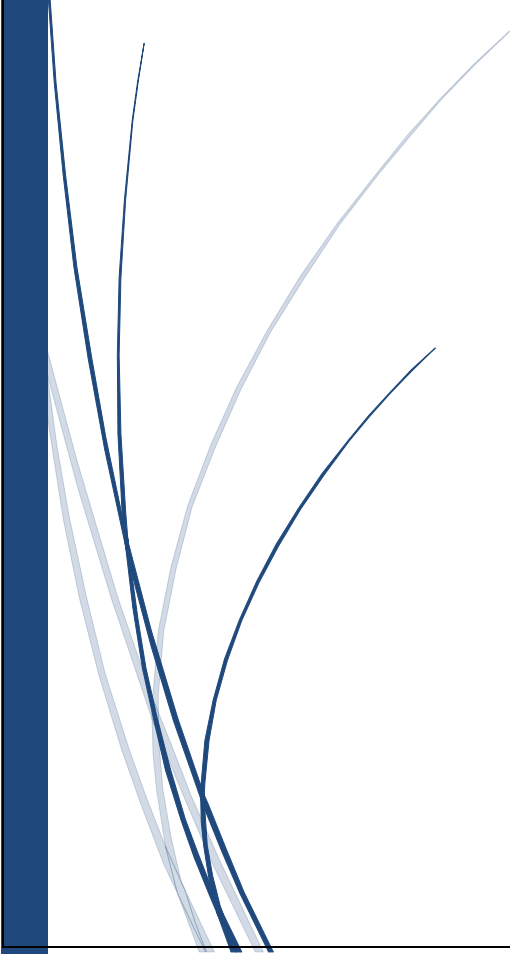


A thick dark blue vertical bar runs along the left edge of the page. A medium blue arrow points to the right, overlapping the bar and containing the text '[Date]'.

[Date]

PROJET JAVA

Casse Tête Lignes

Several thin, curved lines in dark blue and light grey originate from the bottom left corner and sweep upwards and to the right, creating a dynamic, abstract design.

BAH Thierno P1518166
ZAFIFOMENDRAHA Gabriello P1311399

PLAN

INTRODUCTION.....	2
I. LES DIAGRAMMES UMLS.....	3
1. Diagramme de paquetage.....	3
2. Diagramme des cas d'utilisation.....	3
3. Diagramme des classes.....	4
4. Diagramme de séquence.....	5
II. LISTE DES FONCTIONNALITES	6
1. Génération de la grille de jeu.....	6
2. Mise à jour grille.....	6
3. Fin du jeu.....	7
CONCLUSION.....	8

