

# **Rapport UML AP4B**

Professeur d'UV F. Gechter

Automne 2022

---

Alan HERVE, Augustin ATHANE, Gilles MAURER, Raphaël PERRIN

---

Décembre, 2022

## TABLE OF CONTENTS

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Matériel et lexique</b>	<b>4</b>
<b>3</b>	<b>Déroulé du jeu</b>	<b>5</b>
3.1	Principe général . . . . .	5
3.2	Déroulement d'une partie . . . . .	6
3.3	Déroulement d'une manche . . . . .	7
3.4	Déroulement d'un tour de jeu . . . . .	8
3.5	Jeton Malus . . . . .	12
<b>4</b>	<b>Fin de manche et comptage des points</b>	<b>12</b>
<b>5</b>	<b>Possibilités du joueur (Diagramme de cas d'utilisation)</b>	<b>16</b>
<b>6</b>	<b>Conception MVC (Diagramme de séquence)</b>	<b>18</b>
<b>7</b>	<b>Conception des classes (Diagramme de classe)</b>	<b>21</b>
7.1	La partie Modèle . . . . .	21
7.2	Le pot . . . . .	23
7.3	La partie vision . . . . .	25
<b>8</b>	<b>Interaction des classes du modèle (Diagramme de séquence)</b>	<b>27</b>
<b>9</b>	<b>Projection de l'implémentation en Java</b>	<b>29</b>
<b>10</b>	<b>Modifications après l'implémentation</b>	<b>29</b>

## 1/ INTRODUCTION

Ce projet est réalisé dans le cadre de l'UV AP4B. Notre tâche est de mener un projet collaboratif afin de mettre en application les connaissances acquises durant notre semestre. Le but est de développer un programme en se focalisant principalement sur la conception et sur la partie programmation du cœur de ce dernier. Dans ce travail de groupe, nous avons choisi d'adapter le jeu Azul en rapport avec le thème de l'UTBM. Le but du jeu est ainsi de réussir à combiner tous les types d'UV afin d'obtenir son diplôme.

Lors de ce rapport, nous allons détailler les règles du jeu, avant d'expliquer les diagrammes UML choisis et réalisés pour la conception du programme.

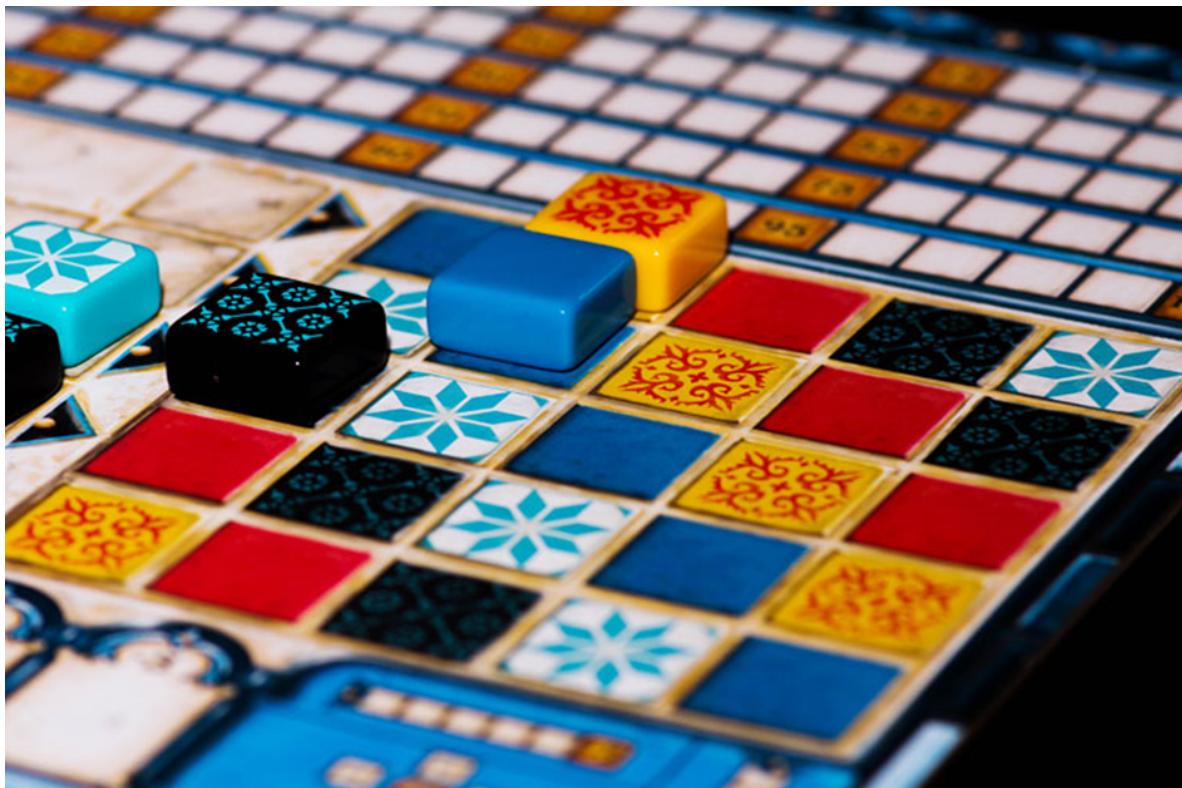


Figure 1: Jeu de société Azul

## 2/ MATÉRIEL ET LEXIQUE

Dans ce jeu, on peut retrouver des tuiles de 5 couleurs différentes représentant les 5 catégories d'UV de l'UTBM.

- Bleu pour Algorithmic and programming
- Orange pour Theoretical Computer Science
- Jaune pour Information Systems and databases
- Violet pour Networks
- Vert pour Hardware and Systems

Dans la suite du rapport et dans le code, on désignera les tuiles par "*Tiles*".

Chaque joueur possède son propre plateau, où on peut trouver une grille qu'on appellera "*Pattern*", des lignes de différentes tailles qu'on appellera "*PlayGrid*", une ligne de malus appelée "*Malus*" et d'un tableau de score appelé "*Score*". Le plateau du joueur sera nommé "*Bord*".

Au centre du jeu on retrouve aussi plusieurs éléments : le sac qui contient les tiles restantes et qui sera appelé "*Bag*", des disques qui seront nommés "*Pile*" et une pile de milieu qui sera appelée "*MiddlePile*".



