PROJET ALGORITHMIQUE ET PROGRAMMATION ORIENTEE OBJET

Le Monopoly – Rapport de Projet

Groupe 10/12

Année Scolaire 2021 – 2022

MU Maxime, FORRAY Léo-Paul, BELKHITER Yannis & SHCHERBAKOV Vadim

TABLE DES MATIÈRES

I. Synthese du Projet

I. SYNTHESE DU PROJET

1. PRESENTATION DU PROJET

A. *L'Histoire*

La longue épopée du Monopoly débute aux États-Unis en 1929, alors que sévit la terrible crise économique. Charles Darrow, un chauffagiste de Pennsylvanie, au chômage comme des millions de personnes s'inspire du rêve de tout américain' spéculation, fortune, prospérité... et invente le Monopoly. Très vite, l'engouement pour le Monopoly se développe où delà de son cercle familial. Avec l'aide d'un petit imprimeur, Darrow produit six jeux par jour, mais est bientôt dans l'incapacité de faire face à la demande.
En 1934, il prend donc contact avec Parker Brothers, l'un des plus célèbres fabricants de jouets américains. Malheureusement, son jeu est refusé cor il contient " 52 erreurs fondamentales ". Darrow ne s'avoue pas vaincu et vend, celte même année, 4 000 Monopoly.
Devant l'ampleur du succès, Parker Brothers reconsidère sa position et acquiert les droits du jeu. Nous sommes en 1935. Depuis cette date, les ventes n'ont cessé de s'accroître. Traduit en 26 langues et diffusé dans 43 pays, Monopoly est le plus célèbre jeu de société au monde.

- Tirée de la règle du jeu du Monopoly -

B. *Définition du jeu*

Le Monopoly est un jeu de plateau dont le but consiste à ruiner ses concurrents par des opérations immobilières. Il symbolise les aspects apparents et spectaculaires du capitalisme, les fortunes se faisant et se défaisant au fil des coups de dés.
Les parties se joue de 2 à 6 joueurs, et durent de 1 à 5 heures.

Les règles du jeu sont énoncées dans un document à part : « Règle du jeu »

Le but de notre équipe à donc été de dématérialiser ce fameux jeu de société, et de créer une version en ligne jouable part différentes personnes sur un même ordinateur.
Dès lors, c'est un projet assez conséquent puisqu'il a fallut prendre en compte toutes les options de jeu et les actions possibles par les différents joueurs. Il a fallu également créer une interface graphique pour rendre le jeu interactif et plus plaisant à jouer.

C. *Les raisons de ce choix*

Nous avons choisi ce projet car on voulait expérimenter le travail nécessaire lors du développement d'un jeu en groupe. En effet, nous voulions choisir un projet qui nécessitais le partage des tâches notamment sur la partie interface graphique et partie interne du code. C'est une démarche qui se rapproche le plus à la conception et au développement de jeux dans le monde professionnel, et c'est pour cette raison que nous avons été séduits par ce projet en particulier.

2. REPARTISSIONS DES TACHES

Dans un premier temps, nous avons dû définir à la fois des objectifs, et des tâches à réaliser pour mener à bien ce projet. Après discussions, et analyse des choses à réaliser, nous avons opter de séparer le groupe en deux équipes de 2 personnes. Une équipe s'est occupée de la partie structure interne du projet, et l'autre équipe s'est occupée de la partie interface graphique. De fait, chaque équipe s'est vu attribuer des tâches à réaliser, que nous avons réparties entre chaque membre des sous-équipes. L'avancement du projet s'est alors fait de manière structurée, de la même façon qu'une entreprise le ferais pour la réalisation d'un projet.

Au fur et à mesure des jours, nous avons ainsi avancé entre chaque équipe de façon indépendante, en nous mettant d'accord sur la structure interne à concevoir au début. En effet, l'équipe interface graphique a utilisé les classes et méthodes conçu par l'autre équipe.
A la fin du projet, il a fallu mettre en commun le travail des deux équipes pour faire tourner le code de façon correcte et pour utiliser ce que chacune des équipes a réalisé. Ce fût la partie la plus délicate puisque nous avons dû faire communiquer ces deux parties, c'est pour cette raison que nous avions commencé par faire un diagramme de classe provisoire en commun afin d'assurer la bonne communication entre les deux parties.

