de gérer les différentes actions possibles du joueur lorsqu'il tombe sur une gare (achat, loyer, vente, hypothèque...).
- Méthode gereCompagnie qui permettra de gérer les différentes actions possibles du joueur lorsqu'il tombe sur une compagnie (achat, loyer, vente, hypothèque...).

- Méthode gereDouble qui permettra de gérer les actions que le joueur doit faire quand il fait un double.

- Méthode gerePrison qui permet de gérer un joueur en prison (achat de carte de libération...).

- Méthode vente qui permet d'ouvrir une fenêtre pour permettre de vendre des terrains ou des constructions.

- Méthode echange qui permet d'ouvrir une fenêtre pour permettre d'échanger des terrains entre joueurs.

Méthode d'accès aux variables d'instances :

- getBanque
- setBanque
- getParc
- setParc
- getJoueurCourant
- getListe_Joueurs
- getCase
- getNbJoueur

4. Modèle dynamique

4.1. Le diagramme des séquences

Commentaires complétant le diagramme de séquence :

Lorsqu'un joueur est concerné par les étapes 3, 4, 5 et 6, il a d'abord effectué les actions de l'étape 2 intitulé « Le joueur joue ». En effet, avant de faire ce que lui indique les cases, le joueur doit lancer les dés et se déplacer sur les cases.

Commentaires complétant le diagramme de collaboration

Les numéros qui précèdent les interactions entre les objets correspondent aux numéros des étapes du diagramme de séquence (1 pour « Initialisation de la partie », 2 pour « Le joueur joue », 3 pour « Comportement du joueur par rapport à un propriété », ...).

Trace des événements:

NB : Ci-dessous nous utilisons des termes du type : « La partie modifie tel objet » et il faut bien comprendre que la partie fait appel à des méthodes de l'objet en question. La partie n'accède pas directement aux attributs de la classe de l'objet.

Initialisation

Choisir le nombre de joueurs.

Chaque joueur rentre son nom.

La partie initialise le solde des joueurs.

Tous les terrains sont mis vierges de construction avec des propriétaires nuls (ou autre que des joueurs). Le parc gratuit est mis à zéro.

On initialise le tas de carte chance et de communauté.

La partie désigne un premier joueur.

La partie en cours

Le joueur courant lance les dés.

Chaque dé est un nombre aléatoire choisit entre 1 et 6.

- Si le joueur obtient un double la partie demande au joueur de relancer les dés. La partie incrémente le compteur du nombre de double consécutif du joueur courant.
- Si le joueur obtient trois fois de suite un double, la partie envoie le joueur courant en prison. La partie change l'attribut en Prison du joueur et la prison ajoute le joueur à sa liste de prisonniers.
- Si ce n'est pas un double le joueur se déplace sur la grille (on modifie l'attribut position du joueur).
- Si il doit passer par la case départ on lui verse une certaine somme (diminution du montant de la banque).

Le joueur tombe sur une case de type constructible (la partie regarde l'attribut type de la case).

La partie regarde le propriétaire du terrain

- Si le joueur courant est le propriétaire de la propriété, la partie regarde si le joueur peut construire. La partie regarde donc si le joueur possède toutes les propriétés d'un même groupe de couleur.
 - S'il peut construire, la partie propose au joueur de construire des maisons ou un hôtel (s'il a déjà 4 maisons sur la propriété).
 - Si le joueur décide de construire la banque lui vend des maisons ou des hôtels (seulement si le solde du joueur est suffisant). La partie augmente le solde de la banque, diminue celui du joueur de la valeur

des propriétés acquises et modifie l'attribut du nombre de maisons construites sur la propriété.

- Si le joueur courant n'est pas propriétaire de la propriété et qu'un autre joueur l'est, le joueur doit verser un loyer au joueur propriétaire, excepté si cette propriété est hypothéquée. La partie regarde le nombre de constructions sur la propriété et récupère le prix du loyer.
 - S'il peut payer le loyer, la partie débite sur le solde du joueur courant le montant du loyer et verse au joueur propriétaire ce montant.
 - Si le joueur courant ne peut pas payer le loyer, la partie lui propose de vendre des terrains s'il en possède. La partie peut aussi lui permettre d'hypothéquer des terrains.
- Dans le cas de la vente, la banque rachète les constructions et terrains à 50% de leurs valeurs. Dans le cas du rachat par la banque, la partie créditera le solde du joueur, diminuera celui de la banque. La partie enlèvera la propriété dans la liste des propriétés du joueur et mettra l'attribut propriétaire (de la propriété) à NULL.
- Dans le cas d'une hypothèque la partie changera l'attribut hypothèque de la propriété. Le joueur pourra lever l'hypothèque en payant le prix de la propriété, majoré de 20%.