### 3/ DÉROULÉ DU JEU

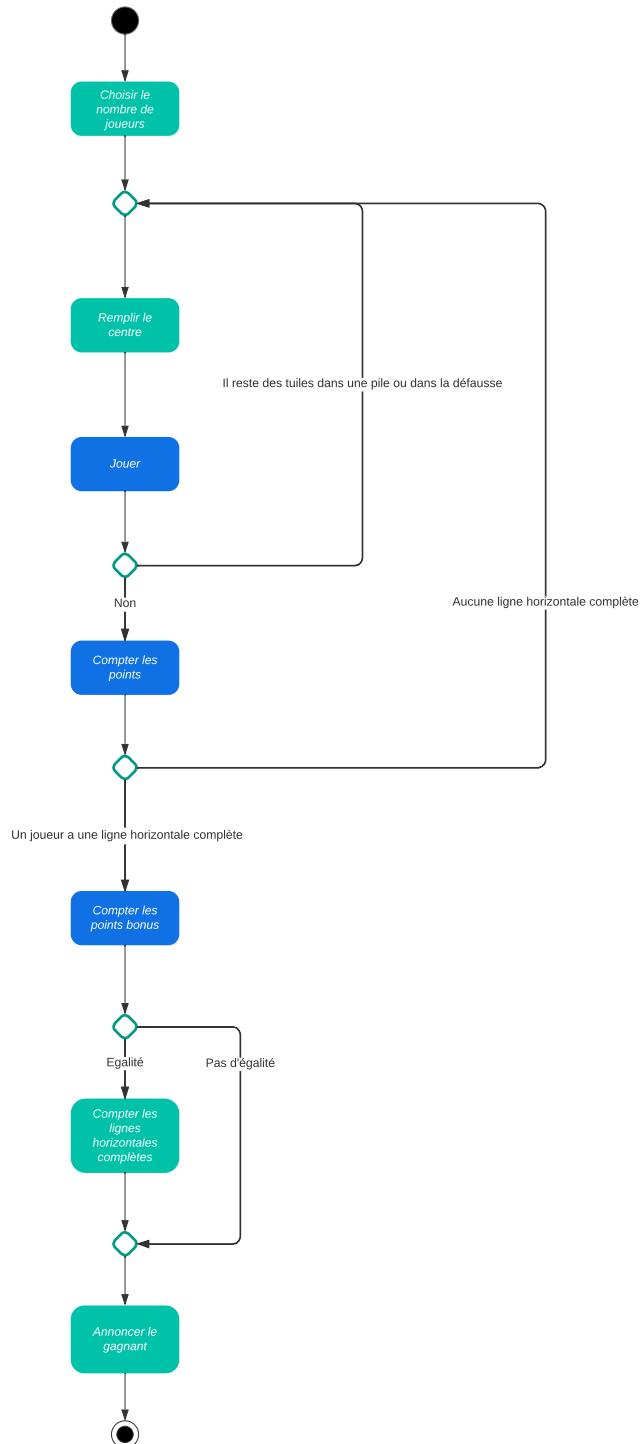
#### 3.1/ PRINCIPE GÉNÉRAL

Azul est un jeu de plateau de stratégie conçu pour 2 à 4 joueurs, une partie dure en moyenne 30 à 45 minutes. Le but du jeu est de compléter sa grille pattern avec des tiles en essayant de les aligner le plus possible. Le jeu se joue en plusieurs manches jusqu'à ce qu'un joueur ait totalement complété une ligne de sa grille pattern. Pour étudier le déroulé du jeu et les règles, nous allons utiliser un diagramme d'activité. Le diagramme d'activité permet de se rendre compte de l'ordre des actions durant la partie, il présente les différentes actions effectuées sans tenir compte de qui les effectue. Il permet aussi de voir les conditions qu'il faut valider pour faire certaines actions.

Le diagramme d'activité est décomposé en plusieurs parties : Le diagramme global qui permet de voir le déroulé d'une partie, et plusieurs diagrammes secondaires qui permettent de se pencher sur des points plus précis, et donc, de ne pas surcharger le diagramme principal.

### 3.2/ DÉROULEMENT D'UNE PARTIE

Diagramme principal



Ce diagramme décrit le déroulé général d'une partie. Après avoir choisi le nombre de joueurs, les joueurs jouent chacun leur tour jusqu'à la fin de la manche. Une nouvelle manche commence ensuite et ainsi de suite jusqu'à la fin de la partie. Ces deux boucles sont représentées dans le diagramme.

À la fin de la partie, les points bonus sont comptés et ajoutés au score (actions décrites dans un autre diagramme d'activité), ensuite une condition permet d'agir en cas d'égalité. Finalement, le gagnant de la partie est annoncé.

### 3.3/ DÉROULEMENT D'UNE MANCHE

Une partie se déroule en plusieurs phases que l'on peut voir sur le diagramme.

La préparation :

- On commence par remplir les Piles avec chacune 4 Tiles prises au hasard dans le Bag et on remet le jeton malus (voir jeton malus) dans la Middle Pile.

Les tours de jeu :

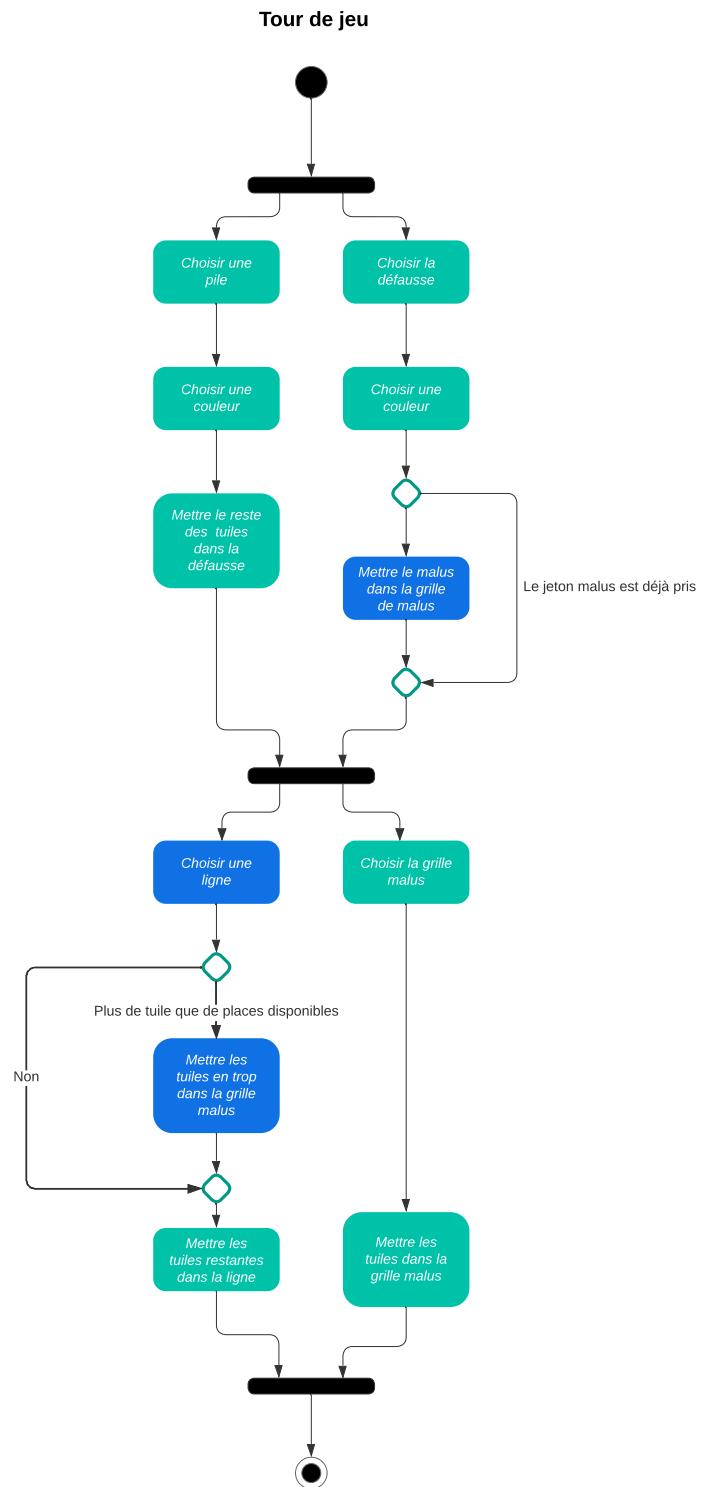
- Les joueurs vont jouer chacun à leur tour (voir déroulement d'un tour de jeu) jusqu'à ce qu'il ne reste plus aucune Tile sur les Piles et dans la Middle Pile.