A. *L'interface graphique*

Membres de l'équipe : MU Maxime & BELKHITER Yannis

Pour la partie interface graphique, nous avons réalisé une interface avec laquelle le jouer peut interagir en fonction de son argent, de ses acquisitions et de la position de son pion, cette dernière étant dictée par le résultat obtenu lors du lancer de dés par le joueur en question. Cette partie prenait également en compte la réalisation du design des cartes, du plateau, des pions, de leurs déplacements, et des animations liées aux décisions des différents joueurs.
Nous nous sommes occupés dans un premier temps du plateau de jeu. Nous avons délimité les cases sur lesquels le pion se déplace, et nous avons mis en place un affichage graphique des paramètres de jouabilité (argent, acquisition, position, etc…)
Ensuite, nous avons créé des interactions graphiques avec les joueurs. En effet, nous avons créé des animations pour chacune des cases du plateau (Cf partie Analyse).
Nous avons ajouté aussi des dés interactifs qui simulent le lancer aléatoire de deux dés, et deux types de cartes : chance et caisse de communauté.

B. *Structure interne*

Membres de l'équipe : FORRAY Léo-Paul & SHCHERBAKOV Vadim

Pour ce qui est de la partie interne du code, elle s'est faite à partir des règles du jeu de Monopoly. Ainsi nous avons codé les cases, les cartes, le plateau, les dés et leurs interactions avec les joueurs. Finalement nous avons une interface graphique avec laquelle nous pouvons interagir et qui envoie les différents paramètres aux méthodes de la partie interne assurant donc le bon déroulement d'une partie de Monopoly classique et tout ça sur un seul terminal. Nous pouvons donc au lancement du programme fixer un nombre de joueur, l'ordre de passage par un lancement de dés et démarrer la partie.

II. Analyse

II. ANALYSE

La résolution du problème passe en grande partie par la séparation des tâches entre l'interface graphique et la partie interne du code.

1. STRUCTURE INTERNE

Concernant la partie interne, le premier problème consistait à trouver les bonnes classes à mettre en œuvre, ainsi que les relations entre elles. Nous avons initialement dégagé les classes suivantes :

- Case : Cette classe sert de classe mère à toutes les cases du type Case Prison, Case Chance etc… Ces cases ont été codée au fur et à mesure et elles implémentent les méthodes de la classe Case. Par ailleurs nous avons constaté que les cases vont effectuer des actions en fonction du Joueur se trouvant sur cette dernière. C'est pour cela que nous avons décidé d'effectuer une relation d'agrégation entre la classe Case et la classe Joueur, dans le but d'avoir accès aux méthodes qui permettent de retirer ou ajouter de l'argent au joueur.
- Joueur : Cette classe sert à contenir l'argent du joueur, ses propriétés, les couleurs possédés, etc... Les couleurs possédés nous permettent de savoir si le joueur possède toutes les cases d'une même couleur et donc on peut ainsi savoir s'il faut compter un loyer plus cher ou non.
- Carte : La classe carte est une classe mère qui va être l'ancêtre de Carte Prison, Carte Payer Argent, Carte Déplacement, Carte Recevoir Argent. Nous avons décidé de regrouper les cartes en fonction de l'action qu'elles vont exécuter.
- Plateau : Enfin nous avons compris la nécessité d'une classe plateau qui permettrait de regrouper toutes les cases, les cartes, les informations sur les joueurs ainsi que des attributs qui permettent de compter le nombre de tours et de connaitre la valeur obtenue lors du jet de dés. Nous avons donc dû réaliser des relations d'agrégation entre la classe plateau et les autres cités précédemment et la classe Dés.
- Dés : Cette classe nous sert qu'à obtenir deux entiers aléatoirement entre 1 et 6.

Concernant le déroulement du jeu, Léo-Paul Foray s'est occupé de la création des classes nécessaires au fonctionnement du jeu. Voici ce qu'il en pense :

Mon apport à ce projet a été l'ensemble des classes des packages plateauMopoly, Famille Terrain, Cases, Donnees. J'avais pour mission de créer le jeu en lui-même et de ne pas toucher l'interface graphique, avec Vadim Shcherbakov. Lui et moi nous avons séparé les taches, il a codé les cartes, moi les cases, et nous avons travaillé ensemble sur le reste.

