Génie logiciel et Projet de programmation

Le jeux de Monopoly

Projet de réalisation de jeux monopoly au cours de la licence 3 Informatique.

Réalisé par: F.AGHRI et A.TAFAT

Encadré par: Ben Salem

Sommaire

Descriptif du jeux Monopoly

1. Le principe du jeux:

Dans ce projet nous utilisons nos compétences pour réaliser le jeu de Monopoly en java de la modélisation UML jusqu'à la conception du jeu. Afin de mener à bien ce projet, nous avons reçu un cahier des charges qu'il nous a fallu respecter afin de correspondre aux attentes du client.

Monopoly est un jeu de société créé en 1931 par **Charles Darrow**; où l'on vend, achète ou loue des propriétés de manière profitable, afin que les joueurs puissent s'enrichir (le plus riche d'entre eux étant déclaré vainqueur). Les joueurs partent de la case « Départ » et déplacent leur pion sur le plateau en fonction du résultat des dés. Lorsqu'un joueur parvient sur la case d'une propriété à vendre, il peut l'acheter à la Banque. S'il ne désire pas le faire, la propriété est proposée aux autres joueurs et ira au plus offrant. Lorsqu'un joueur s'arrête sur la propriété d'un autre joueur, il doit lui verser un loyer. La construction de Maisons ou d'Hôtels augmente considérablement le montant des loyers ; il est donc sage de construire sur ses terrains. Les cases Chance et Communauté permettent de tirer une carte dont il faut suivre les instructions. Il arrive aussi que l'on se retrouve en Prison. Le but du jeu est d'être le dernier joueur à rester en jeu, c'est-à-dire le dernier joueur n'ayant pas fait faillite.

2. Régles du jeux:

Matériel :

- 1 plateau de jeu
- 28 Titres de propriété
- 16 cartes Chance
- 16 cartes Caisse de Communauté
- 1 liasse de billets Monopoly
- 32 Maisons
- 12 Hôtels
- 2 dés

- **2** Pion

Préparation :

1. Chaque joueur entre son nom et se place sur la case "Départ" avec une couleur.

2. Le Banquier remet à chaque joueur une somme de 150.000 euros. La banque possède les Titres de Propriété, les Maisons et les Hôtels. La Banque paye les salaires et les primes, encaisse l'argent des impôts, des taxes, des pénalités. Quand des enchères ont lieu, le Banquier tient le rôle du Commissaire Priseur. La Banque ne peut jamais "faire faillite" (on supposera qu'elle détient une quantité infinie d'argent).

3. Chacun des joueurs lance ensuite les deux dés : le joueur qui obtient le score le plus élevé commence la partie, à sa suite c'est au tour du deuxième joueur.

Au cours d'une partie:

Dans un tour de jeux, chaque joueur peut effectuer l'une des opérations suivantes:

- acheter un terrain ou une autre propriété qui peut-être :une gare/compagnie/ des maisons pour construire des hotel (à partir de 4 maisons),
- payer un loyer (si la propriété appartient à un autre joueur),
- payer des impôts,
- tirer une carte "Chance" ou "Caisse de Communauté",
- aller en Prison,
- bénéficier du Parc Gratuit,
- toucher un salaire de 20.000 euros.

Un double dé permet de jouer deux tours consécutifs.un 3éme vous met en Prison !

3.Méthodologie du travail

Avant de commencer le développement , on a procédé à la première étape qui est la pré analyse, cette étape consiste a spécifier les besoins et les objectifs que notre projet doit réaliser, cela nous a conduit à faire notre diagramme de cas d'utilisation.

Ensuite on fait un diagramme de classes provisoire(afin de l'améliorer dans l'étape du développement) ,on a implémenter ce dernier avec Umbrello .qui a en parallèl générer les classes

avant d'entamer l'étape concrète du pure développement ,on a choisi un environnement(java) qui nous facilite l'implémentation de notre interface graphique, qu'on a codé en premier.

après avoir fait une interface basique , on a commencé à remplir notre code généré précédemment par umbrello . au fur à mesure on remplit notre diagramme de classe pour avoir un diagramme final .

a la fin de l'étape de développement on a put avoir les interactions entre nos entités ce qui nous a permi d'avoir les diagrammes de séquence finals, après avoir déjà fait dans un premier temps (des exemples simple avec une entité d'interaction nommée **Système**)

Les besoins fonctionnels:

Après du cahier de charge, on a pu extraire les besoins du client , dont les besoins fonctionnels sont les suivants:

- Le joueur utilise le clavier et la souris pour jouer.
- Les deux joueurs se déplacent sur un plateau de jeu qui dispose 40 cases, 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 de **150 000 euros.** et **20 000** à chaque passage dans la case de départ.
- Un premier joueur lance les dés et se déplace du nombre de case du résultat.
- Chaque joueur s'arrêtant sur la propriété d'un autre joueur devra payer un loyer; le loyer des hotel est en double.
- 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 dernier joueur restant sera le vainqueur et aura le monopole de la ville.

Actions principales sur le menu:

- **Une interface (**qui s'ouvre à l'exécution du programme**):**
    1. **Entrer le nom des joueurs**
    2. **lancer la partie (en appuyant sur << Jouer >>)**
    3. **Quitter**

- **Une deuxième interface (au cours de la partie de jeux) où le joueur peut:**
    1. **Lance la partie**
    2. **Se déplace**
    3. **Achète**
    4. **Vend**
    5. **Construit ou avoir un bien**
    6. **Verse de l'argent**
    7. **Reçoit de l'argent**
    8. **Hypothèque**
        - ➔ **A chaque instant les joueurs peuvent << Quitter le jeu >>**

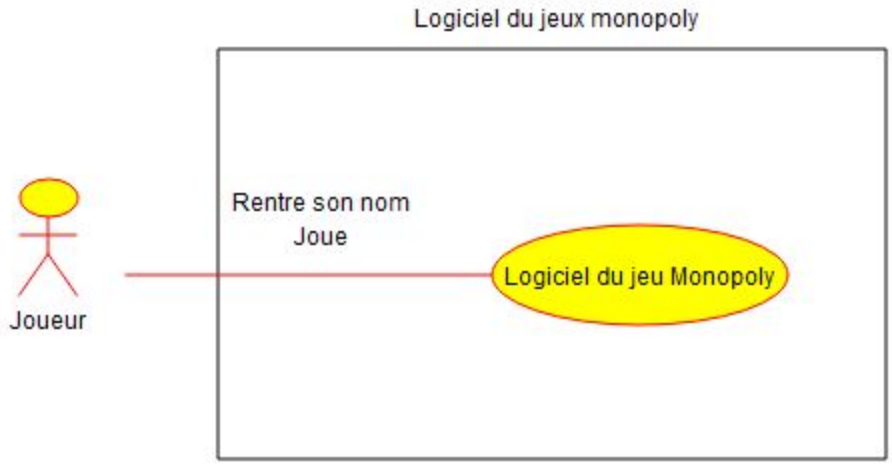
La partie UML du projet

Modélisation des diagrammes des cas d'utilisations

Dans cette partie on décrit les scénarios possible dans une partie du jeux avec les diagrammes de cas d'utilisations correspondants pour chaque cas possible.

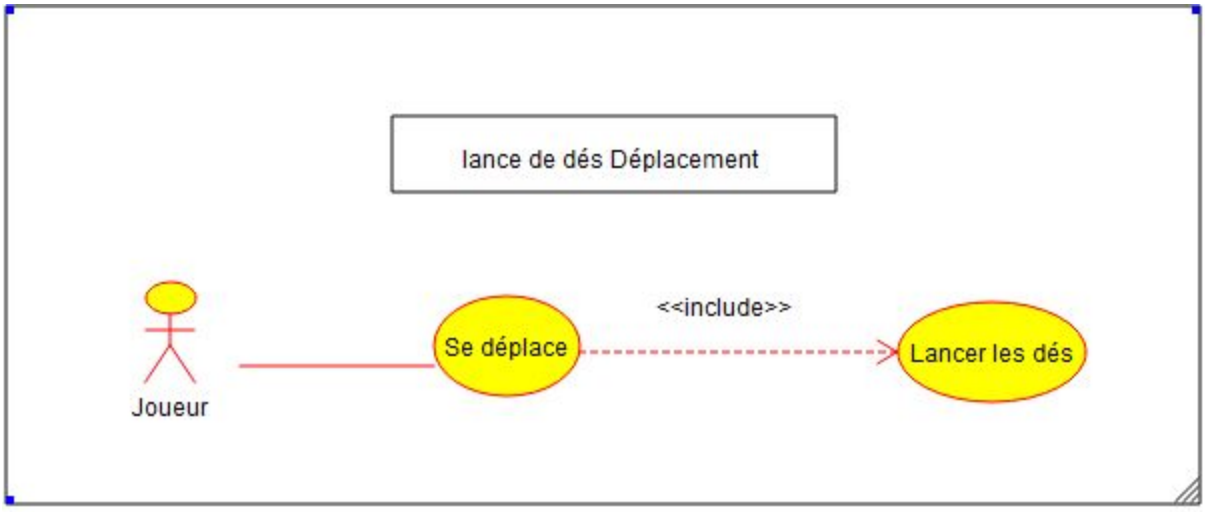
1. Dans un premier temps, si on voit le jeux d'une manière globale, on constate que le joueur est un acteur primaire du logiciel.