Le comptage des points :

- Pour chaque joueur, on déplace certaines tiles de la playgrid à la grille pattern et on compte les points gagnés durant la manche (voir fin de manche et comptage des points).

La partie se termine dès qu'un joueur complète une ligne sur son motif.

### 3.4/ DÉROULEMENT D'UN TOUR DE JEU



Lors du tour d'un joueur, celui-ci a deux choix, comme on peut le voir sur le diagramme, il doit sélectionner une tile de son choix, soit dans une des piles disponibles, soit dans la Middle Pile.

- Il prend toutes les tile de la couleur qu'il a choisi qui sont sur la même Pile (ou dans la Middle Pile).
- Les tiles non choisies donc d'une autre couleur vont dans la Middle Pile (s'il a choisi une Pile).
- S'il est le premier à prendre des tiles dans la Middle Pile il reçoit le jeton malus (voir jeton malus) à placer sur sa grille de pénalités.

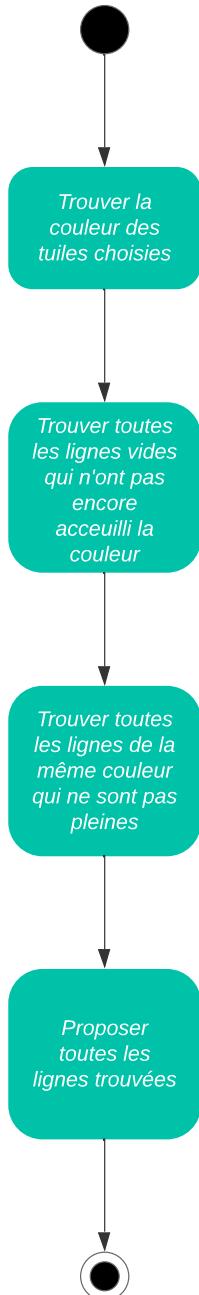
Par exemple si une pile contient 2 tiles orange, 1 jaune et 1 bleue et que le joueur choisit une orange, il récupérera les 2 tiles orange. La tile jaune et la tile bleue iront dans la Middle Pile. La Middle Pile ne contient aucune tiles en début d'une manche elle se remplit ensuite avec toutes les tiles restantes à chaque fois qu'un joueur choisi une pile. Quand le joueur sélectionne la Middle Pile il récupère toutes les tiles de la même couleur qui y sont, les autres tiles sont laissées dans la Middle Pile.

Une fois la source et la couleur de tile choisie, le joueur doit sélectionner où les placer sur sa Playgrid. La Playgrid est composée de 5 lignes qui ont une taille allant de 1 à 5 tiles. Pour placer des tiles dans une ligne les conditions suivantes doivent être respectées :

- La ligne ne doit pas être pleine.
- Si des tiles sont déjà présentes sur la ligne, les tiles que le joueur cherche à placer doivent être de la même couleur que celles-ci.
- Chaque ligne est associée à une ligne de la grille Pattern et chaque ligne de la grille Pattern possède une case de chaque couleur de tiles. Quand la ligne de PlayGrid sera complète, la case de Pattern correspondante à la couleur sera complétée. Ainsi, si la ligne est vide, il faut également qu'elle n'ait pas déjà été occupée par la couleur des tiles actuellement choisie.

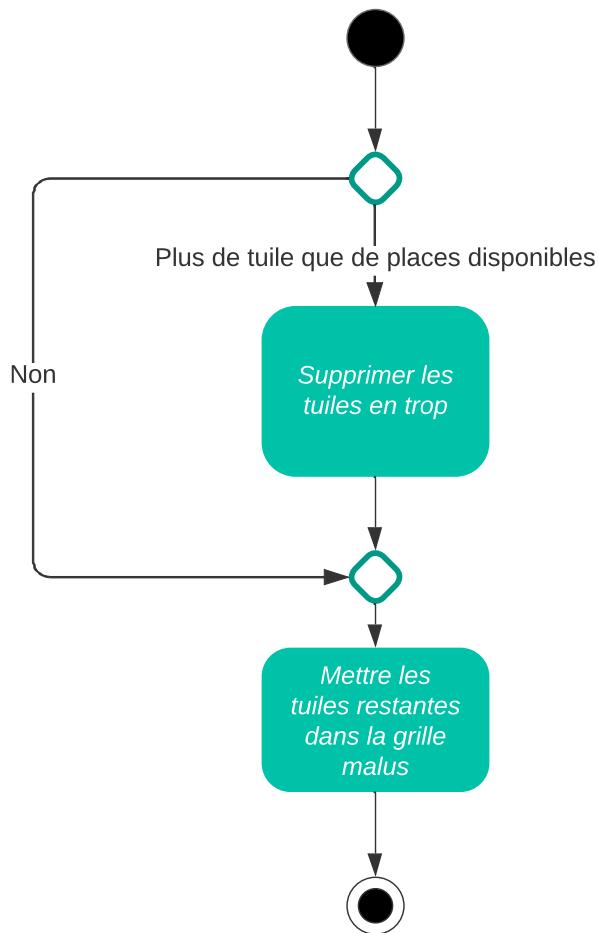
Ces conditions sont détaillées dans le diagramme d'activité *sélection ligne*.

## Sélection ligne



Si le nombre de tiles est supérieur au nombre de cases vides sur la ligne de playgrid, les tiles en trop devront être placées sur la grille de malus. Si la grille de malus est déjà remplie, on supprime les tiles en trop, cette action est décrite dans le diagramme d'activité remplissage grille malus. La suppression des tiles revient en fait à les stocker puis à les remettre dans le bag à la fin de la manche.

## Remplissage grille malus

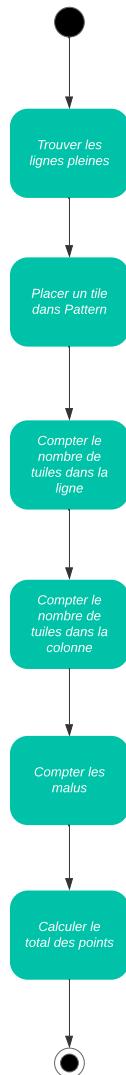


### 3.5/ JETON MALUS

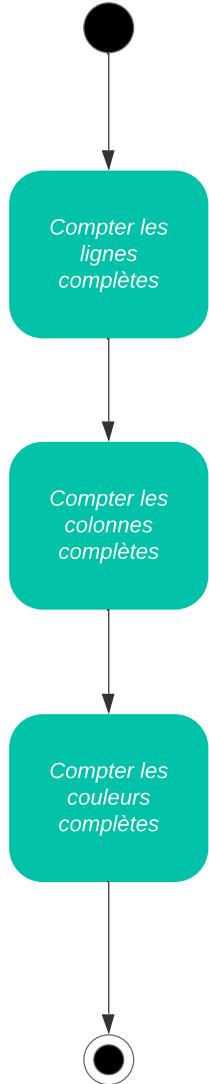
Le jeton malus est une petite particularité, à chaque manche, il est mis dans la Middle Pile puis le premier joueur qui veut récupérer des tiles dans la Middle Pile est obligé de récupérer ce jeton et de le rajouter dans sa grille de malus.