Si le joueur tombe sur une case gare

La partie regarde le propriétaire de la gare (attribut propriétaire de la case)

- Si le joueur n'est pas le propriétaire de la gare mais qu'un autre joueur l'est alors le joueur courant doit verser un loyer proportionnel aux nombres de gares que le propriétaire possède sauf si la gare est hypothéquée. La partie regarde le nombre de gares que le joueur propriétaire possède (liste des propriétés dans la classe joueur) et regarde le prix à payer en fonction (appel de la méthode `getprixLoyer` de la classe `gare` en fonction du nombre de gare).
 - Si le joueur a un solde suffisant, la partie débite le solde du joueur courant du montant du loyer et verse au propriétaire le loyer.
 - Si le joueur n'a pas assez d'argent, la partie va proposer au joueur de vendre des biens de la même façon que pour les propriétés constructibles. S'il n'a pas de bien le joueur sera mis en faillite.
- Si le joueur est le propriétaire alors il ne paye pas d'amende.

Si le joueur tombe sur une case service (caisse de communauté, chance)

La partie propose au joueur de tirer une carte. Le joueur pioche une carte (méthode de la classe `carte`) et la carte applique l'action à faire.

- Si c'est une amende la partie débite le joueur de la valeur de l'amende.

- Si le joueur n'a pas assez d'argent pour payer son amende, la partie va lui proposer de vendre des biens comme pour les gares et constructibles ci-dessus. S'il ne peut pas couvrir sa dette le joueur sera mis en faillite (attribut faillite du joueur).
- Si le joueur peut payer l'amende la partie débitera le compte du joueur et mettra cette somme dans le parc gratuit.
- Si l'action est un gain, la partie crédite le montant du gain sur le compte du joueur et débite la banque du montant du gain.
- Si l'action est un déplacement la partie va déplacer le joueur sur la grille à la position indiquée sur la carte.

Le joueur tombe sur une case Allez en prison

La partie envoie le joueur en prison : modification de l'attribut prisonnier du joueur et modification de la liste des joueurs prisonniers de la classe prison.

Si le joueur doit passer par la case départ pour rejoindre la prison il ne touchera pas la somme de la case départ.

Si le joueur possède une carte sortie de prison la partie lui demandera s'il veut l'utiliser. Si c'est le cas la partie lui enlève sa carte de libération et propose au joueur de relancer les dés.

Le joueur passe sur la case départ

Si le joueur passe sur la case départ, excepté le cas où il se dirige en prison, le joueur touche la somme de la case départ. Pour se faire la partie crédite le joueur de la somme et débite la banque.

Le joueur tombe sur une case parc gratuit

Le joueur touche le montant du parc gratuit. La partie augmente le solde du joueur de la somme du parc et remet le montant du parc à zéro.

Le joueur tombe sur la case simple visite en prison

Dans ce cas la partie n'envoie le joueur prison, il reste juste positionné sur la case.

Le joueur tombe sur la case « taxe de luxe » ou « impôt sur le revenu »

La partie débitera le solde du joueur du montant de la taxe ou de l'impôt. Cette somme débitée doit être mise dans le parc gratuit.

Le joueur est en prison

Si le joueur est en prison, la partie regarde si le joueur possède une carte de libération et si c'est le cas la partie lui demande de l'utiliser.

S'il ne possède pas de carte de sortie de prison le joueur peut payer pour sortir de prison. L'argent va dans le parc gratuit.

Si au bout de trois tour de jeu le joueur n'est pas sorti de prison il doit obligatoirement payer une amende pour en sortir.