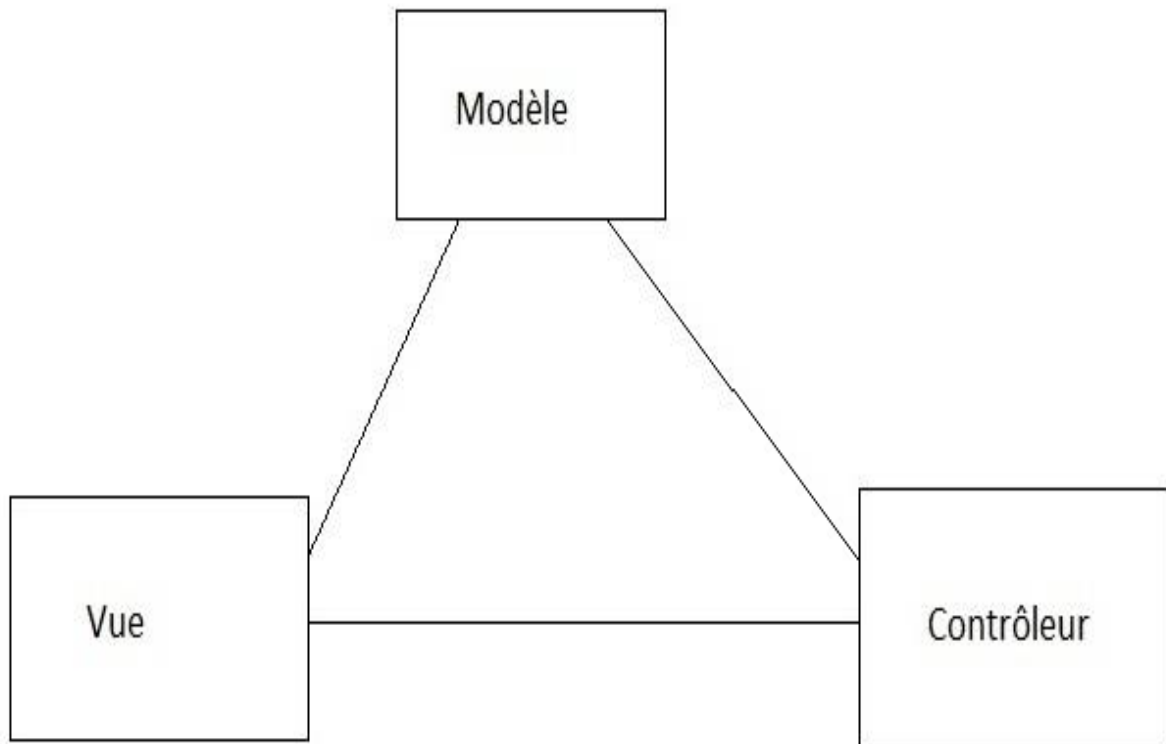
INTRODUCTION

Le jeu de Casse-tête lignes consiste à connecter deux paires de symboles en passant par un chemin et qu'une fois toutes les paires d'objets seront reliées la partie prend fin et le joueur aura donc la possibilité soit de passer à un niveau supérieur, soit terminer le jeu. Pour ce faire nous avons utilisé Java FX qui est une bibliothèque du langage Java et qui nous offre maintes solutions pour la création d'interfaces graphiques. Nous avons travaillé selon le modèle MVC (Model View Controller) dont l'objectif est d'appliquer le principe de séparation des traitements des informations et de la mise en forme logicielle.

I- LES DIAGRAMMES UMLs

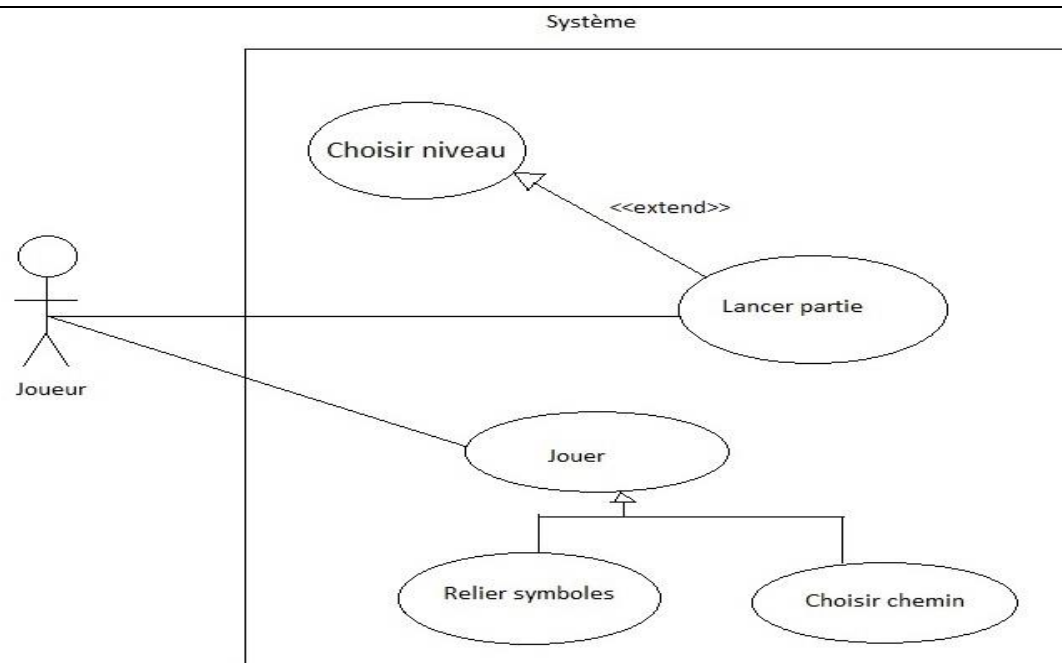
1) Le diagramme de paquetage

Il permet d'organiser notre projet en groupe et donc mettre en évidence le modèle MVC, nous avons donc trois packages dont le Modèle, la Vue et le Contrôleur.