## 4/ FIN DE MANCHE ET COMPTAGE DES POINTS

### Comptage des points



## Comptage des points bonus



Le comptage des points se sépare en deux parties, le comptage des points réalisé par les joueurs après chaque manche et le comptage des points bonus calculés en fin de partie. Ces deux parties ont chacune un diagramme d'activité qui décrit leur fonctionnement.

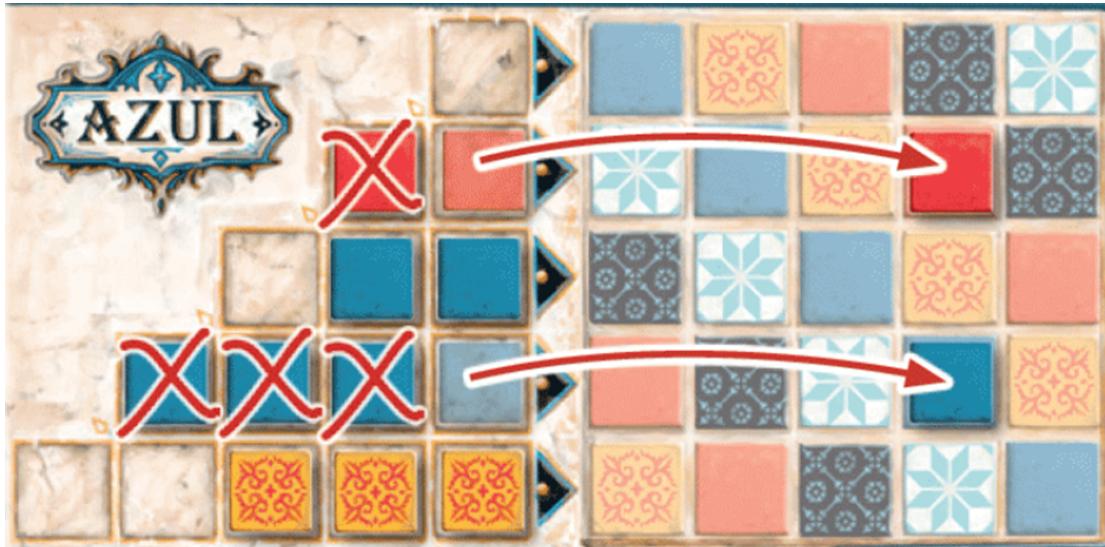
- A la fin de chaque manche :

- On regarde les lignes de PlayGrid

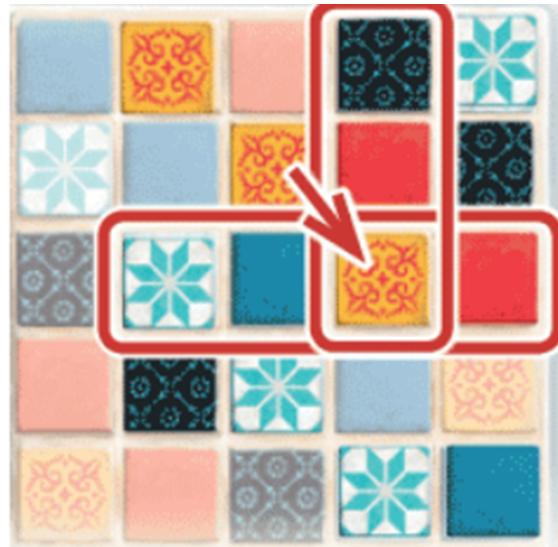
- \* Pour chaque ligne pleine, le joueur prend une tile de celle-ci et la place sur la grille pattern, sur

la même ligne et sur la case ayant la même couleur que la tile, les autres tiles sont remises dans le bag.

- \* Il faut ensuite compter le nombre de tiles qui se touchent dans la ligne où on vient d'en placer une. Chaque tile rapporte 1 point.
- \* On fait ensuite la même chose pour la colonne correspondante.
- \* Toutes les tiles qui sont dans la grille Pattern ainsi que toutes les lignes qui ne sont pas complètes sont conservées pour la manche suivante.



Dans cet exemple, on voit que la tuile orange qui vient d'être placée rapporte 3 points en vertical et 4 en horizontal soit un total de 7 points



- On regarde ensuite la grille de malus
  - \* Pour chaque case de la grille malus remplie, on enlève au score la valeur de la case en question.
  - \* On remet dans le bag toutes les tiles de la grille.

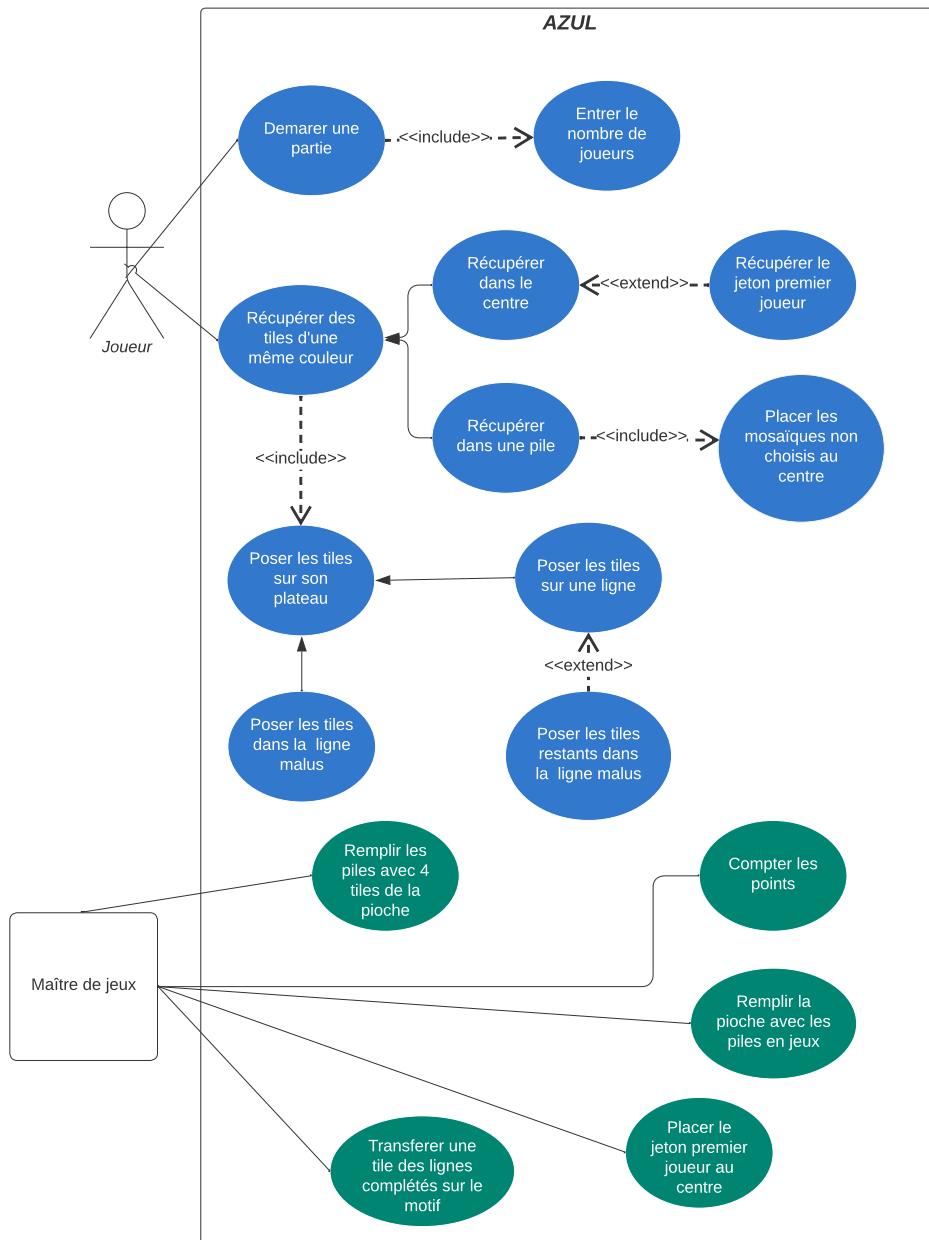
- À la fin de la partie :

- Chaque colonne remplie en entier rapporte 7 ECTS.
- Chaque ligne remplie en entier rapporte 2 ECTS.
- Chaque couleur dont on a placé 5 tiles rapporte 10 ECTS.