   Le diagramme de cas d'utilisation correspondant:



On voit bien que ce cas d'utilisation n'apporte pas vraiment d'avantages, il est beaucoup très générique.

2. **Les cas possibles dans un tour pour un joueur :**
   a. **Le joueur lance les "dés" et se déplace:**



b. Le joueur s'arrête sur une case Chance ou Caisse de communauté

Si le joueur se positionne sur une case <<chance>> ou <<Caisse de communauté >> ;

Acteur primaire : Le joueur

Acteurs secondaires : Aucun

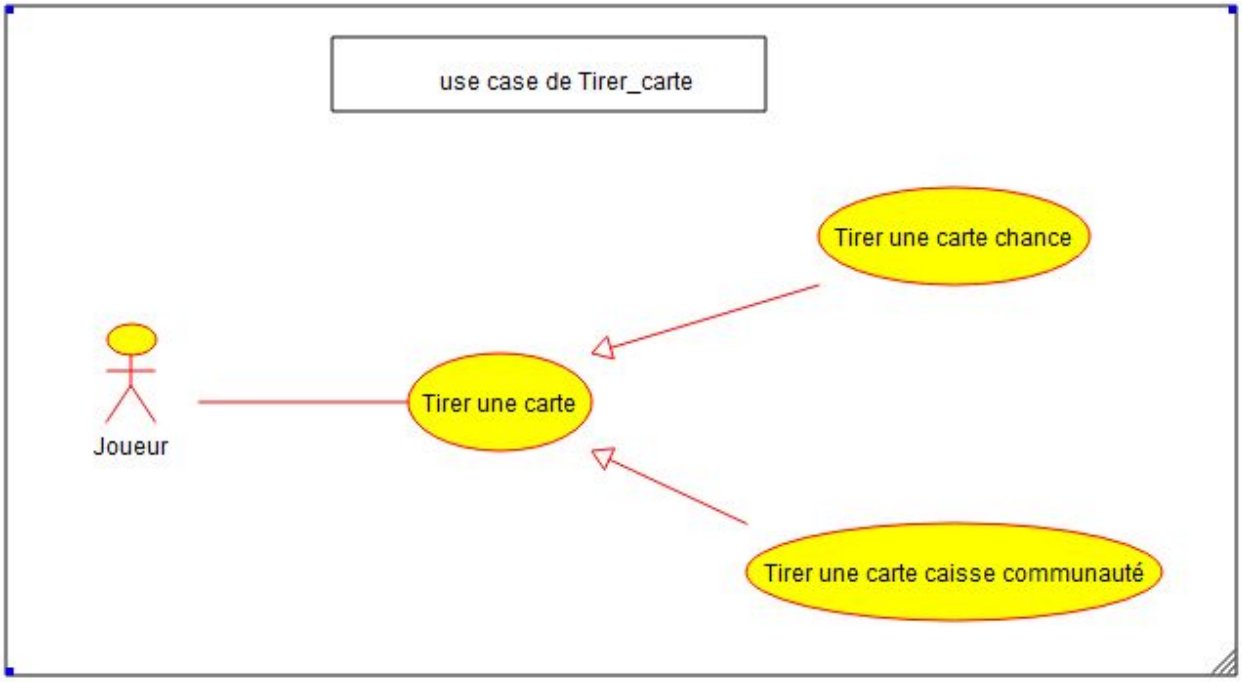
Initialisation: Le joueur lance les dés.

Exception : Le joueur est en prison.

***Description du 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,...).

Exception : Le joueur est en prison

Le diagramme de cas d'utilisation correspondant:



c. Le joueur s'arrête sur une propriété:

Si le joueur s'arrête sur une case de type <<Propriété>>

Acteur primaire : Le joueur

Acteurs secondaires : les autres joueurs

Initialisation:

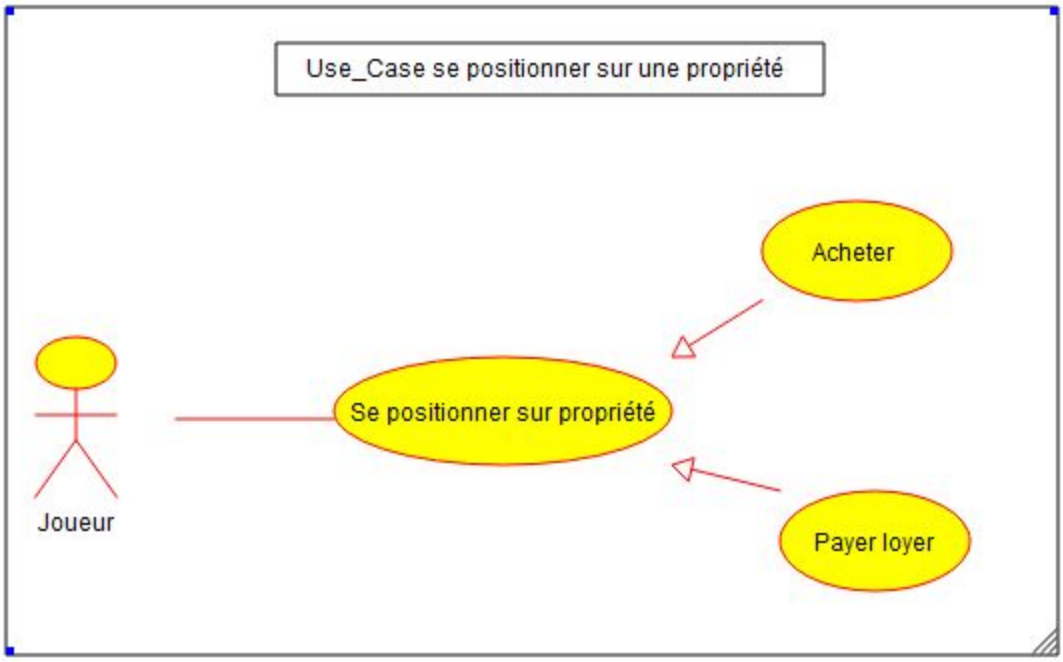
Le joueur lance les dés.

