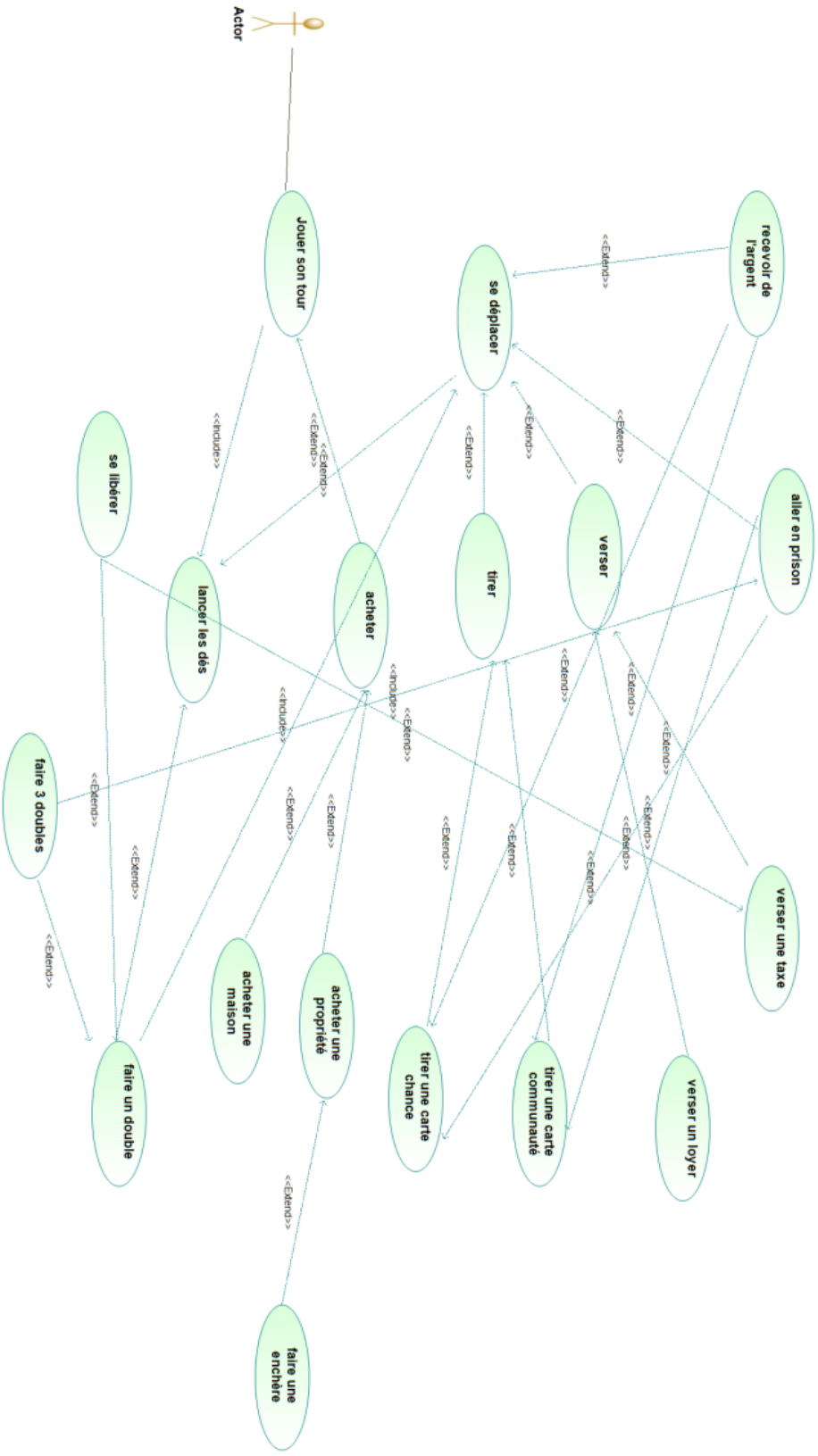
UML & Java

Analyse UML et implémentation du Monopoly

Johan KHA
Hugo MICHEL
Yanis PERRIN
François SOULIÉ

Capture des exigences fonctionnelles

Diagramme des cas d'utilisation



Fiche descriptive

Titre	Jouer son tour
Acteurs	Joueur
Description	Lorsque c'est à son tour de jouer, le joueur peut : ● Acheter/Améliorer une maison ● Hypothéquer ses propriétés ● Lancer les dés ● Terminer son tour
Scénario	1. C'est au tour du joueur A de jouer 2. Il lancer les dés 3. Il décide d'acheter la propriété correspondant à la case sur laquelle il a atterri 4. Il hypothèque une des ses propriétés 5. Il termine son tour
Pré-conditions	● L'utilisateur est inscrit ● L'utilisateur n'a pas fait faillite ● La propriété qu'il achète n'appartient à personne
Post-conditions	● La propriété a est assignée au joueur ● L'argent obtenu par la vente de l'autre propriété est ajouté au solde du joueur ● C'est au tour du prochain joueur de jouer