Pour ma part, j'ai commencé part créer mes classes en les liants entre elles, a la manière d'un diagramme de classe en leur créant des attributs et méthodes, mais je ne donnais que le nom aux méthodes pour l'instant. Apres avoir créé le squelette de mon programme, j'ai commencé par remplir la classe main(). Ma progression devait se faire telle celle d'une partie, j'ai donc commencé par l'ajout de joueurs, sur ma partie, on pouvait jouer de 2 à 6 joueurs. Pour chaque joueur, je demande un nom, puis j'initialise un plateau. Ensuite, j'ai créé la boucle while qui permet de faire le système de tours, avec alternance de joueur selon une liste. J'ai donc par la suite codé la classe joueur et sa méthode jouer() ainsi que toutes ses méthodes auxiliaires telles que payer() ou hypotequer(), car la boucle précédente y fait appel. Une fois ceci fait, j'ai dû ajouter sur chaque famille de terrain les méthodes ajouterMaison(), achetee(), actionCase(), qui sont indispensables au bon fonctionnement du jeu.

Grâce à ceci, on a pu créer une ébauche du jeu, mais il manquait encore beaucoup de fonctionnalités. Par exemple, trouver un moyen de classer les joueurs selon un lancer de dés m'a pris pas mal de temps. L'autre complexité fut de rationnaliser l'ajout des maisons, car cela ne peut pas se faire n'importe comment. J'ai pu régler la quasi-totalité des problèmes sur ma partie. Ceux qui restent sont l'impossibilité de choisir le nombre de maisons à hypothéquer, de plus après hypothèque, il n'y a pas équilibrage des maisons sur une famille de terrain, créant un déséquilibre interdit par les règles. L'absence de la fonctionnalité d'échange et de revente des terrains entre joueurs est aussi à déplorer.

2. INTERFACE GRAPHIQUE

Dans cette partie, le but était de créer un lien entre la partie et le joueur. Plusieurs problèmes se sont alors présentés à nous : Comment rendre le plateau de jeu dynamique ? Comment créer une interaction avec le joueur ? Comment implémenter des fonctionnalités jouables sur le plateau de jeu ?

Nous avons dans un premier temps commencer à réaliser la délimitation du plateau de jeu. En partant d'une image, nous avons délimité chacune des cases pour faire interagir le pion avec le type de case.

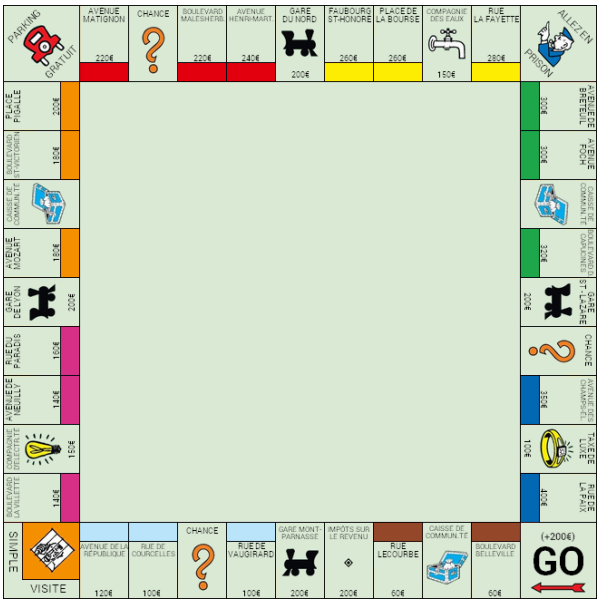


Figure 1 : *Plateau de jeu vierge*

Maxime s'est occupé de cette partie-là. Ce fût un gros travail puisqu'il a fallu référencer chaque type de case, et plus de 40 emplacements.
Ensuite, il a également été nécessaire d'afficher les informations des joueurs en temps réel sur le plateau.