Tirer une carte chance qui envoie le joueur dans une case <<propriété>>

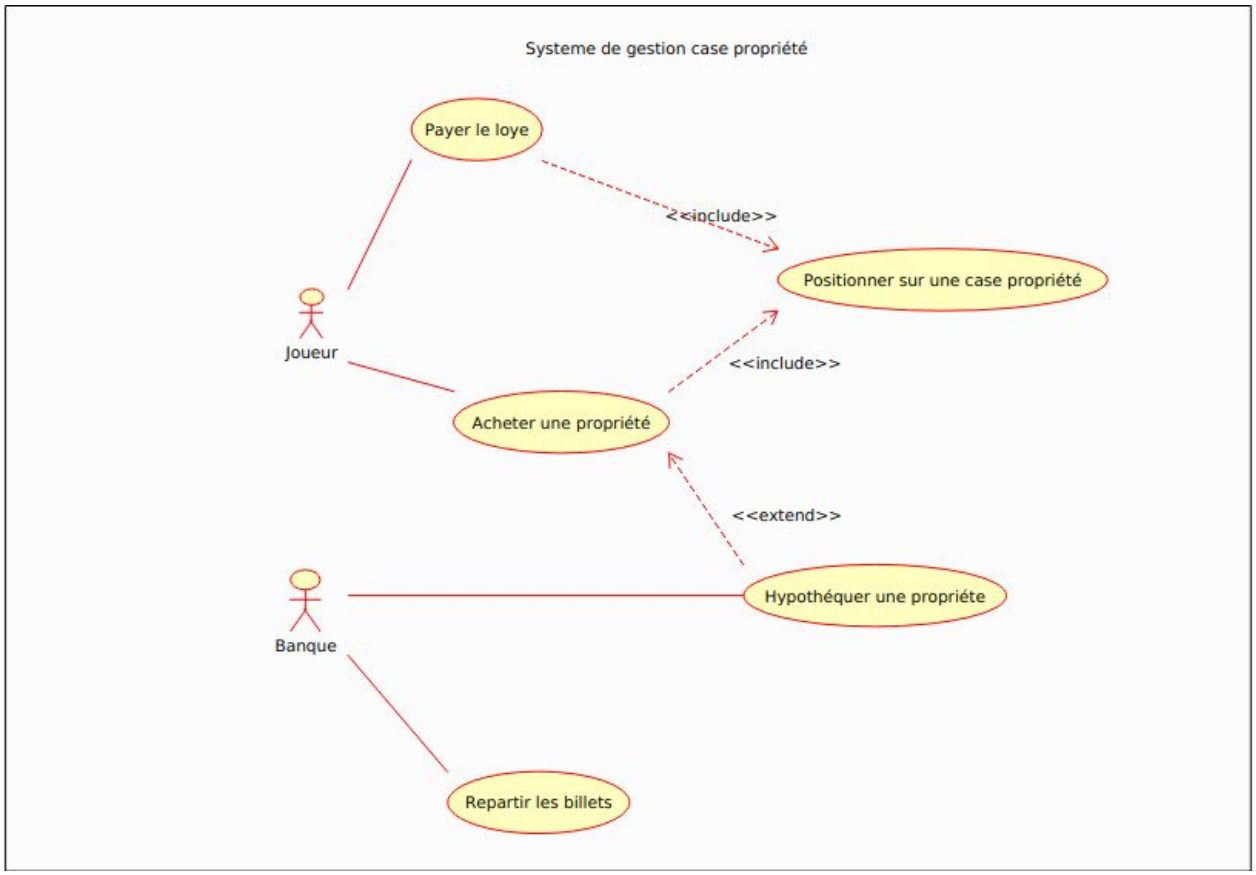
***Description du processus*:** Si le joueur arrive sur une case qui correspond à une propriété, le système vérifie si elle a déjà un propriétaire. Si ce n'est pas le cas, le système demande au joueur s'il veut l'acheter. S'il accepte, le système lui débite la somme correspondante et la verse à la banque.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

propriété appartient déjà à quelqu'un, le système l'indique au joueur et celui-ci doit payer un loyer.Une propriété peut s'agir aussi d'une **<< gare >>** ou d'une **<< compagnie >>**

Exception : Le joueur est en prison



Le diagramme de cas d'utilisation correspondant: **(avec les actions que le joueurs peut faire sur une propriété)**



d. Le joueur s'arrête sur une case taxe ou impôt:

Acteur primaire : Le joueur

Acteurs secondaires : aucun (car il ne peut pas y avoir de transactions avec les autres joueurs)

Initialisation:

Le joueur lance les dés.

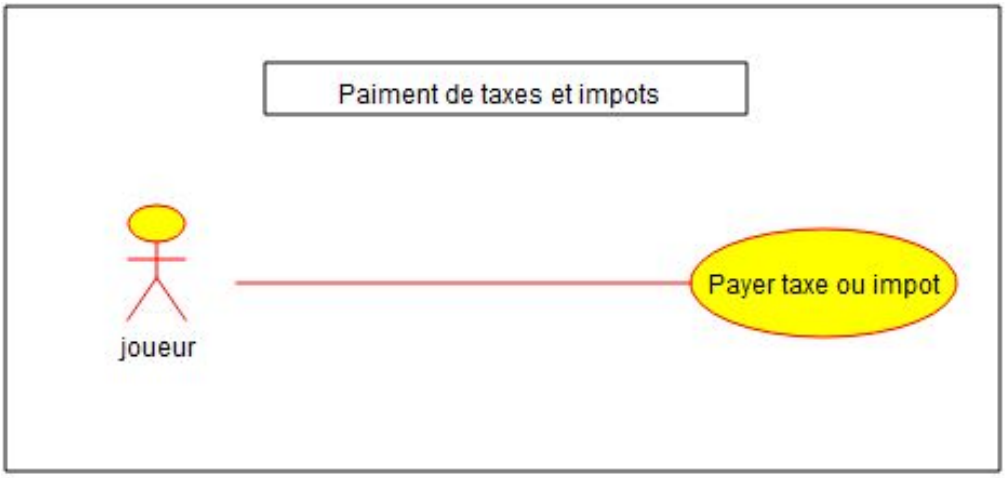
Tirer une carte chance qui envoie le joueur dans une case << taxe >>

Exception : Le joueur est en prison.

***Description du processus*:** , il doit payer le montant qui est indiqué sur la case; Le système débite alors le capital du joueur de la somme correspondante.

Exception : Le joueur est en prison

Le diagramme de cas d'utilisation correspondant:



e. Le joueur va en Prison :

Acteur primaire : Le joueur

Acteurs secondaires : aucun (car il ne peut pas y avoir de transactions avec les autres joueurs)

Initialisation:

Le joueur lance les dés.

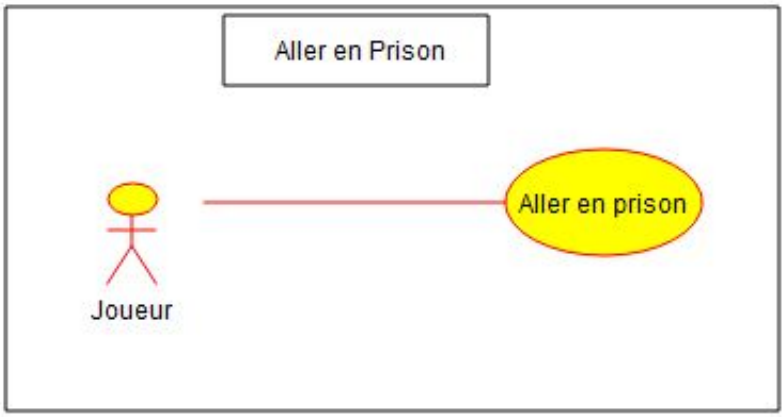
Il fait un double avec les dés trois fois de suite

Tirer une carte chance qui envoie le joueur dans une case << aller en prison >>

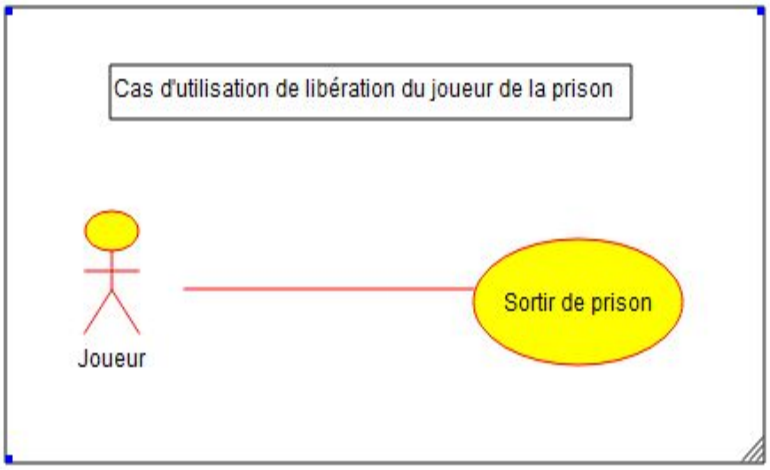
Exception : Le joueur possède une carte sortie de prison.

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

Exception : Le joueur est en prison



Le joueur peut bien sûr sortir de la prison (heureusement !)