Titre	Aller en prison
Acteurs	Joueur
Description	Le joueur peut aller en prison si ● Il fait un double 3 fois d'affilé ● Il tire une carte "Aller en prison" ● Il tombe sur la case "Aller en prison" Et il peut être libéré aux prochains tours si ● Il fait un double ● Il possède une carte de libération ● Il est dans la prison depuis 3 tours (il paye un taxe dans ce cas là)
Scénario	1. Le joueur tombe sur la case "Aller en prison" 2. Le joueur n'a pas de chance et ne fait pas de double pendant 3 tours 3. Il paye la taxe pour sortir de prison

Post-conditions	Le joueur est libéré

Prototype des interfaces

Historique

Yanis (Bleu) Solde: 1 M
François (Rouge) Solde: 200k

10:30 - François lance les dés, il obtient 6
10:30 - François se déplace sur la case RUE DE LA PAIX
10:32 - François refuse d'acheter RUE DE LA PAIX
10:32 - Fin du tour de François

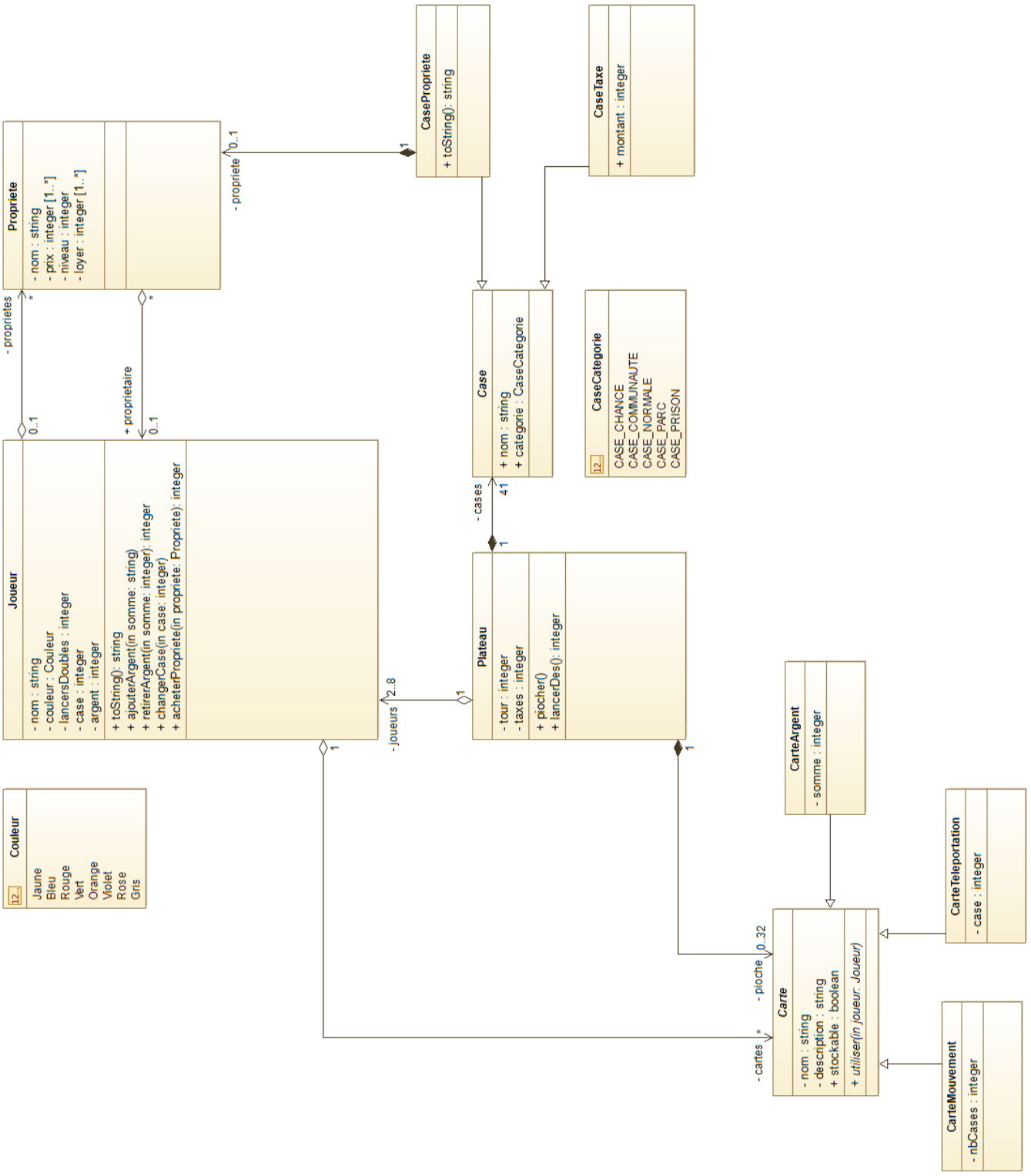
10:33 - Yanis lance les dés, il obtient 3
10:33 - Yanis se déplace sur la case AVENUE DES CHAMPS ÉLYSÉES