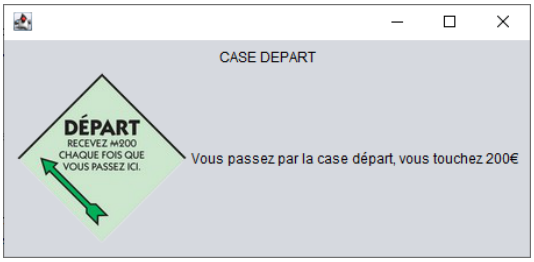
Figure 2 : *Plateau de jeu final*

Il a également été nécessaire de créer une animation pour le lancer des dés, pour la pioche de cartes chances / caisse de communauté, et pour passer son tour. Tout cela a également été réalisé par Maxime.

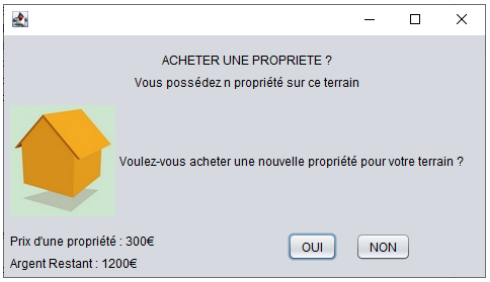
Concernant l'interaction du joueur avec le plateau de jeu, il a fallu créer des animations lorsqu'un pion tombe sur une case. En effet, à chaque case, nous avons créé une interface graphique d'action comme « Acheter une propriété », « Payer un loyer », ou encore « Tirer une carte chance ». Cette partie a été réalisé par Yannis.

Acheter_terrain() : Acheter un terrain en cas de non appartenance à un autre joueur.

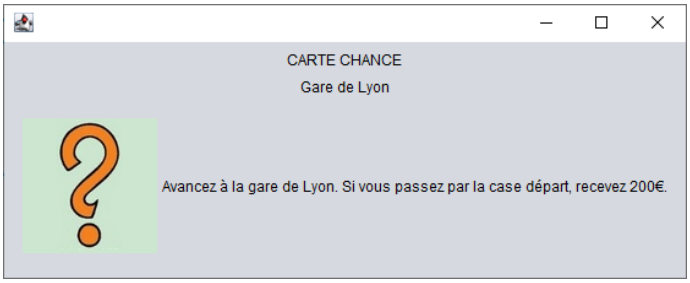
Case Départ : Toucher une prime en passant par la case départ



Acheter_prop() : Placer une maison sur un terrain possédé

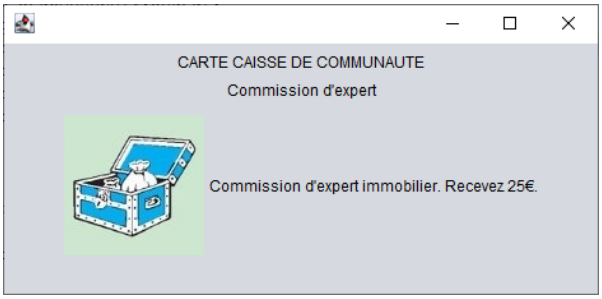


Carte Chance : Tirer une carte au dessus de la pile



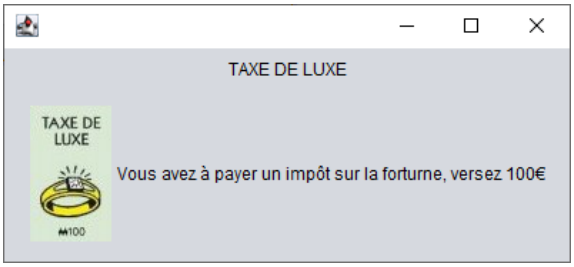
Payer_Loyer() : Payer un Loyer à un autre joueur