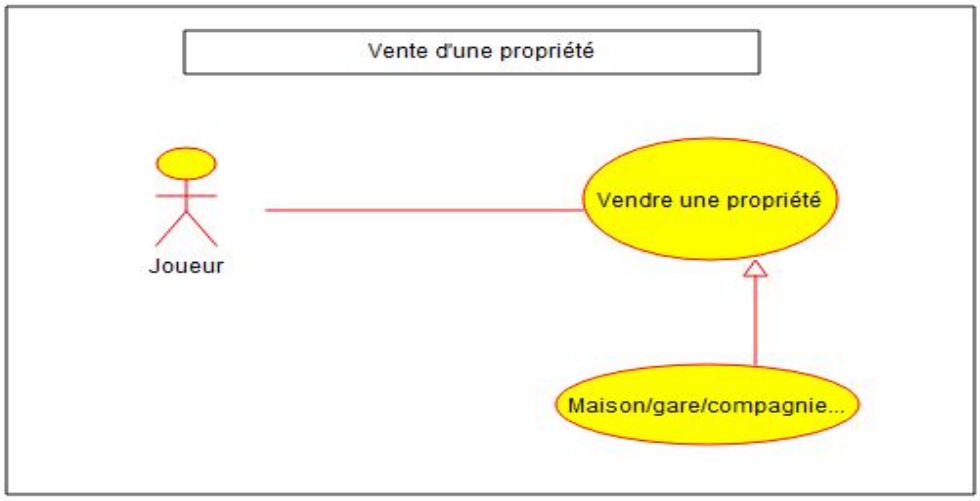
f. Le joueur vend une propriété :

Acteur primaire : Le joueur.

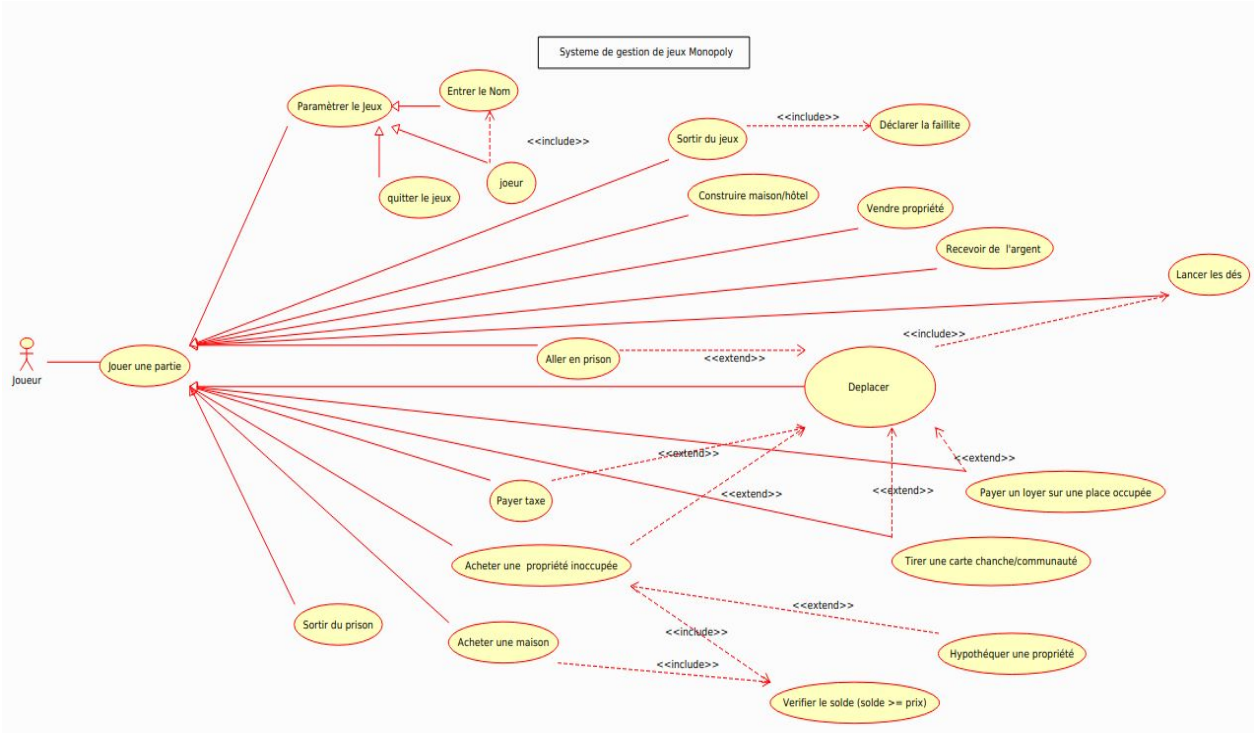
Acteurs secondaires **:** Un autre joueur.

Règles d'initialisation : 1. Chaque joueur concerné possède au moins une propriété.

Le diagramme correspondant:



Cela nous a conduit à ce diagramme général suivant:



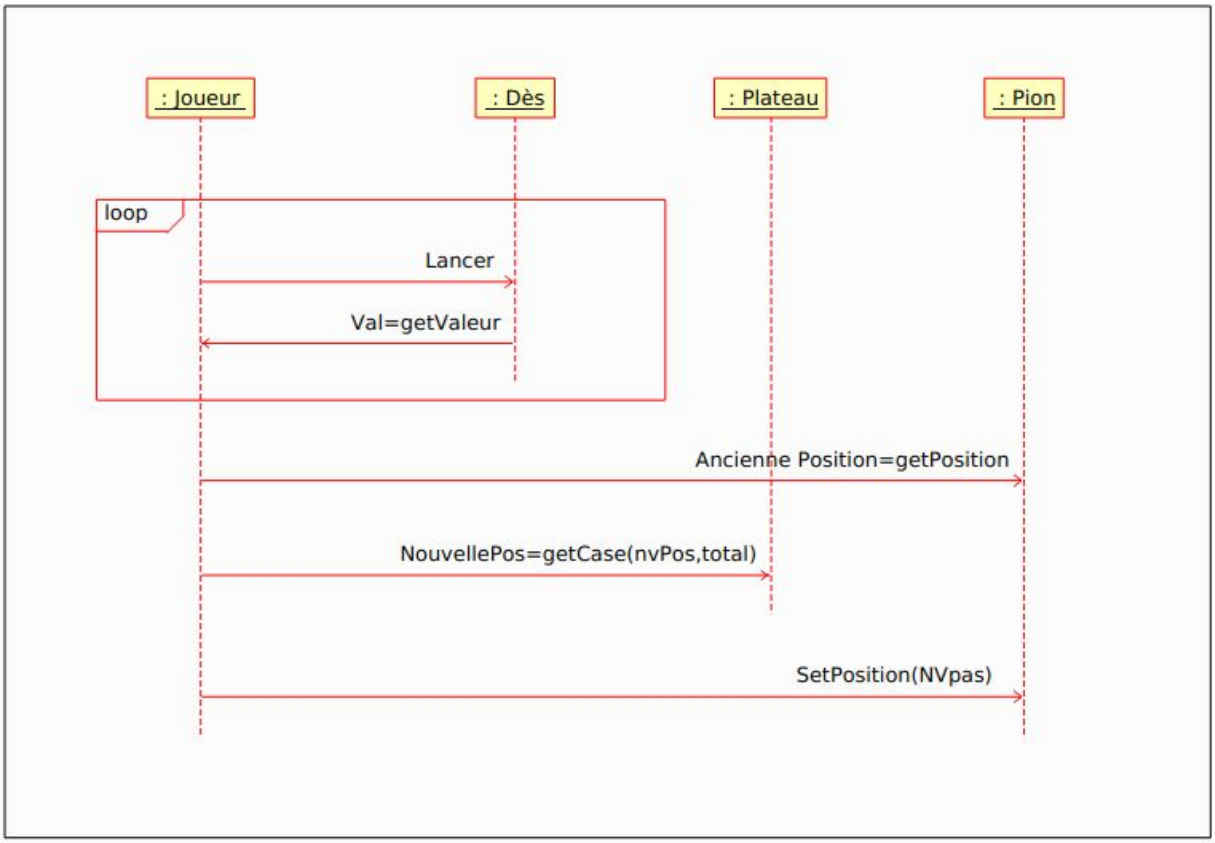
De “use case “ au diagramme de séquence

On a choisi quelques diagrammes de cas d’utilisations afin de produire des diagrammes de séquences pour quelques cas qu’on a constaté importants:

- **Diagramme de séquence pour le lancement de “dés“et le “déplacement” de pion :**

Description :