C'est le tour de Yanis
Souhaitez-vous acheter la propriété AVENUE DES CHAMPS ÉLYSÉES pour 10 000 Francs ? (Y/N)

Envoyer une commande... utiliser *help* pour voir les commandes disponibles

Étude du modèle statique

Première ébauche du diagramme de classe



Dictionnaire de données

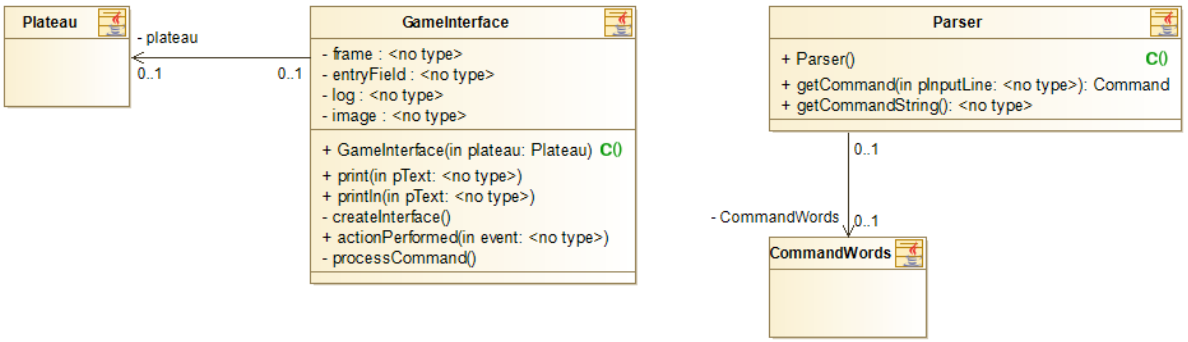
Nom	**Description**
Plateau	Le plateau, contrôle le déroulement du jeu (changements de tour, lancer des dés, etc.) et contient les cases et la pile d'argent
Joueur	Le joueur, contient les informations le concernant tel que sa case actuelle, son solde, ses propriétés, etc. ainsi que les méthodes permettent de modifier ces informations
Propriete	Une propriété, contient son nom et ses prix/son loyer selon son niveau.
Case	Une case du plateau, contient son nom et sa catégorie provenant de *CaseCategorie*
CaseCategorie	Enum contenant les différentes catégories de case
CaseTaxe	Une case du plateau, oblige un joueur à payer une taxe lorsque ce dernier tombe dessus
CasePropriete	Une case du plateau, contient une propriété
Couleur	Contient toutes les couleurs disponibles (max 8 couleurs)
Carte	Une carte, pouvant être une carte communauté, une carte chance ou une carte propriété
CarteMouvement	Contient le nombre de cases que le joueur doit parcourir
CarteTeleporation	Carte contenant le numéro de la case où téléporter le joueur
CarteArgent	Carte contenant la somme d'argent à ajouter au solde du joueur l'ayant tiré

Diagrammes de classe finaux

Nous avons décidé de découper l'application en 3 parties principales : Model, View et Controller.

1. Model correspond essentiellement au diagramme réalisé lors de la première ébauche, il contient les classes correspondant aux données qui seront manipulées.
2. View contient les classes correspondant à l'interface graphique
3. Controller contient les classes qui feront le lien entre View et Model
4. Commands contient les classes liées aux commandes