## 5/ POSSIBILITÉS DU JOUEUR (DIAGRAMME DE CAS D'UTILISATION)

D'après les règles du jeu et le déroulé d'une partie, on peut analyser quelles sont les actions qui vont devoir être réalisées par les joueurs. Nous avons donc fait un diagramme use case qui permet de représenter ces actions.

Diagramme de cas d'utilisation



Le joueur peut interagir avec le jeu uniquement lors de son tour, ou lorsqu'il lance le jeu. Lorsque l'utilisateur lance le jeu, il entre le nombre de joueurs.

Durant le tour du joueur, celui-ci commence par choisir des tiles qu'il récupère soit dans le centre (MiddlePile), soit sur une pile. Le joueur pose ensuite ses tiles sur une de ses lignes ou, s'il le souhaite, directement sur sa ligne malus. Sur le diagramme, ces différentes actions sont détaillées en montrant les sous-actions qui sont induites (include). On peut aussi voir sur le diagramme les différents choix que doit faire le joueur. Par exemple, quand il récupère des tiles, il doit choisir entre les récupérer dans une pile ou dans la Middle Pile. Le deuxième choix du joueur est réalisé durant la pose de ses tiles : il choisit soit une ligne de sa Playgrid, soit il pose directement ses tiles dans la ligne malus. Ces choix sont représentés dans le diagramme par des généralisations des actions globales récupérer des tiles et poser des tiles.

Dans ce diagramme nous pouvons retrouver une entité Maître du jeu. Le Maître du jeu représente toutes les actions effectuées par le programme en début et fin de manche et en fin de partie. Le "Maître du jeu" sur le jeu plateau classique serait celui qui mélange, distribue les tiles, et effectue toutes les opérations pour pouvoir jouer dans les règles. Nous avons trouvé plus claire de représenter cela de cette manière, plutôt que par des relations d'inclusions avec les actions des joueurs, car ces opérations sont effectuées régulièrement à des moments précis du jeu.

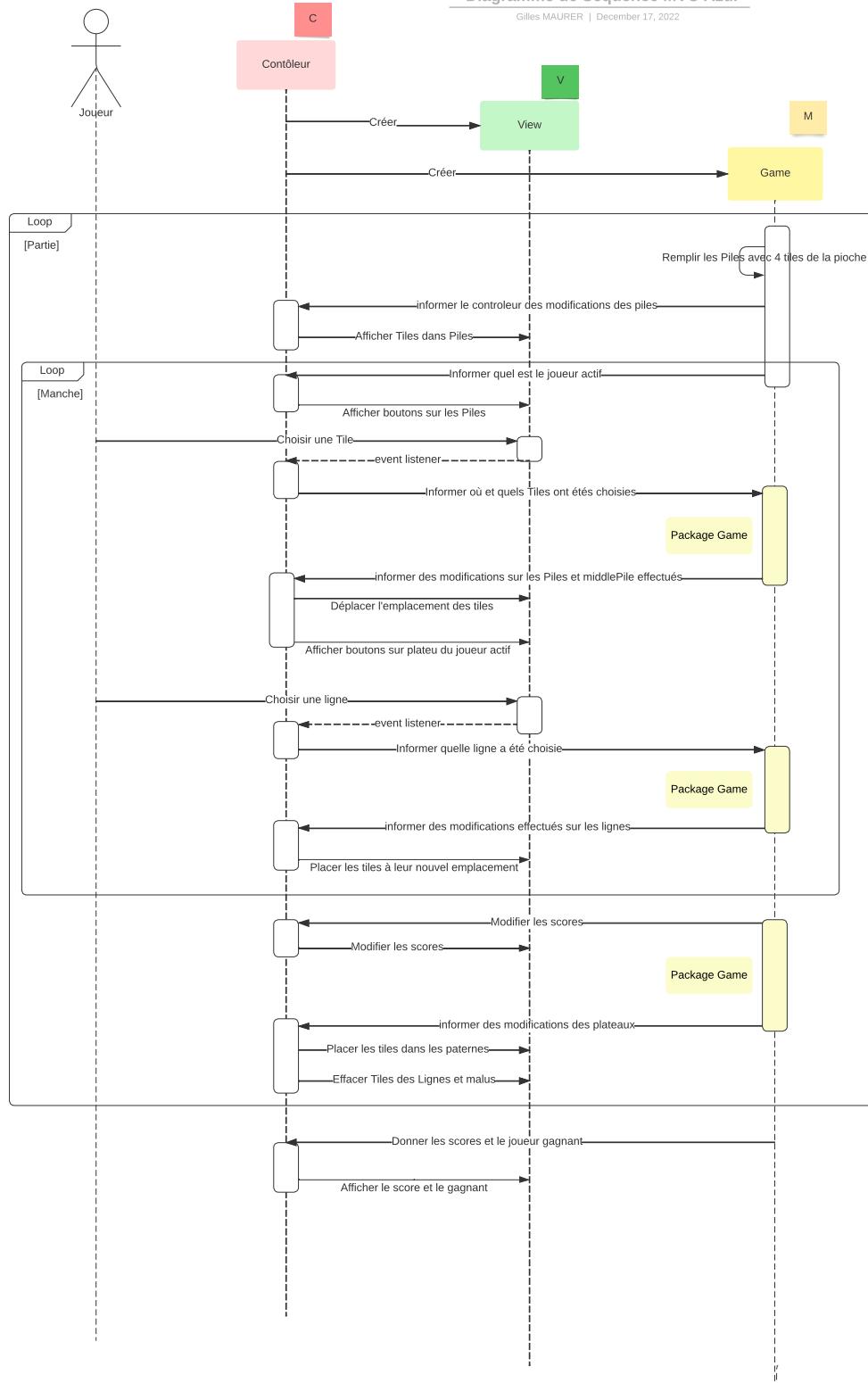
## 6/ CONCEPTION MVC (DIAGRAMME DE SÉQUENCE)

Nous avons décidé de baser notre conception sur le MVC (Model-View-Controller) qui est un modèle architectural qui sépare une application en trois composants logiques principaux : modèle, vue et contrôleur. Ces trois parties du MVC sont interconnectées. La vue affiche le modèle pour l'utilisateur. Le contrôleur accepte les entrées de l'utilisateur et met à jour le modèle et la vue. L'utilité est qu'il est facile de modifier la vue sans modifier le modèle, ou au contraire de modifier le modèle sans modifier la vue. Cela permet des économies de temps lors de la résolution de bugs ou dans l'ajout de nouvelles fonctionnalités pour le programme.