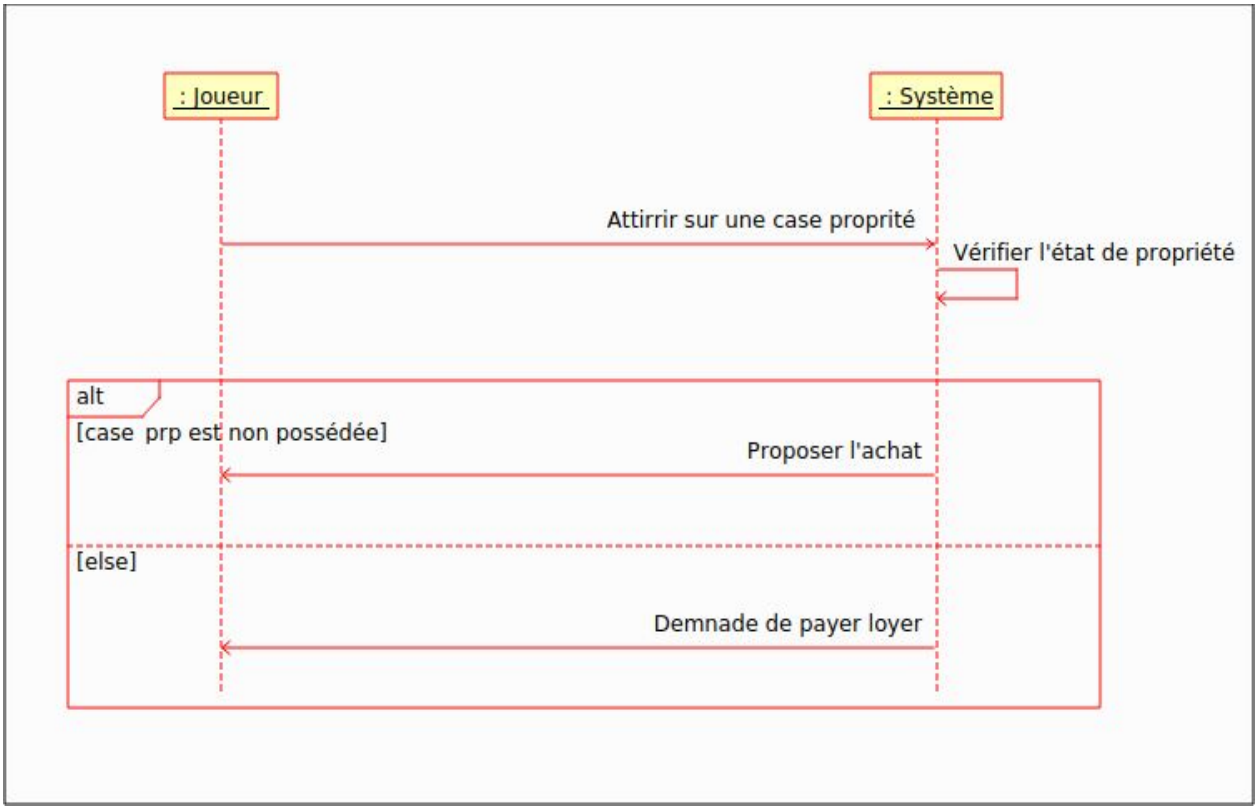
le joueur lance les dés et obtient la valeur du lancement , récupère l’ancienne position du pion, le plateau ensuite modifie la position selon le total obtenu, la position du joueur est donc modifiée.



- **Diagramme de séquence pour l'opération "*se positionner sur une propriété et processus d'achat* " :**

Le Joueur se positionne sur une *Case Propriété* . Le *système* alors compare alors si ce joueur est propriétaire ou pas. Et si la Case n'est pas possédée, il envoie le message "*proposer l'achat*" au Joueur, le joueur confirme l'achat(ou non); dans le cas où la propriété appartient à un autre joueur; il envoie "*demande de payer le loyer* " ,son capital sera donc débité de la somme du loyer.

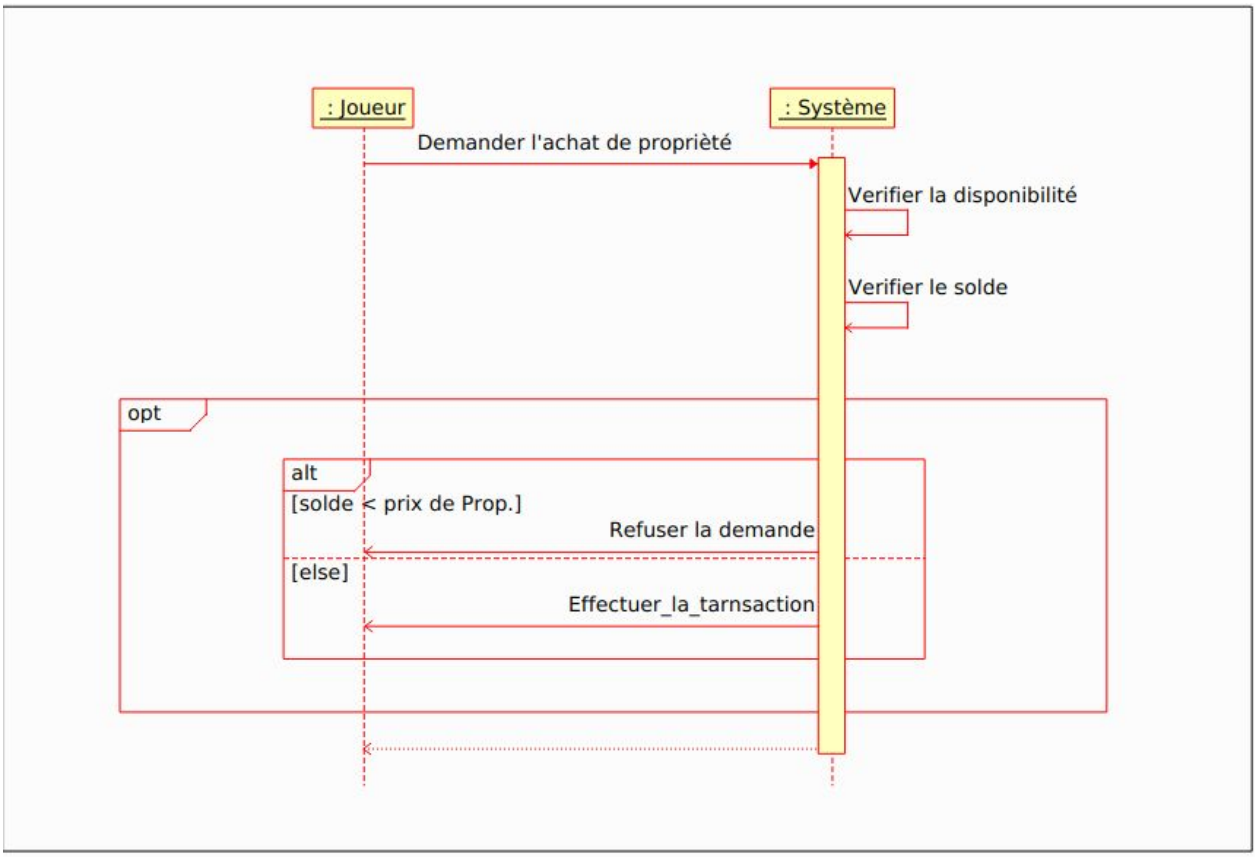
Le diagramme de séquence correspondant :