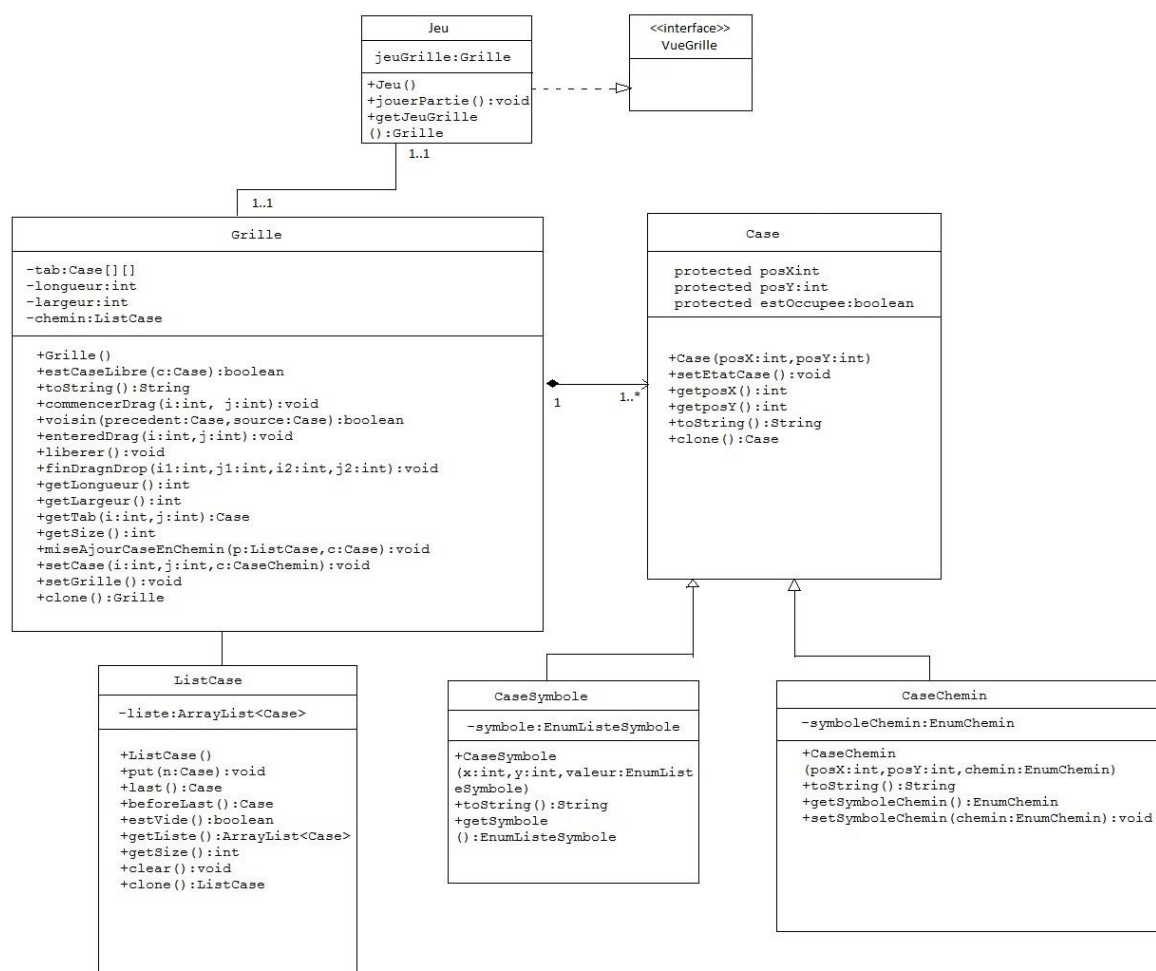
2) Le diagramme des cas d'utilisations

Il implémente les différentes fonctionnalités de notre système. Dans notre cas l'acteur principal est le joueur et il interagit directement avec notre système.