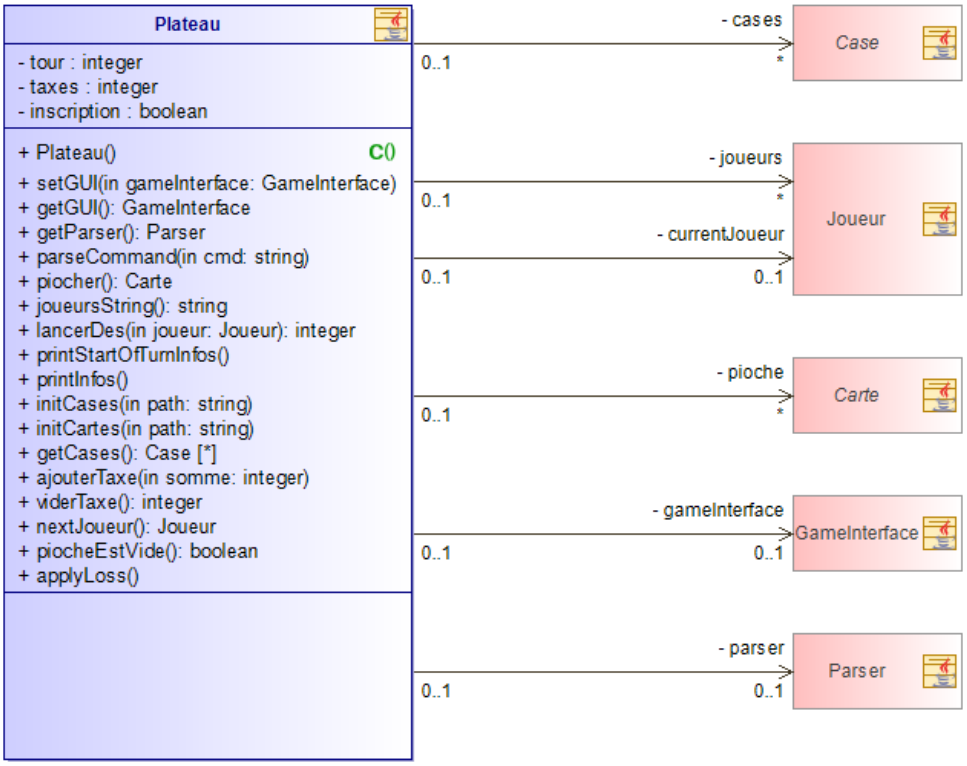
Diagramme de classe View



Dictionnaire de données de View

Nom	**Description**
GameInterface	L'interface du jeu, contient l'initialisation de la fenêtre, des boutons, … et les interactions possibles
Parser	La classe chargée d'interpréter les commandes entrées par l'utilisateur

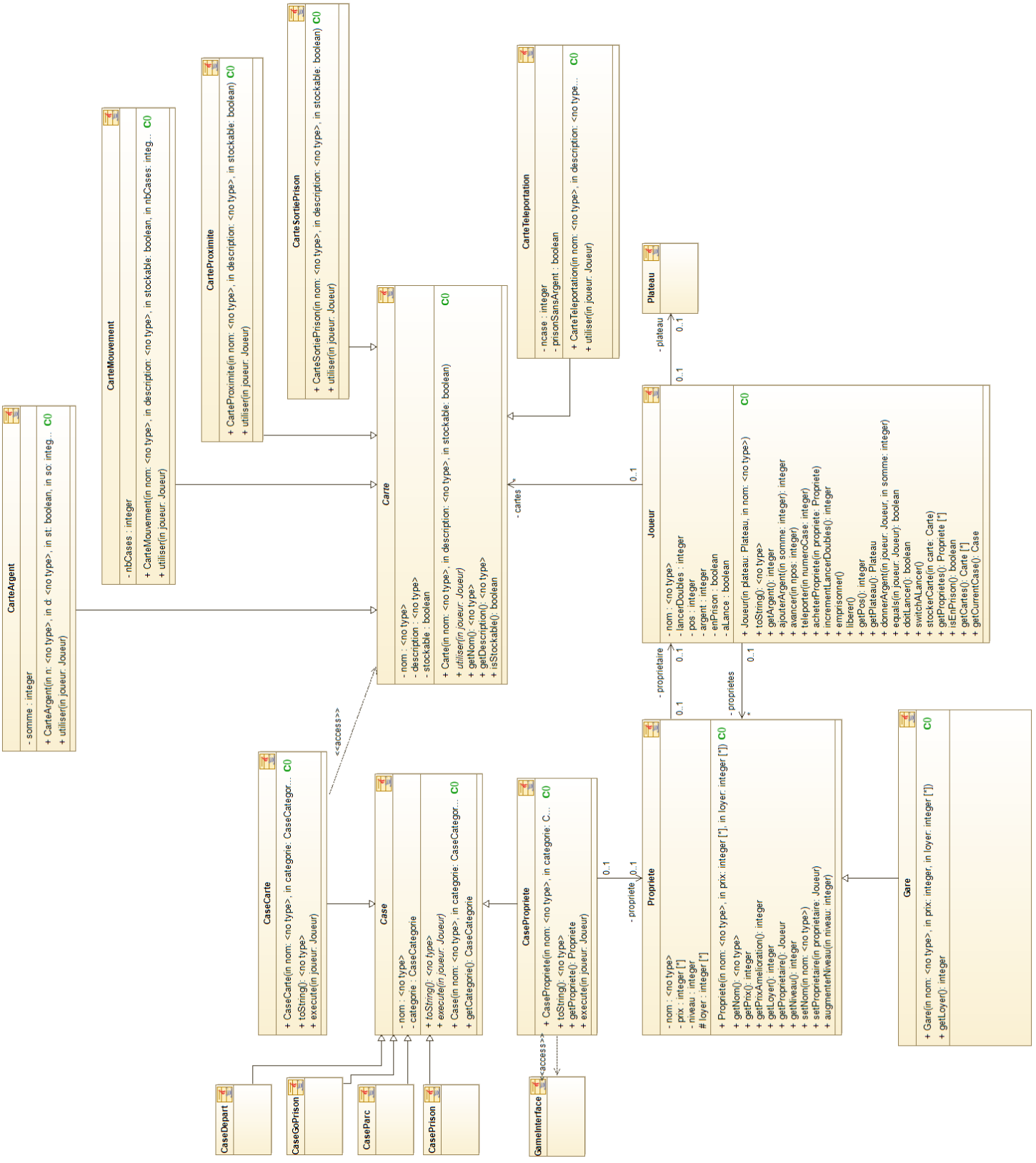
Diagramme de classe Controller



Dictionnaire de données de Controller

Nom	**Description**
Plateau	La classe chargée de faire l'interface entre l'interface et les données. Elle sert en gros de moteur de jeu.