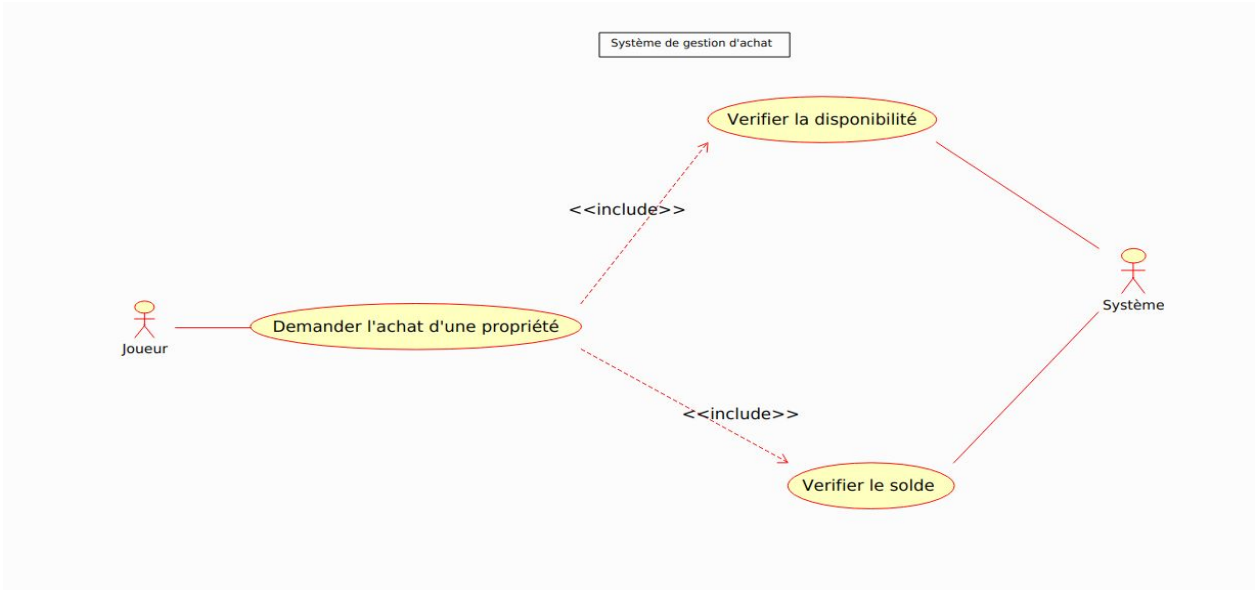
On peut détailler le mécanisme de l'achat d'une case propriété comme suit:

Le Joueur demande l'achat d'une propriété(au prix indiqué sur la case) ,ce dernier la disponibilité et vérifie si il peut payer, puis le joueur se déclare propriétaire et diminue son cash.

Le diagramme de séquence correspondant :



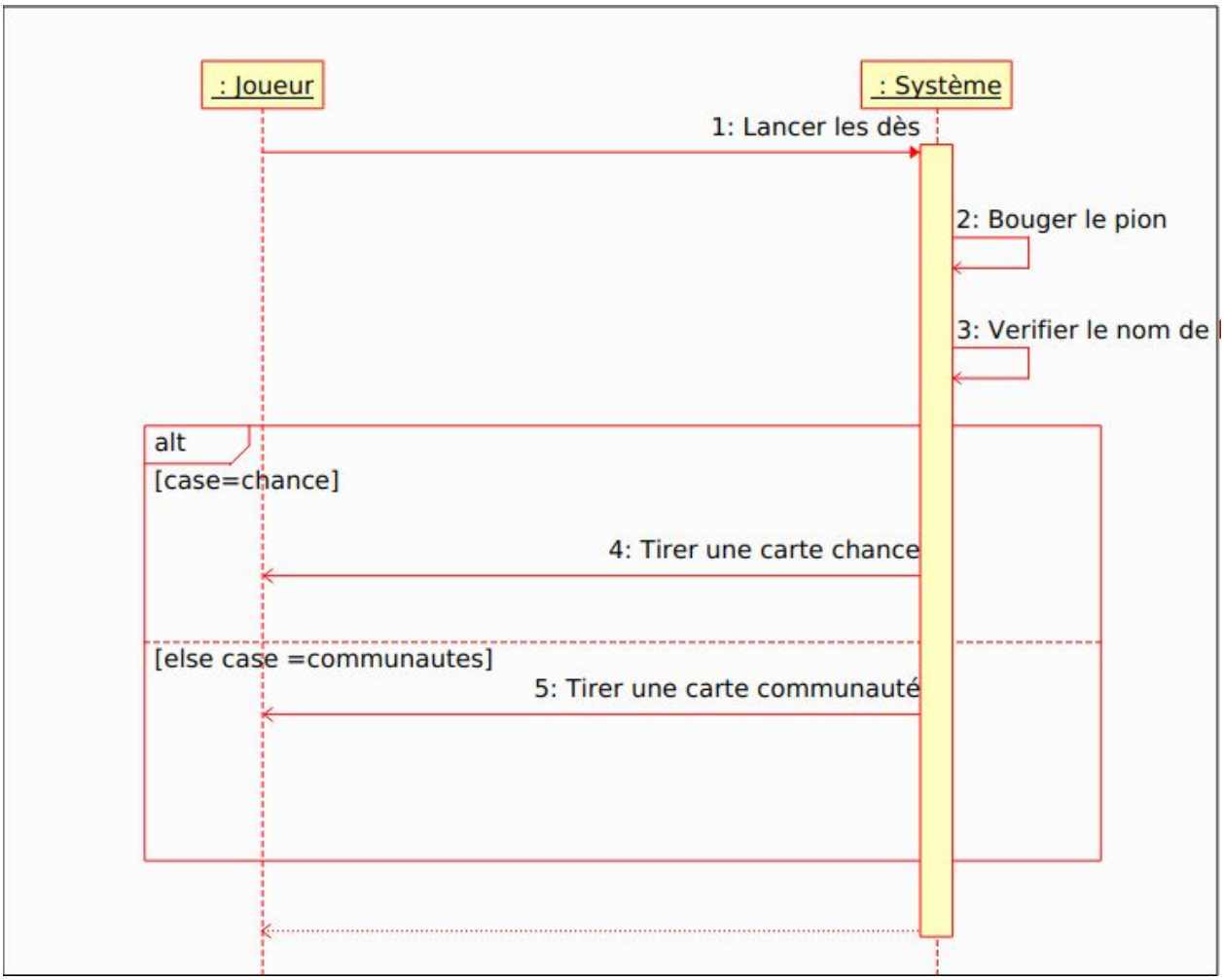
(son diagramme de cas d'utilisation) [voir le diagramme ci-dessus]



❖ **Diagramme de séquence pour l'opération "*tirer carte chance/communauté* " :**

Description:

Le joueur lance les dés, si le joueur se positionne sur une case,puis le système vérifie le type de la case; si la case est de type <<*chance*>> il va la récupérer et exécuter la consigne, de même pour les carte de <<*caisse communauté*>>.



Grâce au diagramme de séquence, nous avons vérifié la cohérence de notre solution. Le Plateau connaît les Joueurs, chaque Joueur connaît sa Case actuelle, chaque Joueur connaît ses Propriétés, et chaque Propriété connaît son Propriétaire, le joueur peut faire des transactions en se déplacement sur des propriétés,tirer des cartes(chance/communauté)...