Carte Caisse de Communauté : Tirer une carte au dessus de la pile



Compagnie de distributions

Taxe de Luxe



III. DIAGRAMME DE CLASSES

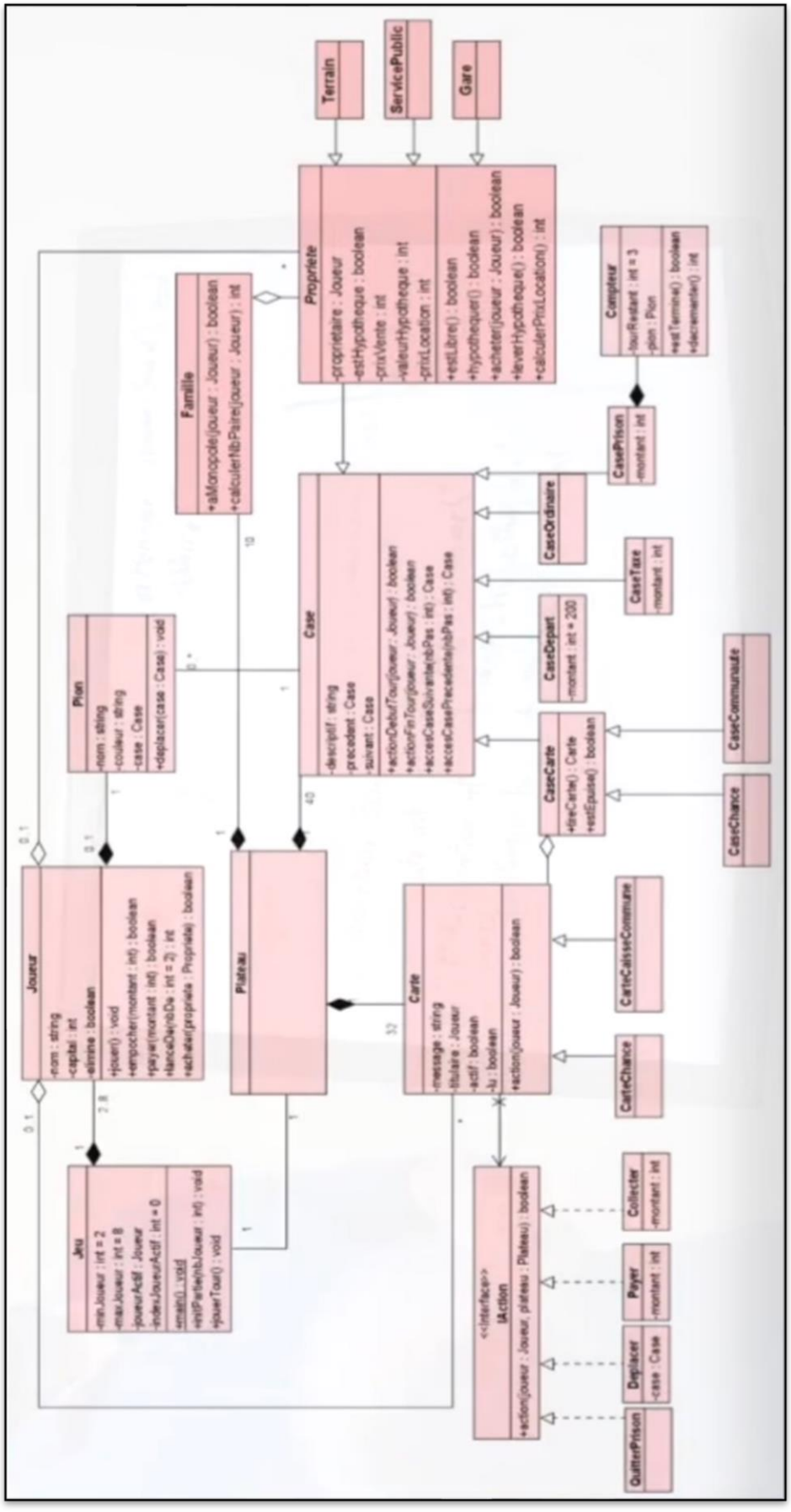


Figure 3 : *Diagramme de Classe*

IV. MISE EN EVIDENCE DES ELEMENTS LES PLUS IMPORTANTS

1. ANALYSE D'UN ENCHAINEMENT

Il est impossible d'analyser entièrement notre long code… Néanmoins, pour porter une analyse sur notre code, nous allons décrire les processus d'échanges d'information entre les différentes classes/interfaces pour mieux comprendre la structure de notre code.
De fait, nous allons maintenant vous décrire l'algorithme qui s'effectue lors de ***l'achat d'une case*** :

Nous nous situons donc dans le cas où le joueur a jeté ses dés et s'est retrouvé sur une case.
Nous avons une première méthode de la classe plateau qui va se lancer et qui est la suivante :