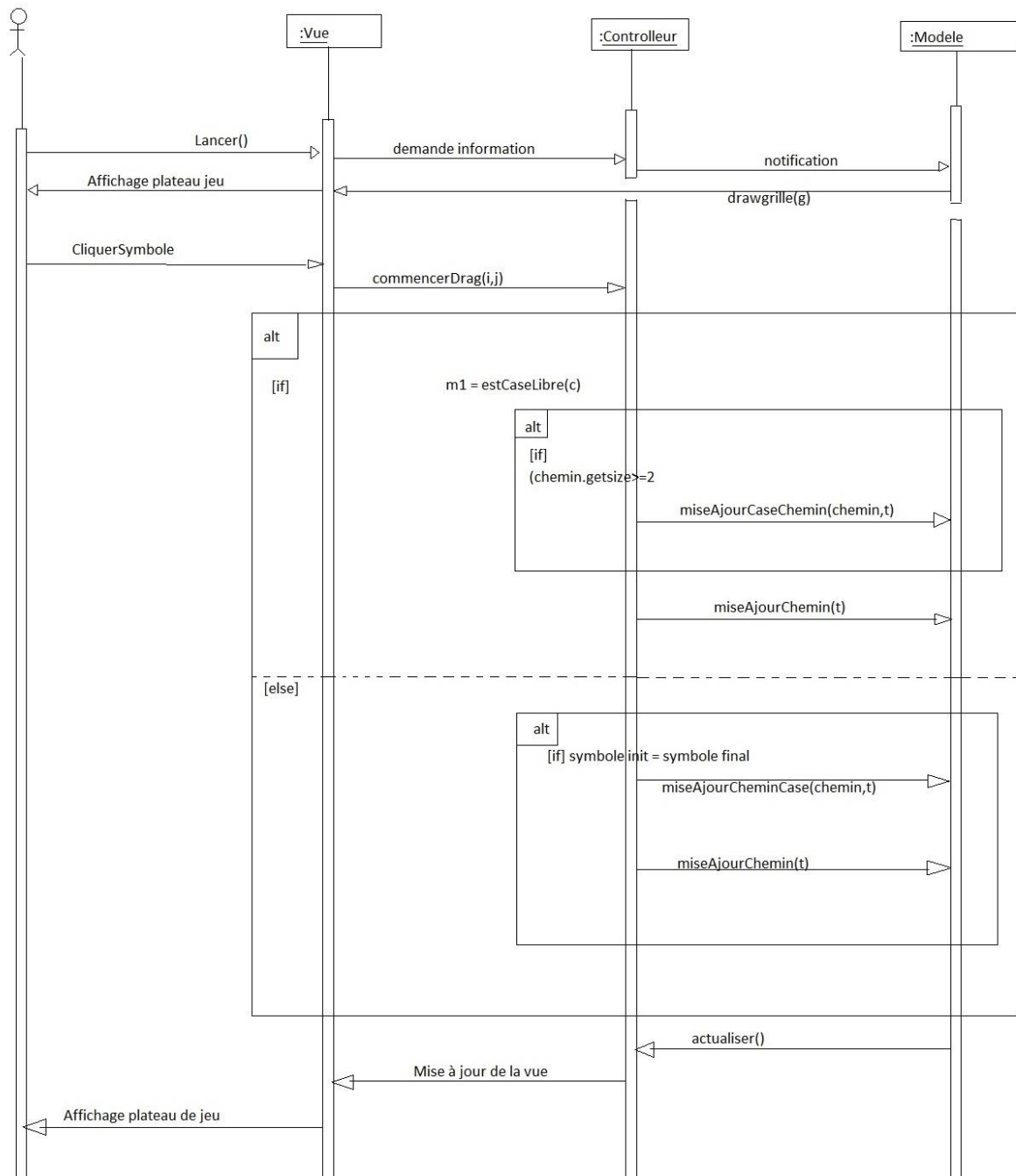
3) Diagramme des classes.

Il met en exergue, toutes les classes de notre système et leurs interactions entre elles.



4) Diagramme de séquence

Il permet de montrer comment le joueur interagit avec les éléments de notre système et comment ces éléments interagissent entre eux aussi. Nous implémentons le diagramme de séquence du drag and drop entre deux objets.