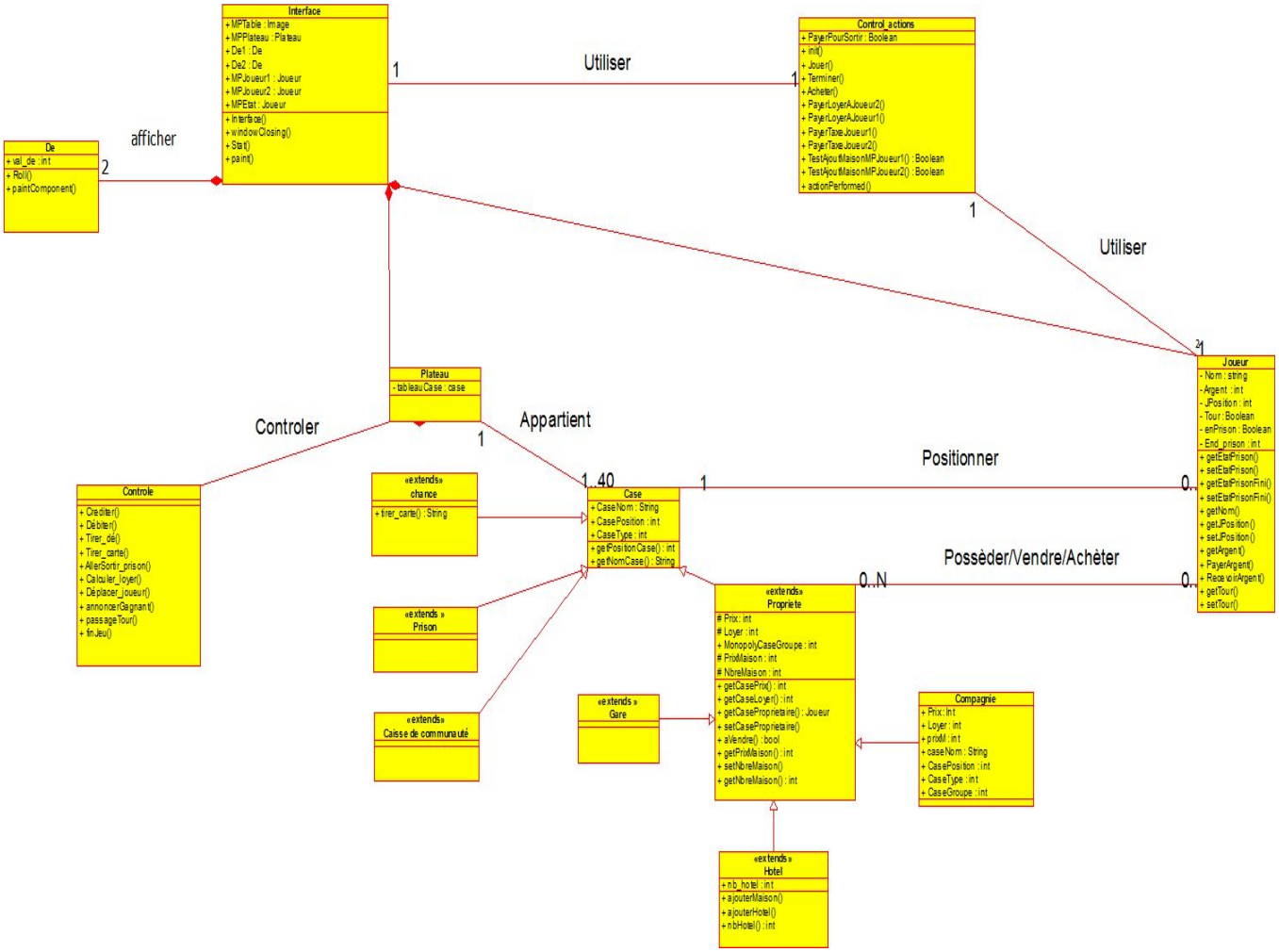
Diagramme de classes

Le diagramme de classes exprime, de manière générale, la structure statique d'un système, en termes de classes et de relations entre ces classes. De même qu'une classe décrit un ensemble d'objets, une association décrit un ensemble de liens. Les objets sont le résultat de l'instance de la classe, et les liens sont les instances des relations.

Nous allons consacrer cette partie du rapport pour la modélisation de diagrammes de classes, en utilisant toujours le logiciel **<< *Umbrello* >>** pour ensuite générer du code qui correspond à notre modélisation et nos besoins.

Plusieurs modifications ont été apporté à notre diagrammes de classes tout au long du réalisation du projet dû à quelques difficulté de programmation ou parfois à des ambiguïtés du fonctionnement du jeu.

Regardons de plus près notre diagramme :


Interface
+ MPTable : Image
+ MPPlateau : Plateau
+ De1 : De
+ De2 : De
+ MPJoueur1 : Joueur
+ MPJoueur2 : Joueur
+ MPEtat : Joueur
+ InterFace()
+ windowClosing()
+ Start()
+ paint()
Utiliser
1
1
Control_actions
+ PayerPourSortir : Boolean
+ init()
+ Jouer()
+ Terminer()
+ Acheter()
+ PayerLoyerAJoueur2()
+ PayerLoyerAJoueur1()
+ PayerTaxeJoueur1()
+ PayerTaxeJoueur2()
+ TestAjoutMaisonMPJoueur1() : Boolean
+ TestAjoutMaisonMPJoueur2() : Boolean
+ actionPerformed()
afficher
2
De
+ val_de : int
+ Roll()
+ paintComponent()
1
Utiliser
2
Joueur
- Nom : string
- Argent : int
- JPosition : int
- Tour : Boolean
- enPrison : Boolean
- End_prison : int
+ getEtatPrison()
+ setEtatPrison()
+ getEtatPrisonFini()
+ setEtatPrisonFini()
+ getNom()
+ getJPosition()
+ setJPosition()
+ getArgent()
+ PayerArgent()
+ RecevoirArgent()
+ getTour()
+ setTour()
Plateau
- tableauCase : case
Controler
1
Appartient
Positionner
1
0..
1..40
Controle
+ Crediter()
+ Debiter()
+ Tirer_dé()
+ Tirer_carte()
+ AllerSortir_prison()
+ Calculer_loyer()
+ Déplacer_joueur()
+ annoncerGagnant()
+ passageTour()
+ finJeu()
«extends»
chance
+ tirer_carte() : String
Case
+ CaseNom : String
+ CasePosition : int
+ CaseType : int
+ getPositionCase() : int
+ getNomCase() : String
Possèder/Vendre/Achèter
0..N
0..
«extends»
Propriete
Prix: int
Loyer : int
+ MonopolyCaseGroupe : int
# PrixMaison : int
# NbreMaison : int
+ getCasePrix() : int
+ getCaseLoyer() : int
+ getCaseProprietaire() : Joueur
+ setCaseProprietaire()
+ aVendre() : bool
+ getPrixMaison() : int
+ setNbreMaison()
+ getNbreMaison() : int
«extends »
Prison
«extends»
Caisse de communauté
«extends »
Gare
Compagnie
+ Prix: Int
+ Loyer: int
+ prixM : int
+ caseNom : String
+ CasePosition : int
+ CaseType : int
+ CaseGroupe : int
«extends»
Hotel
+ nb_hotel : int
+ ajouterMaison()
+ ajouterHotel()
+ nbHotel() : int

Les classes:

Joueur:

Attributs :

- ***Nom*** : nom du joueur(entré)
- ***Jposition*** : (position sur la grille de jeu, elle est représentée par un chiffre compris entre 1 et 40)
- ***Tour*** : 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.
- ***enPrison***: booléen permet de savoir si le joueur est en prison ou pas.
- ***End_Prison*** : permet de connaître le nombre de tour que le joueur a passé en prison (au bout de trois tours, il est libéré).
- ***Argent*** : la somme d'argent dont le joueur dispose.

Méthodes :

- *getNom()* : Renvoie le nom du joueur.
- *getPosition() / setPosition ()* : permet de changer la position du joueur sur la grille de jeu
- *getArgent()* : Renvoie le solde du joueur.
- *getEtatPrison() / setEtatPrison()* : permet de libérer ou d'emprisonner le joueur.
- *getEtatPrisonFini() / setEtatPrisonFini()*
- *getArgent()*
- *PayerArgent() / RecevoirArgent()*
- *getTour() / setTour()* : passer le tour

Dans notre diagramme :

➔ Un <<*joueur*>> *se positionne* sur une <<*case*>>, qui *appartient* à un <<*Plateau*>> cette dernière est contrôlé par la classe <<Control>> et également une composition de <<interface>>

- ➔ Un <<*joueur*>> *utilise* <<*Control_actions*>> où il y'a les actions que le joueur peut effectuer(acheter/ payer loyer…) cette classe qui utilise également <<*interface*>>
- ➔ Un <<joueur>> peut *posséder* une << propriété >> héritée de la classe <<case>>
- ➔ <<*gare*>> << *hotel*>> << *compagnie*>> <<*terrain*>> héritent de <<*propriété*>>
- ➔ <<*Joueur*>> << *Dés* >> << *Plateau* >> sont des compositions de << interface >>

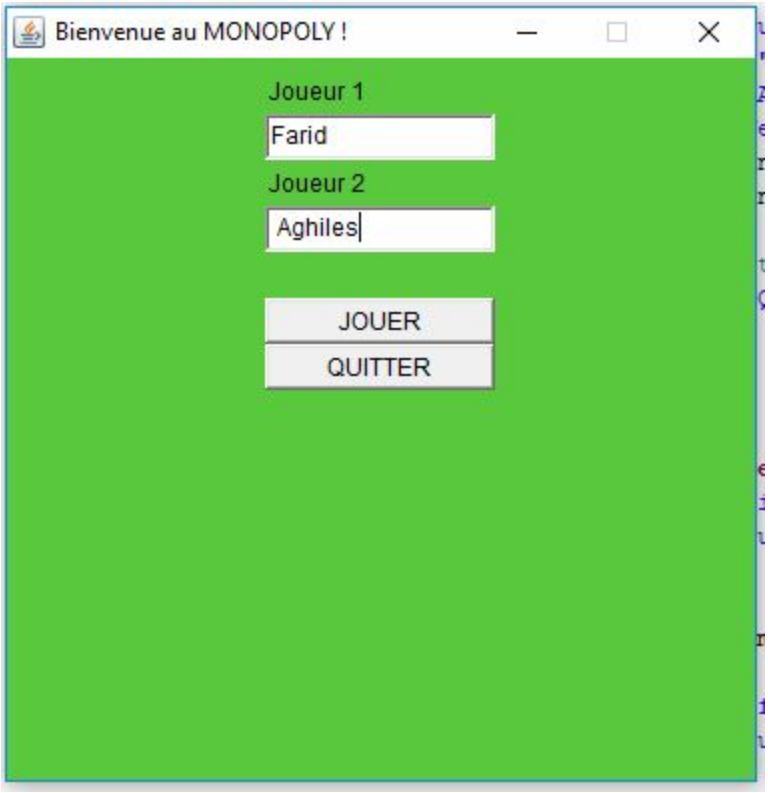
On a également penser à faire un petit diagramme d'état (concernant le joueur), voir le diagramme suivant:

<<Mettre le diagramme ici >>

Jouant une petite partie !