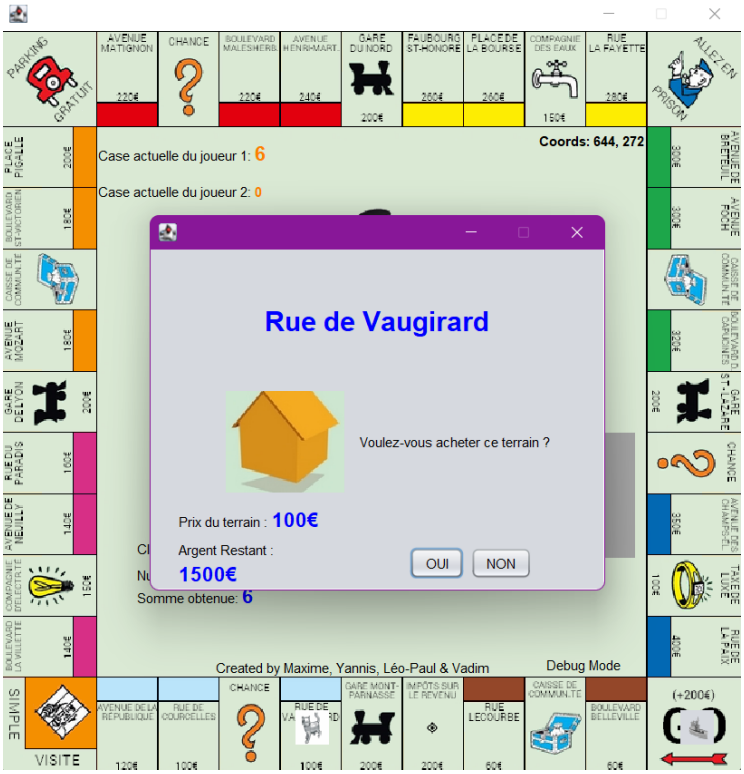


Figure 4 : *Le joueur actuel vient de tomber sur la rue de Vaugirard, il a la possibilité de l'acheter*

Soit p le plateau et j le joueur qui joue son tour.

```
case 6:
    {
    if(p.verif_appartenance_case_vide(j)){
        Acheter_Terrain Ach_terrain = new Acheter_Terrain(p, j);
        break;}
    else{
        Payer_Chez_Qlqun payer = new Payer_Chez_Qlqun();
        p.getCase(12).actionCase(j, p);
        break;
    }
    }
```

Figure 5 : *Méthode de la case 6*

Ici le joueur j a atterri sur la case numéro 6 et la méthode « verif_appartenance_case_vide » s'enclenche. Cette méthode vérifie dans un premier temps si la case n'appartient à personne. Nous supposons que dans ce cas la case n'a pas de propriétaire, ainsi nous avons l'interface graphique Ach_terrain qui se lance et qui prend en paramètre le plateau et le Joueur.

Dans le cas où **la case appartient à un propriétaire,** le joueur doit lui verser de l'argent avec la méthode « actionCase » et l'interface « Payer_Loyer » s'affichera.

```
default ->{
    if( etat != joueur.getId()){
        boolean constructible = famille[0] == famille[1];
        if(getMaison()== 0){
            int transaction = joueur.payer(getLoyers()[0]);
            if(constructible){
                transaction += joueur.payer(getLoyers()[0]);
            }
            Joueur j = plateau.getListeJoueur().get(etat);
            j.setPorteMonnaie(j.getPorteMonnaie() + transaction);
        }
        else{
            int transaction = joueur.payer(getLoyers()[getMaison()]);
            Joueur j = plateau.getListeJoueur().get(etat);
            j.setPorteMonnaie(j.getPorteMonnaie() + transaction);
        }
    }
}
```

Figure … : *Méthode Payer_Loyer*

Dans le cas où **le joueur atterrit sur une propriété d'un autre joueur dans la partie,** on entre dans l'algorithme « actionCase ». En fonction du nombre d'hôtels le joueur d'adverse possède sur cette case, le joueur « joueur » paiera le montant du loyer et le joueur « j » recevra l'argent de « joueur ».