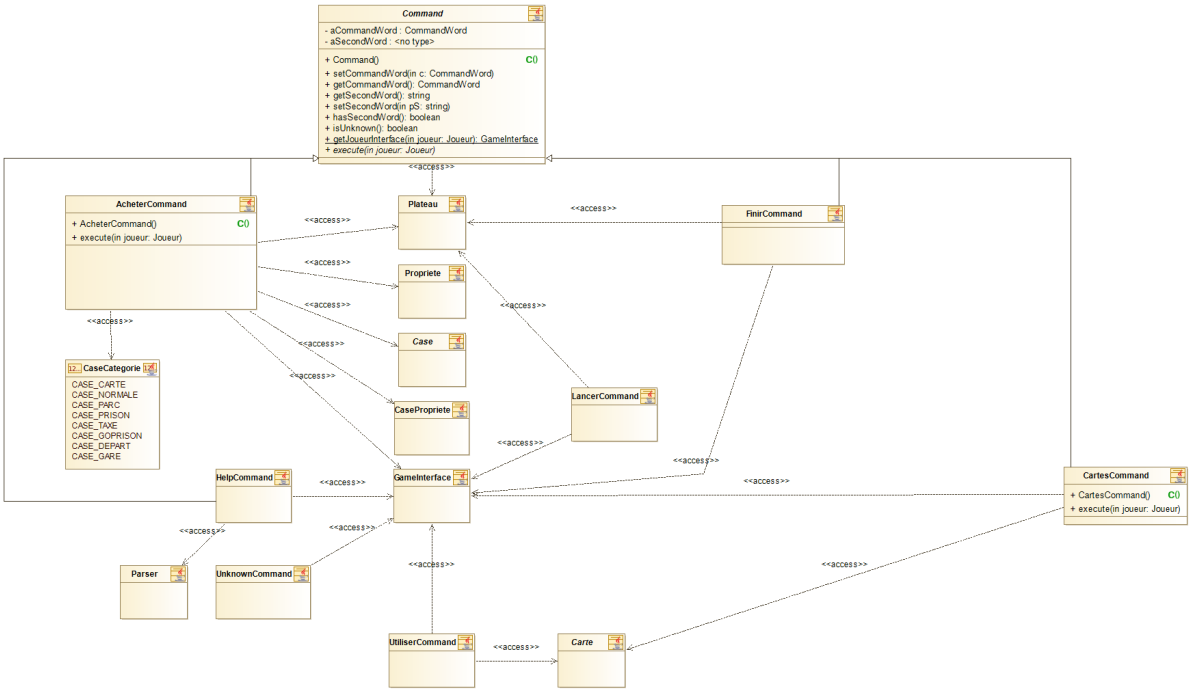
Diagramme de classe Model



Dictionnaire de données de Model

Nom	**Description**
Joueur	Le joueur, contient les informations le concernant tel que sa case actuelle, son solde, ses propriétés, etc.
Propriete	Une propriété, contient son nom et ses prix/son loyer selon son niveau.
Case	Une case du plateau, contient son nom et sa catégorie provenant de *CaseCategorie*
CaseCategorie	Enum contenant les différentes catégories de case
CaseTaxe	Une case du plateau, oblige un joueur à payer une taxe lorsque ce dernier tombe dessus
CasePropriete	Une case du plateau, contient une propriété
CaseParc	Une case du plateau permettant au joueur de récolter les taxes déposées par les autres joueurs
CasePrison	Une case ne contenant aucune propriété correspondant à la visite de la prison
CaseGoPrison	La case “Aller en prison” faisant aller le joueur en prison
Carte	Une carte, pouvant être une carte communauté, une carte chance ou une carte propriété
CarteMouvement	Contient le nombre de cases que le joueur doit parcourir
CarteTeleporation	Carte contenant le numéro de la case où téléporter le joueur
CarteArgent	Carte contenant la somme d’argent à ajouter au solde du joueur l’ayant tiré
CarteSortiePrison	Permet au joueur de sortir de prison