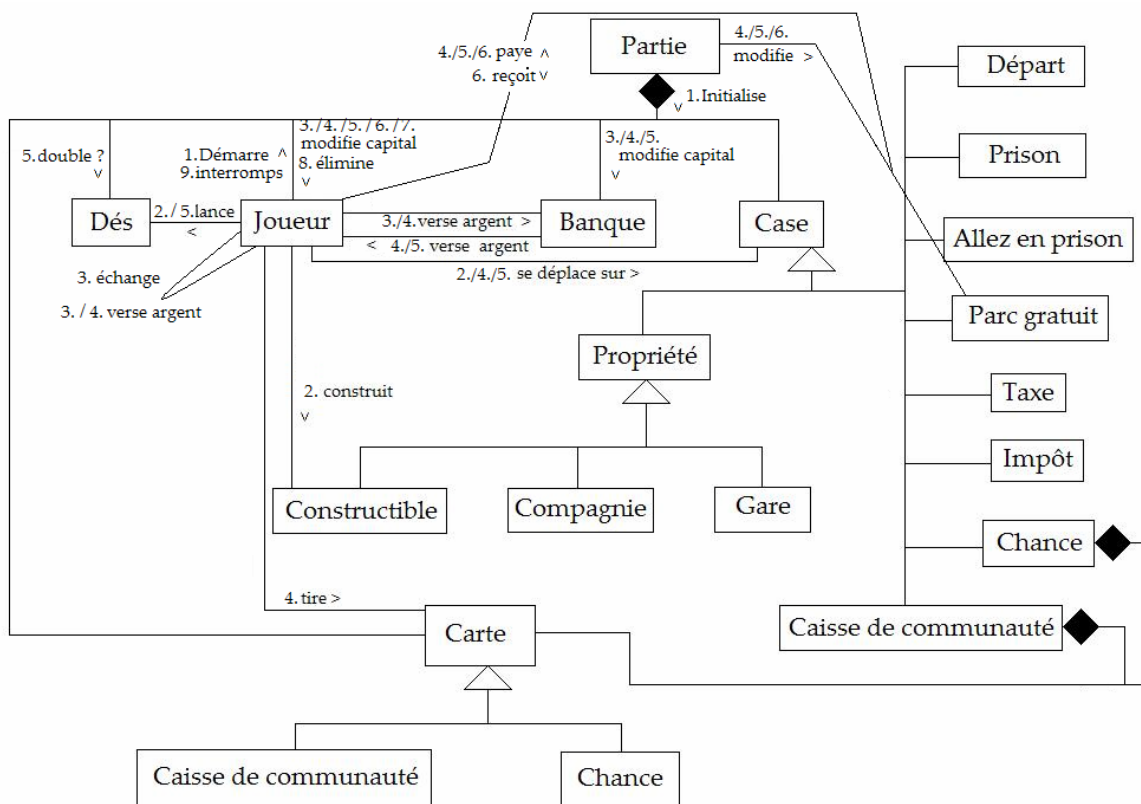
Le joueur est en faillite

Si le joueur a un solde négatif et qu'il n'a plus de bien alors la partie le met en faillite, la partie diminuera son nombre de joueurs actifs.