Dans le cas où **le terrain n'appartient à personne,** l'interface graphique « Acheter_Terrain » proposera au joueur d'acheter ou non le terrain.

```
public class Clicklistener implements ActionListener
{
  @Override
  public void actionPerformed(ActionEvent e)
  {
    if (e.getSource() == oui)
    {
      Achete ach = new Achete("ce terrain");
      c.acheter(j, p);
      dispose();
    }

    if (e.getSource() == non)
    {
      dispose();
    }
  }

}
```

Figure 6 : *Méthode Acheter_Terrain*

Si le joueur appuie sur non, la fenêtre se ferme. Sinon la méthode « acheter » se lance en prenant comme argument le joueur qui joue son tour et le plateau sur lequel se trouve le joueur.

```
public void acheter(Joueur j, Plateau p){
    j.payer(getPrix());
    this.proprietaire = j.getId();
    j.addListeTerrain(super.getId());
    p.changementProprietaire(j.getId(), super.getId());
    achetee(p, j.getId());
}
```

Figure 7 : *Attribut du joueur*

Le joueur « j » va payer le prix du terrain et on attribue son identifiant au terrain avec « changementPropriétaire(id du joueur, id du terrain) ».

V. MOTIVATIONS PERSONNELLES

BELKHITER Yannis : Au cours de ce projet, j'ai réellement apprécié la grande autonomie dont nous avons fait preuve, et la grande liberté créative qui nous a été laissée. Nous avons vraiment pu apprendre par nous même via internet et les cours un nouveau type de programmation : l'interface graphique. Nous avons connu beaucoup de difficulté au départ puisque nous ne connaissions rien à ce sujet. Néanmoins, nous avons fait preuve d'une grande persévérance avec Maxime pour résoudre certains bugs persistants, ou réussir à implémenter nos interfaces aux codes de nos collègues. J'ai aussi beaucoup apprécié l'organisation de travail que nous avons réussi à mettre en place. En effet elle se rapproche vraiment de la structure organisationnelle des grandes entreprises lorsqu'elles travaillent en équipe sur un long projet. La mise en commun des différentes parties du programme est certainement la partie qui nous a demandé le plus de temps et le plus d'énergie. Nous avons connu de gros bugs complexes à résoudre (exemple : soucis lors d'ajouts de plusieurs joueurs). Mais en persévérant, et en revoyant la structure interne de la gestion d'une partie (automatisation du système), nous avons réussi à enfin produire une version jouable à plusieurs joueurs.
Je suis cependant assez frustré de ne pas continuer ce projet car nous aurions aimé implémenter plus de fonctionnalités (parties en réseau, hypothèque, ou encore enchères).
Je suis vraiment content d'avoir participé à un projet comme celui-ci et je pense avoir appris énormément de choses sur Java grâce à ce travail de groupe.

FORRAY Léo-Paul : Lors de ce projet, j'ai principalement été ralentis par mon manque d'habitude à coder en orienté objet. Ce souci s'est principalement illustré sur le plateau, qui est une liste de Cases. Or pour pouvoir appliquer des méthodes sur ces cases, il ne suffit pas de les définir dans une des classes finales, mais il faut la définir comme abstraite dans la classe Cases et l'implémenter dans chacune de ses classes filles. J'ai ainsi dû ajouter de nombreuses méthodes de partout, mais aussi parfois des arguments qui permettent de simplifier le code. Mais j'ai obtenu grâce à cela plus d'expérience concernant la création de diagrammes de classes.

MU Maxime : L'idée de concevoir un jeu m'a donné la curiosité de savoir comment un jeu est développé sur java. Savoir comment les objets sont reliés entre eux. Le Monopoly est un parfait exemple. Les cartes, les cases, les joueurs et les cagnottes sont tous liés entre eux. Puisque le joueur affecte les cartes qui elle modifie les cases et même les cagnottes des joueurs. C'est là où réside toute la difficulté du jeu, lier ces classes entre elles tout en respectant l'ordre de passage des joueurs.

SHCHERBAKOV Vadim : Ce projet m'a notamment permis d'appliquer les principes vus en cours et de découvrir le travail de groupe lors d'un développement de projet. La concrétisation de ce qui a été appris est donc un point positif de ce projet. De plus j'ai pu rencontrer les problèmes auxquels on doit faire face lors de projets informatiques qui nécessitent la mise en commun de code et surtout d'idées.

VI. ANNEXE

En scannant ce QRCode, les règles du jeu de ce monopoly vont vous être accessible.

Figure 8 : *QRCode des règles du jeu*