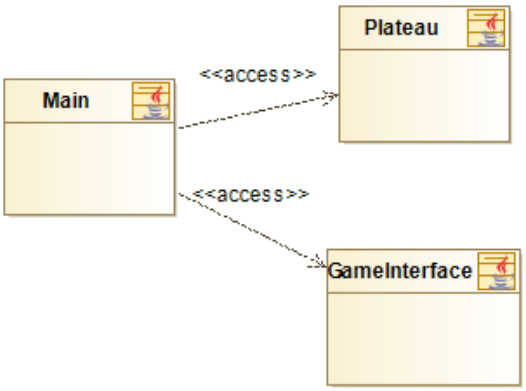
Diagramme de classe Commands



Dictionnaire de données de Commands

Nom	**Description**
Command	Classe abstraite correspondant à une commande du jeu, pouvant être exécutée pour appliquer des changements au joueur et/ou au plateau

Diagramme de classe Main



Étude du modèle dynamique

Diagramme de séquence de l'initialisation du jeu dans Main

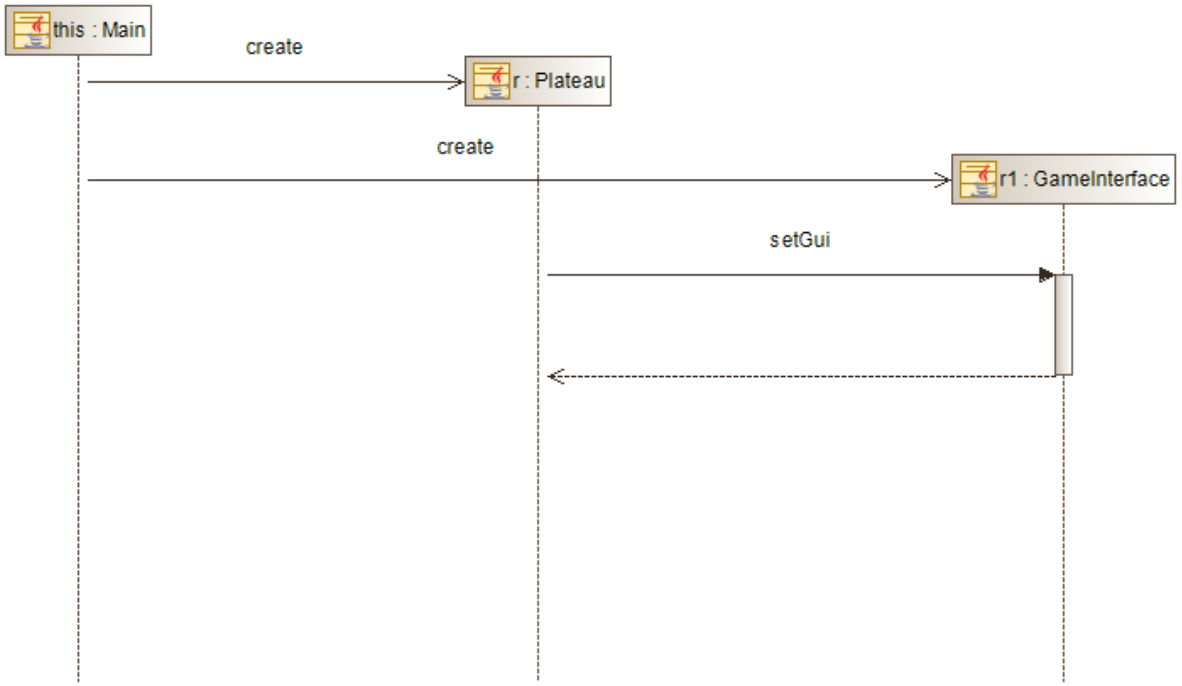
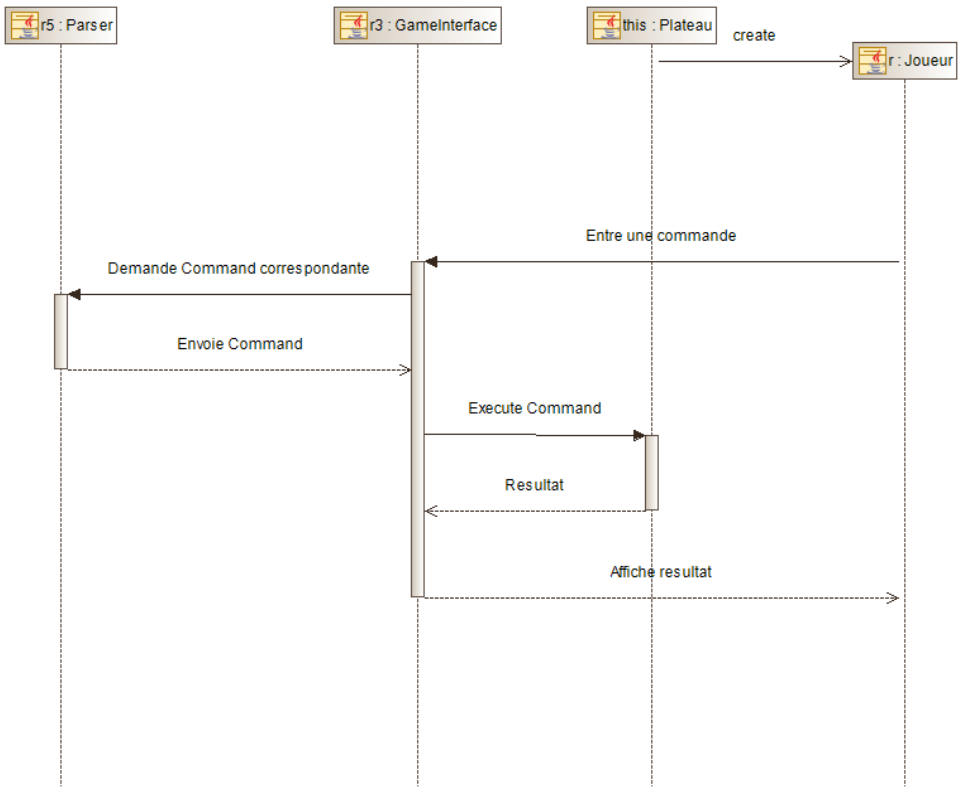
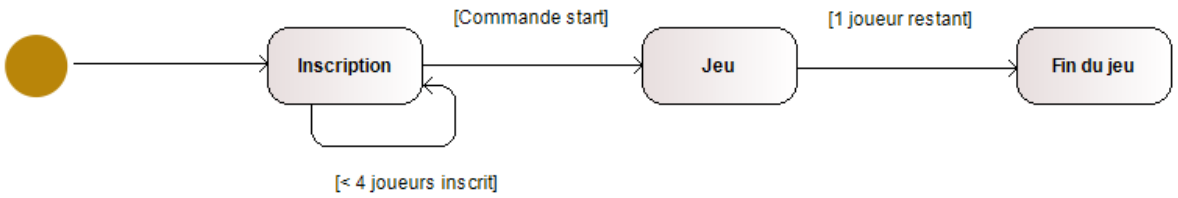