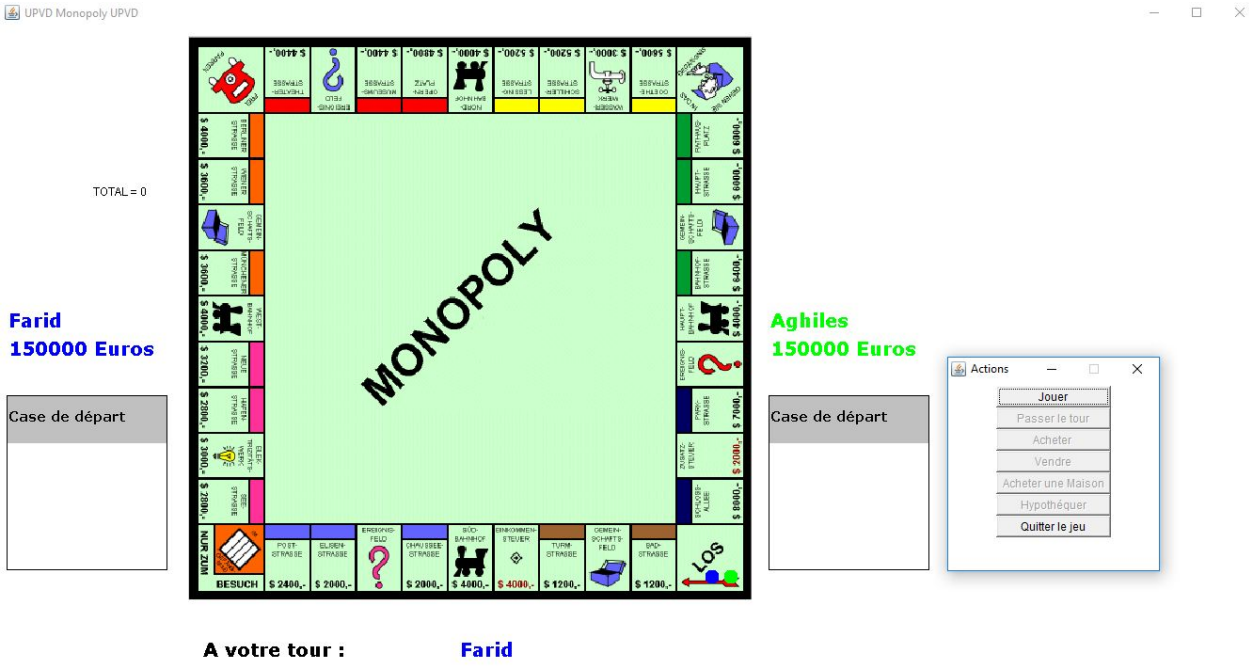
[En capture d'écran]

Le premier menu qui se lance à l'exécution du programme:



1. Les joueurs rentrent leurs nom
2. ils peuvent jouer une partie
3. ils peuvent quitter le jeu

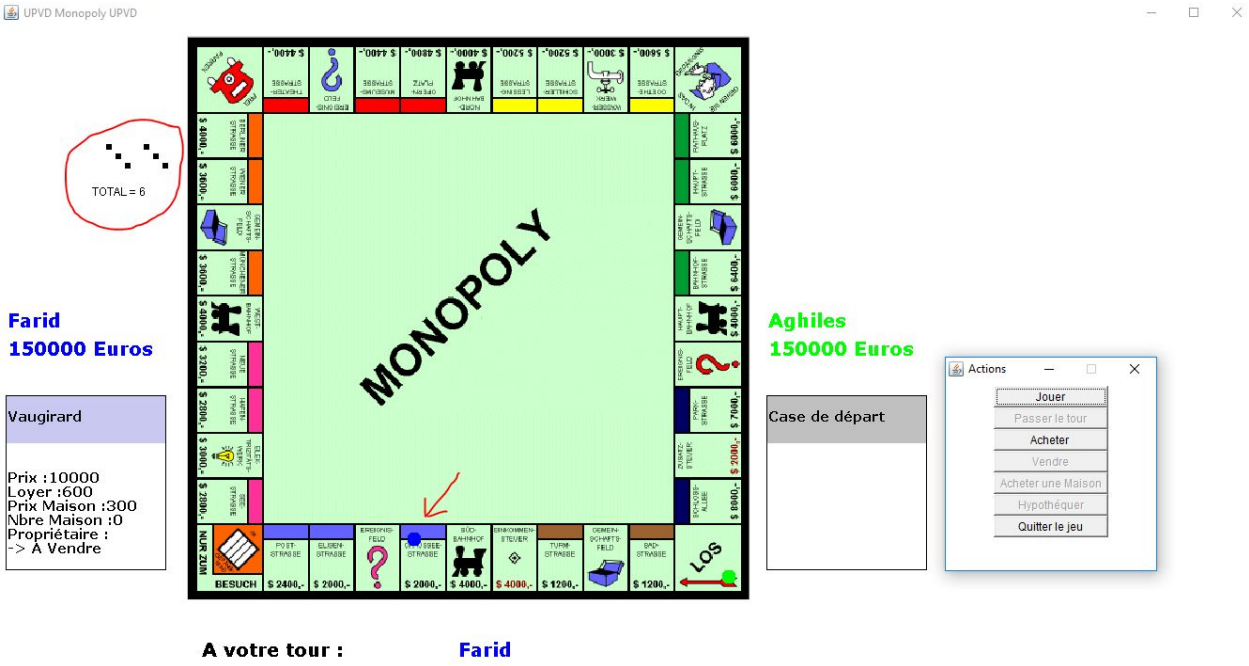
Si on clique sur jouer ...



L'interface de jeux affiche :

- Le plateau de jeu avec tous les types de cases.
- Le joueur qui va jouer
- la case actuelle. (initialement les joueurs sont sur la case de départ)
- l'interface affiche également le solde des deux joueur instantanément(au départ la banque leurs verse la somme de 150000 Euros.
- Le total des dés est affiché en haut à gauche (Total = au nombre de cases de déplacement)
- Un mini menu à droite des actions que le joueur peut effectuer (en **gras** : les actions possible à l'instant t, les autres actions s'affichent seulement si le joueur à la possibilité de faire l'action)

Le joueur clique sur jouer …



- Le joueur qui a lancé les dés (ça se fait aléatoirement par le logiciel) avance du nombre de total de lancement des dés.
- le joueur à=a avancé de 6 cases et tombe sur la case terrai qu'il peut acheter au prix affiché…
- puis le joueur passe le tour à l'autre joueur.
- A tout moment le joueur peut quitter la partie.

Conclusion

Ce projet bien que simple en apparence mais compliqué en réalisation car c'est un jeu qui est très riche d'actions en jouant mais également qui nécessite beaucoup de travail.Or il nous a permi parfois de rafraîchir nos compétence et parfois de les mettre en oeuvre,mais également d'apprendre plein de choses.
La réalisation de ce projet nous a permis d'appliquer ce qu'on a appris en cours de "Génie logiciel et projet de programmation" pour la modélisation des différent diagrammes (use case/ diagrammes de séquence/ diagramme de classes/d'états).
On a appris à diriger un projet depuis le début et de résoudre des problèmes de grandes tailles en les décomposant et mener de bout en bout la modélisation UML générale du logiciel .
Dans la partie logiciel on a appris aussi pas mal de choses en programmation Java comme la construction des interfaces graphiques et d'autres fonctionnalité de ce langage…