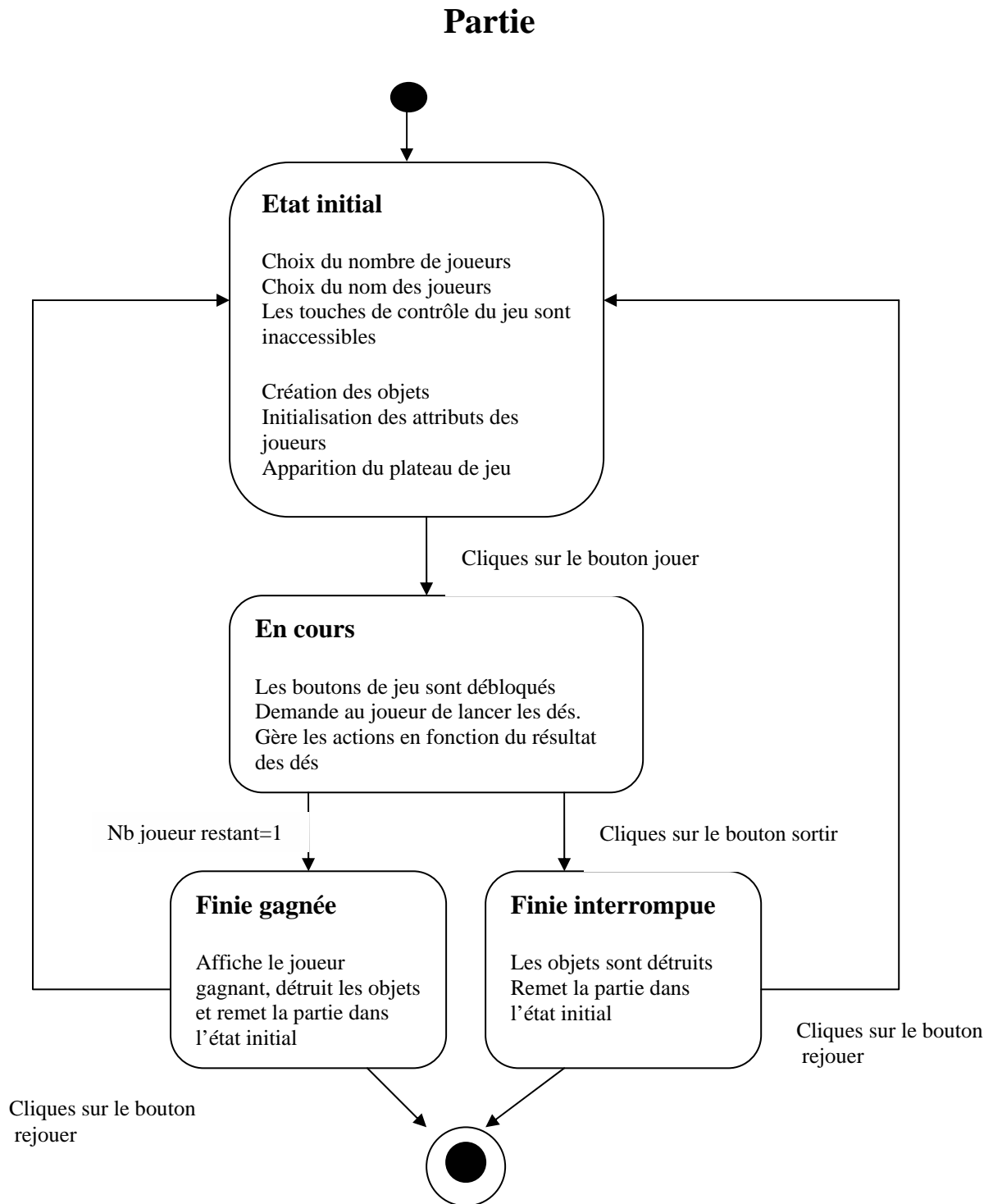
La partie se termine

A chaque fois que l'on change de joueur la partie regarde le nombre de joueurs restants. S'il en reste un alors le joueur restant est déclaré vainqueur sinon la partie continue. Les joueurs peuvent décider d'arrêter la partie en cours par une action sur un bouton.