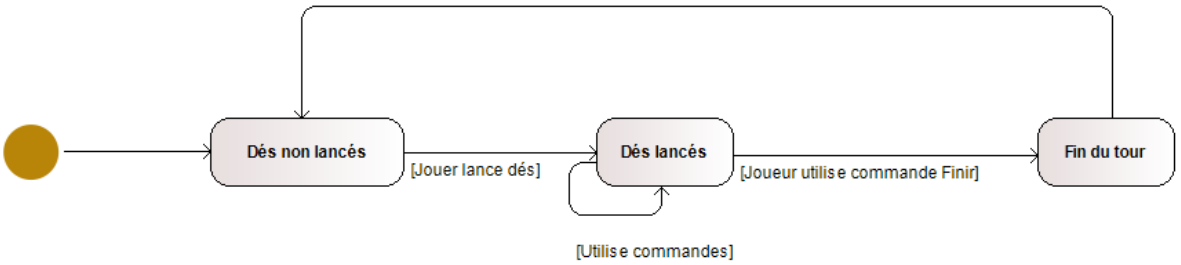
Diagramme de séquence de l'envoie d'une commande



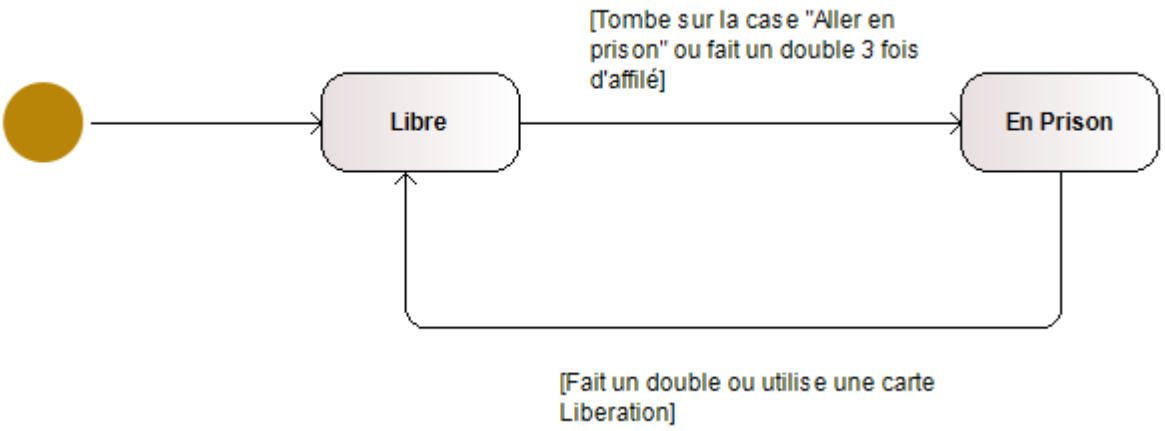
Graphe d’état du déroulement du jeu



Graphe d’état du déroulement d’un tour



Graphe d’état du joueur



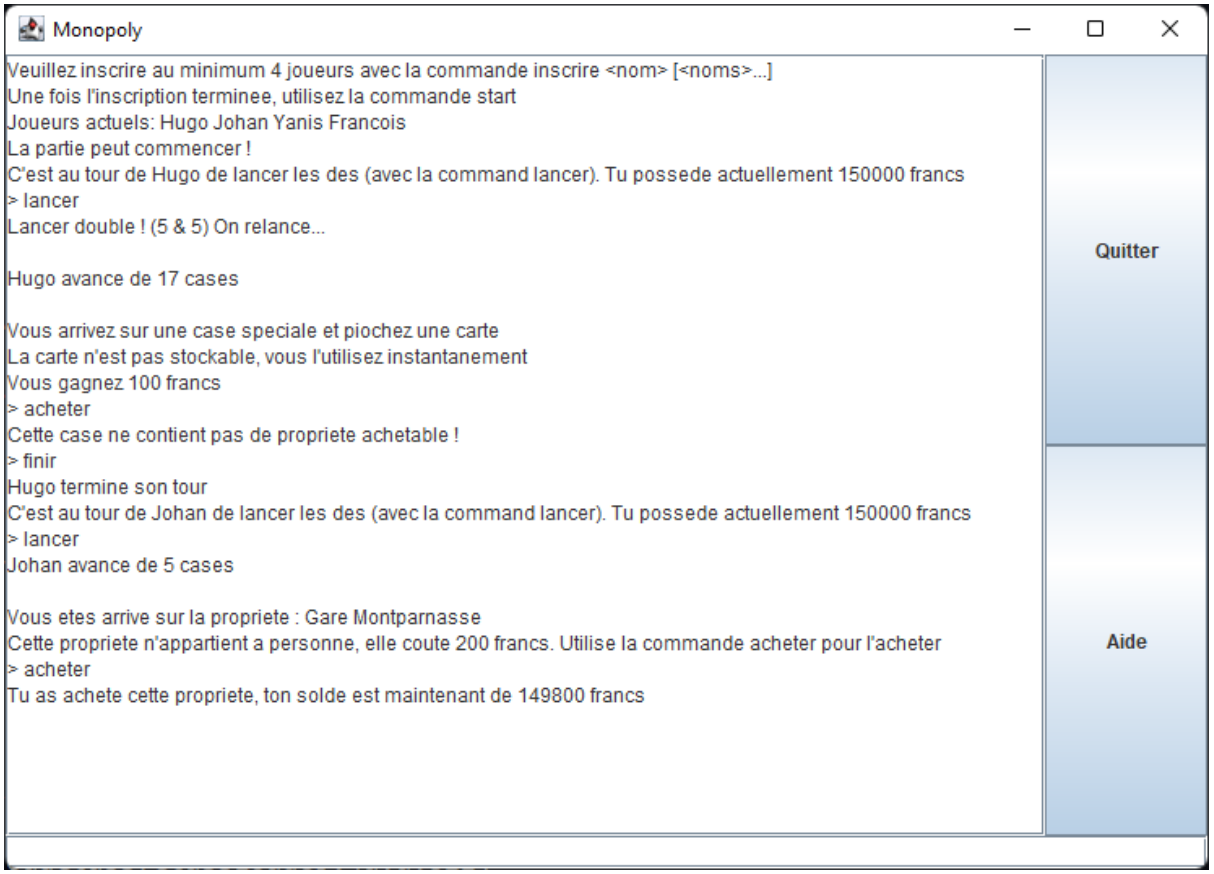
Phase d'implémentation

L'ensemble des cas d'utilisation décrit ont été implémentés lors de la phase d'implémentation

Justifications technologiques

- Nous avons utilisé swing pour l'interface car nous étions déjà familiers avec, et c'est une bibliothèque plutôt simple à utiliser
- Nous avons décidé d'écrire les informations des cases et des cartes dans des fichiers csv ensuite importés car cela nous semblait plus propre puisque cela crée une séparation entre les données et le code.

Copies d'écran



Livrable

L’ensemble du code est disponible sur github : https://github.com/francois07/monopoly

Bilan de l’état du projet

1. Actuellement, puisque nous sommes plusieurs à avoir travaillé sur cette implémentation, certains aspects de ce dernier ne sont pas homogènes c'est-à-dire que nous avons des manières différentes de nommer les variables, de prendre en compte les erreurs, etc.
2. L'expérience utilisateur est pratiquement inexistante. Il pourrait être intéressant d’étudier ce qui ne va pas à ce niveau là et d’essayer de l’améliorer avec une approche Design Thinking mais cela irait au delà du cadre de l’unité
3. Le jeu est purement textuel, une implémentation graphique serait elle aussi intéressante