Nous avons donc modélisé cette architecture à l'aide d'un diagramme de séquence pour bien montrer toutes les interactions entre le modèle, la vue et le contrôleur. Le contrôleur est représenté par la classe Contrôleur, le module par la classe Game et la vue par la classe View.

### Diagramme de Séquence MVC Azul

Gilles MAURER | December 17, 2022



Ce diagramme montre uniquement les interactions durant le jeu. Ce qui se passe dans le menu et en fin de partie n'est pas représenté. On considère donc que le joueur a déjà lancé le jeu et entré le nombre de joueurs.

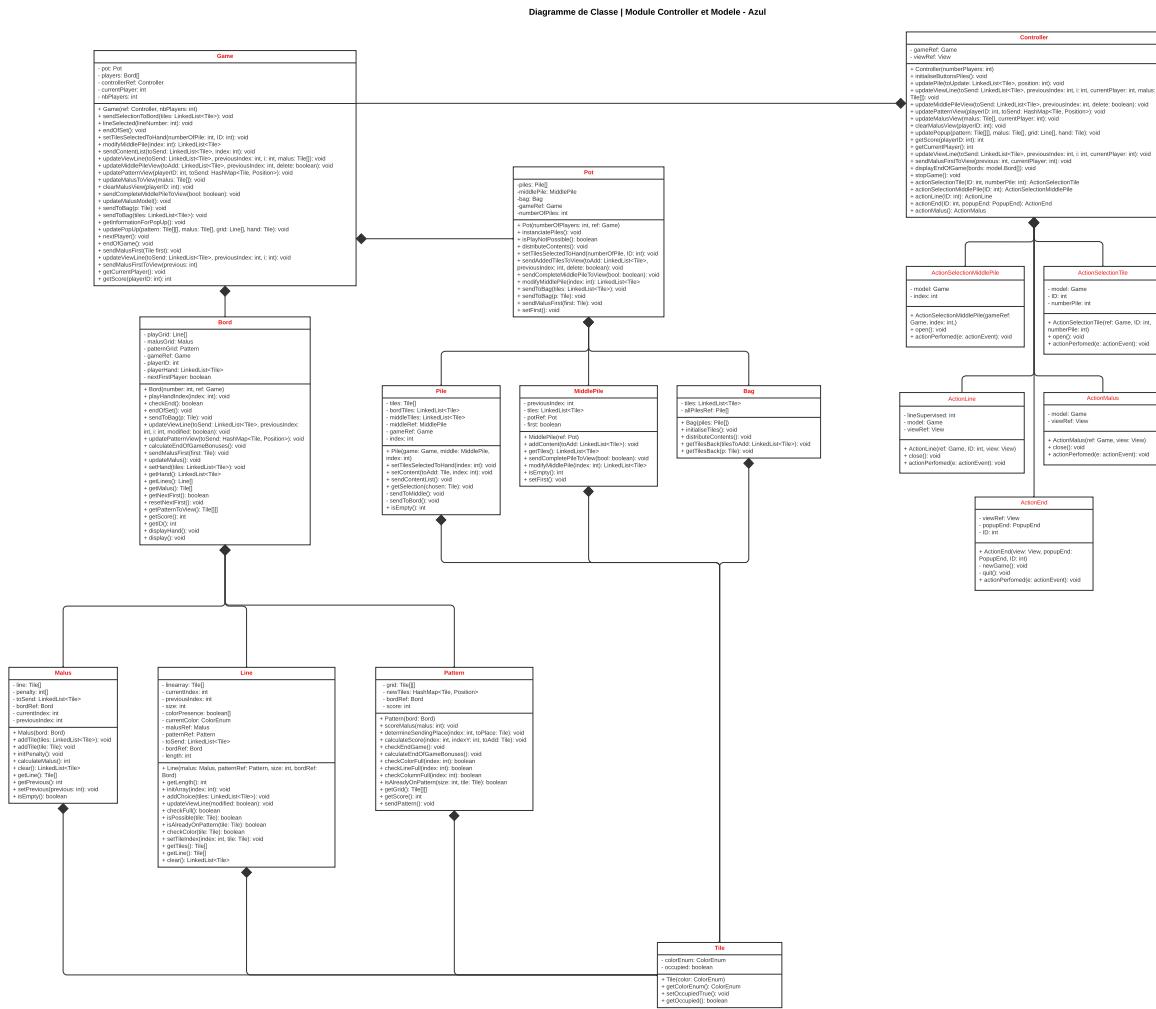
Tout ce qui se passe dans la vue ou le modèle n'est pas représenté ici, à l'exception d'un événement que nous avons ajouté pour une meilleure compréhension. L'événement Remplir les Piles avec 4 tiles de la pioche a été ajouté pour justifier la modification des piles de la vue. La vue et le module sont organisés en arborescence de telle manière que, tout ce qui est modifié dans l'un où dans l'autre est accessible facilement par la vue ou le modèle. Cela permet de faciliter les échanges entre ces deux composants.

Le joueur interagit avec la vue pour choisir une tile et ensuite placer cette tile, ces actions sont récupérées par le contrôleur grâce aux méthodes “event listener” des boutons. Il envoie ensuite les informations au modèle, qui informera plus tard le contrôleur de toutes les modifications à effectuer dans la vue.

Pour finir, tout ce qui se déroule dans le module aux endroits indiqués par “Package Game” sera expliqué dans la partie “Interaction des classes du modèle”.

## 7/ CONCEPTION DES CLASSES (DIAGRAMME DE CLASSE)

Comme dit précédemment ce projet respecte l'architecture MVC nous allons donc maintenant nous intéresser plus précisément à ces 3 parties.



## 7.1/ LA PARTIE MODÈLE

Les Bord: Il y a autant de Bord que de joueurs et ils sont contenus dans un array de Bord dans Game. Ils contiennent les 3 types de grilles présentes sur un plateau (hormis celle de score) :

- Malus : Contient une ligne de Tile, et qui diminue le score en fonction du nombre de Tile présentes sur celle-ci.
  - Un array de Lines : Line contenant une ligne de Tile sur laquelle on place nos Tiles après avoir fait notre choix auprès d'une Pile ou de MiddlePile (en vérifiant bien sur quelles ligne il est possible de placer notre

choix).

- Pattern : Contient un double array de Tile, parmi les trois grilles, il s'agit de la seule avec laquelle le joueur ne peut pas interagir.