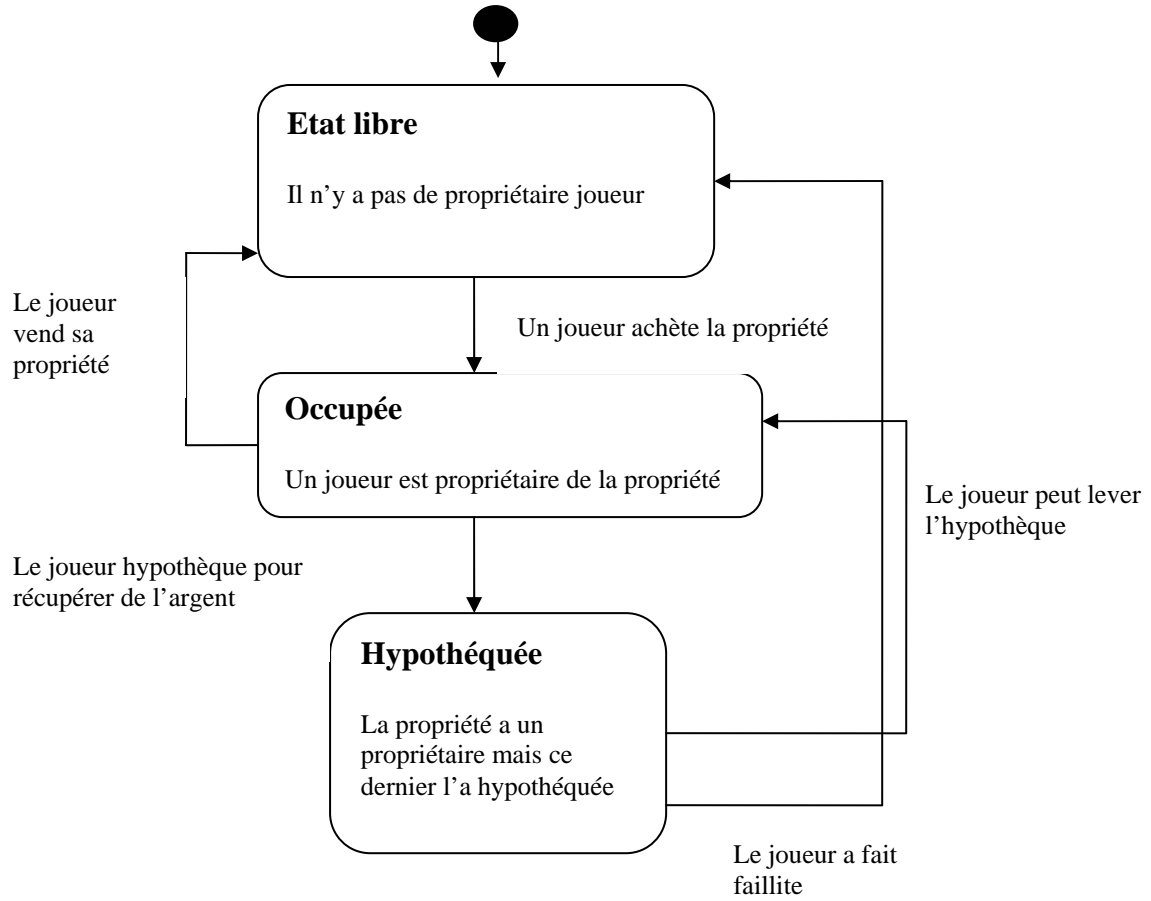
4.2. Le diagramme de collaboration



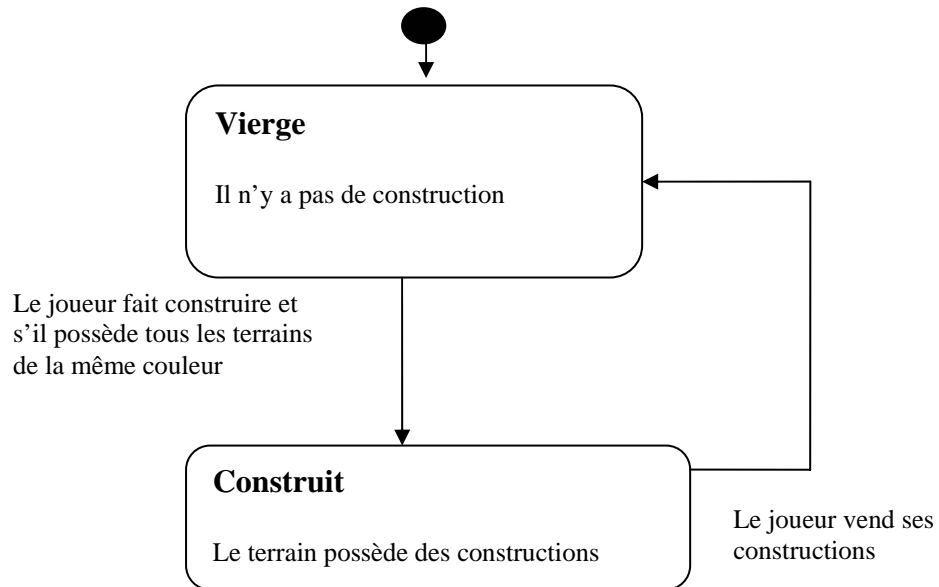
4.3. Le diagramme des états des objets



Propriété



Constructible



Joueur