II) LISTE DES FONCTIONNALITES

1- Génération de la grille de jeu

La grille fait partie de nos modèles et sera appelé dans la vue pour l'affichage. Elle est initialisée par nous-même et la taille est modifiable. Lors de la génération de celle-ci, les symboles à relier sont mis en place par nos soins, c'est-à-dire qu'ils ont déjà des places prédéfinies et ne se fait pas aléatoirement.

```
public class Grille extends Observable implements Cloneable{
    private Case[][] tab;
    private int longueur;
    private int largeur;
    private Pile chemin;

    public Grille(){
        longueur = 3;
        largeur = 3;
        chemin = new Pile();
        tab = new Case[longueur][largeur];
        for(int i=0; i<longueur; i++){
            for(int j=0; j<largeur; j++){
                tab[i][j] = new Case(i, j);
            }
        }
        //on met e place les symboles à relier
        tab[0][0] = new CaseSymbole(0, 0, EnumListeSymbole.SPIRALE);
        tab[0][2] = new CaseSymbole(0, 2, EnumListeSymbole.ETOILE);
        tab[2][1] = new CaseSymbole(2, 1, EnumListeSymbole.SPIRALE);
        tab[2][2] = new CaseSymbole(2, 2, EnumListeSymbole.ETOILE);
    }
}
```

2- Mise à jour grille

Le but du jeu est de relier deux mêmes symboles sans passer par un chemin déjà emprunter auparavant. Cela est fait grâce au « drag and drop » (glisser et déposer). Le principe est que le joueur clique sur le symbole à relier, le glisse (drag) jusqu'au symbole cible (l'autre pair) et le dépose (drop). Tout au long de l'action, le chemin emprunté est tracé (la liste de chemin emprunté est stockée dans une pile ici dans la class ListCase).

```
public void commencerDrag(int i, int j){
    //on vérifie si on a bien une symbole
    if(tab[i][j].getClass().getName().equals("Modele.CaseSymbole")){
        CaseSymbole c = (CaseSymbole)tab[i][j];
        //on vérifie si le symbole n'est pas le symbole vide
        if(c.getSymbole() != EnumListeSymbole.VIDE){
            chemin.put(tab[i][j]);
        }
    }
}

public void enteredDrag(int i, int j) {
    if(estCaseLibre(tab[i][j]) && voisin(tab[i][j], chemin.last())){
        if(chemin.getSize() >= 2){
            miseAJourCaseEnChemin(chemin, tab[i][j]);
        }
        chemin.put(tab[i][j]);
    }else if(tab[i][j].getClass().getName().equals("Modele.CaseSymbole") && voisin(tab[i][j], chemin.last())){
        miseAJourCaseEnChemin(chemin, tab[i][j]);
        chemin.put(tab[i][j]);
    }
}
```

3- Fin du jeu

Le joueur gagne lorsqu'il réussit à relier ensemble tous les paires de symboles de la grille, c'est-à-dire qu'il a fait l'action glisser (drag) et déposer (drop) et qu'au cours du jeu il n'est pas passé par un chemin déjà emprunter auparavant.

```
public void finDragnDrop(int i1, int j1, int i2, int j2){
    // verifier que le type de la case est CaseSymbole
    if(chemin.getListe().get(0).getClass().getName().equals("Modele.CaseSymbole") &&
        chemin.getListe().get(chemin.getSize()-1).getClass().getName().equals("Modele.CaseSymbole")){
        CaseSymbole c1 = (CaseSymbole)chemin.getListe().get(0);
        CaseSymbole c2 = (CaseSymbole)chemin.getListe().get(chemin.getSize()-1);
        if(c1.getSymbole() != c2.getSymbole()){
            liberer();
        }else{
            setGrille();

            liberer();

            setChanged();
            notifyObservers();
        }
    }else{
        liberer();
    }
}
```


CONCLUSION

Ce projet nous a permis de voir mais aussi d'appliquer le modèle MVC en utilisant la bibliothèque java pour l'interface graphique qui est Java FX.

On n'a pas pu implémenter d'autres fonctionnalités qui auraient été intéressantes pour le jeu dans le temps imparti (faire d'autres niveaux un peu plus difficile, IA pour générer des positions valides pour des paires de symboles sur différents formats de grilles, IA pour résoudre partiellement ou globalement le puzzle, etc...) mais il aurait été intéressant de le faire.