Nous allons par la suite décrire les méthodes et attributs les plus importants de Bord et de ses composants :

Attributs	Public	Privé
Bord	<ul style="list-style-type: none"> <li>• playerID : l'identifiant du joueur</li> </ul>	<ul style="list-style-type: none"> <li>• excedent_tiles_selection : Les tiles en trop qui ne pouvaient plus être contenues dans Malus.</li> <li>• hand_of_player : La sélection actuelle du joueur avant que celui-ci ne la place sur son plateau.</li> </ul>
Malus		<ul style="list-style-type: none"> <li>• penalty: Contient la valeur de malus de chaque case de Malus.</li> <li>• line : Contient les Tiles placées dans la grille Malus.</li> </ul>
Line		<ul style="list-style-type: none"> <li>• current_index : indique jusqu'à quel point les cases d'une lignes ont étées remplies.</li> <li>• current_color: La couleur actuelle de la ligne.</li> <li>• discarded: les Tiles qui ne sont pas placées sur le plateau après qu'une ligne soit pleine.</li> <li>• color_presence: note quelles couleurs sont déjà présentes sur cette ligne.</li> </ul>
Pattern	<ul style="list-style-type: none"> <li>• end_trigger : Vérifie si un joueur a rempli la condition de fin de partie.</li> </ul>	<ul style="list-style-type: none"> <li>• new_colorpos: Est utilisé pour l'initialisation de la grid en début de partie.</li> </ul>

Méthodes	Public	Privé
Bord		
Malus		
Line	<ul style="list-style-type: none"> <li>• isPossible : Vérifie si on peut placer la sélection dans la ligne.</li> </ul>	
Line		<ul style="list-style-type: none"> <li>• calculateScore : Place une tile après avoir déterminé où l'envoyer et calcule les points rapporté par la tile.</li> <li>• determineSendingPlace : Détermine où la tile sera envoyée dans la ligne.</li> </ul>

## 7.2/ LE POT

Le pot représente toute la partie du jeu qui permet au joueur de récupérer des Tiles, c'est-à-dire :

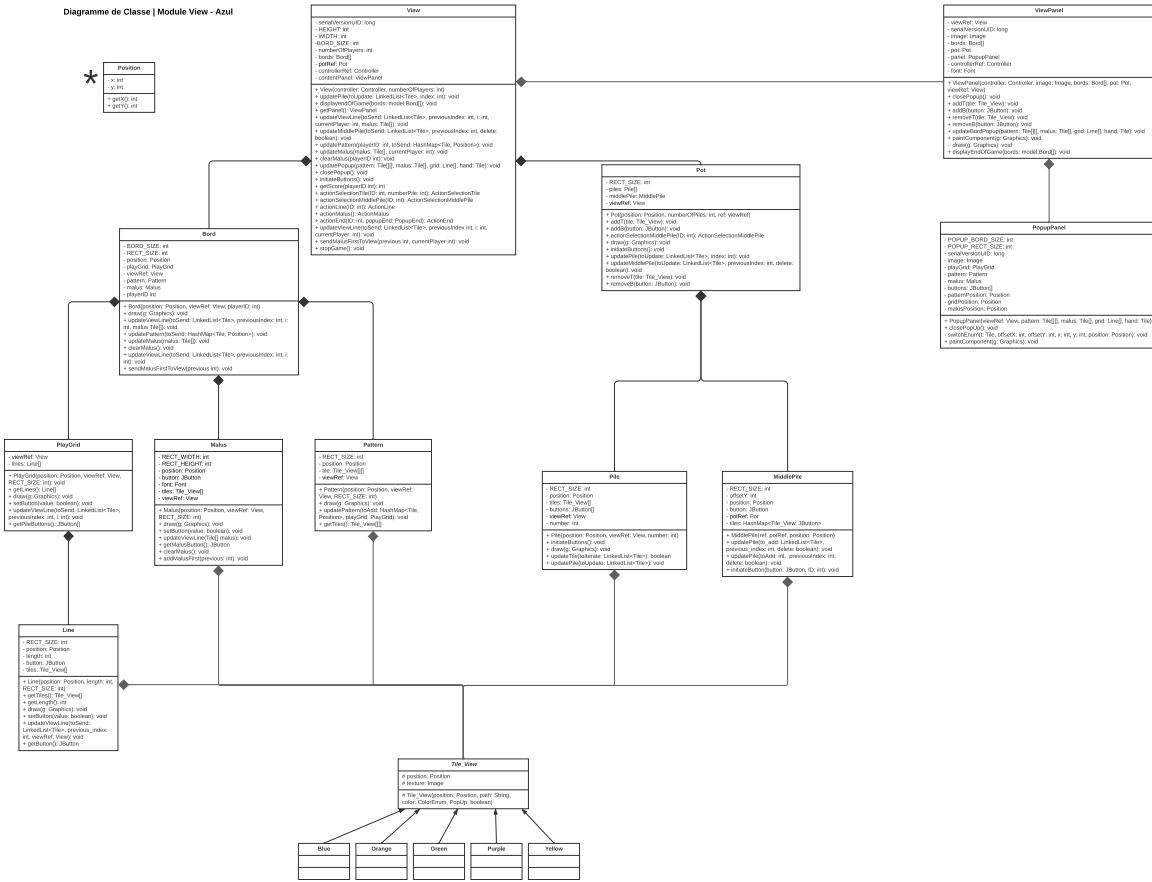
- Piles dont le nombre dépend du nombre de joueurs et auprès desquelles l'on peut récupérer jusqu'à 4 tiles maximum.
- MiddlePile qui est le rebut où vont les Tiles qui n'ont pas été choisies sur les Piles.
- Bag qui contient l'ensemble des Tiles qui ne sont pas actuellement en jeu et qui est chargé de les distribuer aléatoirement auprès des Piles au début de chaque manche.

Nous allons par la suite décrire les méthodes et attributs les plus importants de Pot et de ses composants :

Attributs	Public	Privé
Pot		
Pile		<ul style="list-style-type: none"> <li>• tiles_bord : Liste de tiles qui seront à envoyer au plateau du joueur actif.</li> <li>• tiles_middle : Liste de tiles à envoyer au rebut après que la selection soit faite.</li> </ul>
Middle Pile		<ul style="list-style-type: none"> <li>• first : Permet de déterminer si quelqu'un a déjà récupéré une tile.</li> </ul>
Bag		<ul style="list-style-type: none"> <li>• discarded_tiles : Tiles jetées qui seront à remettre dans le sac en fin de manche.</li> </ul>

Méthodes	Public	Privé
Pot		
Pile		
Middle Pile	<ul style="list-style-type: none"> <li>• addContent : Ajoute des Tiles au rebus.</li> <li>• getPartOfContent : Récupere toute les tiles d'une certaine couleur du rebus.</li> </ul>	
Bag	<ul style="list-style-type: none"> <li>• getTilesBack : Remet des Tiles depuis la partie Bord dans le sac.</li> </ul>	<ul style="list-style-type: none"> <li>• initialiseTiles : Met le nombre correct de Tiles dans le sac et mélange.</li> </ul>

## 7.3/ LA PARTIE VISION



Les composants de la partie modèle possèdent pour la plupart un équivalent dans la partie vision. Les composants n'ayant pas d'équivalents sont Game, car c'est une entité abstraite et Bag, car nous avons choisi de ne pas représenter le sac de Tiles sur le plateau afin de ne pas trop encombrer la vue.

De plus, cinq nouvelles classes héritent de l'équivalent de Tiles, il s'agit de :

- Blue
- Orange
- Green
- Purple
- Yellow

Chaque couleur représentant une catégorie de crédit qu'il est possible d'obtenir en branche info.

Chaque objet de la Vue possède comme attribut un attribut de type Position qui permet de repérer plus facilement

leur position sur la fenêtre.

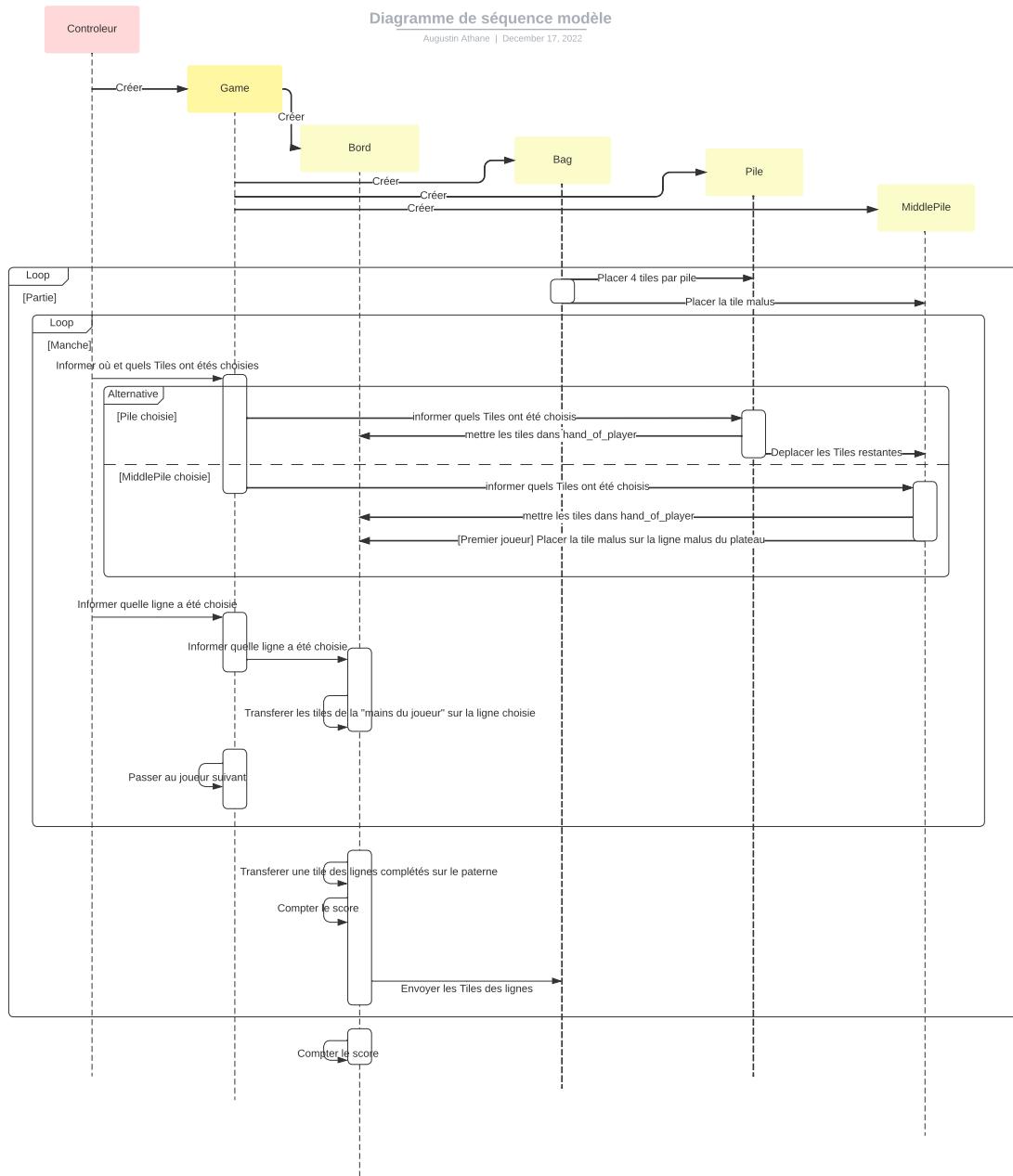
Position
x : int
y : int
+ getX() : int
+ getY() : int

Nom de l'objet	Boutons que contient l'objet	ActionListener
Line	SelectButton : JButton	ActionLine
Malus	SelectButton : JButton	ActionMalus
Pile	SelectButton : JButton OptionsButton : JButton[]	ActionDisplayPile ActionSelectionPile
Middlepile	SelectButton : JButton OptionsButton : JButton[]	ActionDisplayMiddlePile ActionSelectionMiddlePile

Nom du Listener	Action
ActionLine	Rajoute les tiles de la sélection actuelle dans la ligne que cette itération de ActionLine supervise.
ActionMalus	Rajoute les tiles en trop (ou mises volontairement) dans la ligne de Malus du joueur actuel.
ActionDisplayPile	Affiche les choix possible dans cette Pile, rajoute les boutons nécessaires à la sélection des tiles.
ActionDisplayMiddle	Affiche les choix possible dans MiddlePile, rajoute les boutons nécessaires à la sélection des Tiles.
ActionSelectionMiddlePile	Parcours la LinkedList de MiddlePile à la recherche de toutes les tiles de la même couleur que notre choix et les envoie au Bord du joueur actif. Envoie aussi si le joueur doit recevoir un malus dans le cas où il est le premier à prendre des Tiles dans MiddlePile.
ActionSelectionPile	Parcours la LinkedList de MiddlePile à la recherche de toutes les tiles de la même couleur que notre choix et les envoie au Bord du joueur actif.

## 8/ INTERACTION DES CLASSES DU MODÈLE (DIAGRAMME DE SÉQUENCE)

Maintenant que les classes ont été présentées, nous pouvons nous intéresser à leurs interactions. Nous avons déjà vu comment la vue, le modèle et le contrôleur interagissent entre eux, nous allons donc nous concentrer ici sur l'interaction des classes du modèle. Pour simplifier le diagramme, nous n'avons pas modélisé les remontées d'informations du modèle (classe Game) au contrôleur. Ces interactions étant déjà représentées dans le diagramme de séquence du MVC.



Avant chaque manche, les piles sont initialisées avec 4 tiles de la pioche. On place ensuite le jeton malus dans MiddlePile. La manche débute ensuite.

Durant la manche, lorsque le contrôleur informe où et quels Tiles ont été choisis, deux options sont possibles :

- Si le joueur a choisi des tiles sur une Pile, toutes les tiles non choisies sont envoyées sur MiddlePile. Les tiles choisis par le joueur sont envoyés dans la “mains du joueur”, représentée par hand\_of\_player dans Bord, en attendant que le joueur choisisse où placer ses tiles dans son plateau.
- Si le joueur a choisi des tiles sur MiddlePile, on donne simplement les tiles dans la main du joueur et, si le joueur est le premier de la manche à récupérer des tiles dans MiddlePile, alors il récupère la tile malus.

Le joueur choisit ensuite où il dépose ses tiles. Cette information est transmise à Bord, qui gardait les tiles choisis dans hand\_of\_player. Ces tiles sont transférées dans la ligne choisie. A noter qu’ici “Ligne” désigne à la fois les lignes de PlayGrid, mais aussi la ligne malus. Par souci de simplification, les cas où le nombre de tiles dépasse la longueur des lignes ne sont pas décrits ici, mais cela est détaillé dans le diagramme d’activité.

Game passe ensuite au joueur suivant pour continuer la manche.

A la fin de la manche, on transfère dans Bord une tile de chaque ligne complétée sur le motif, on compte ensuite les points en conséquence. Pour finir la pioche (Bag) récupère toutes les tiles des plateaux du joueur, excepté les tiles du Pattern. Pour simplifier le diagramme, les classes présentes dans Bord ne sont pas affichées.

## 9/ PROJECTION DE L'IMPLÉMENTATION EN JAVA

Par la suite, nous avons décidé d'utiliser l'API **Swing** de la bibliothèque JFC. Cette API est en effet très performante et permet facilement de réaliser une interface graphique pour notre Projet. Ce choix s'avère également le plus judicieux, car nous avons déjà de nombreuses bases d'utilisation de Swing, que ce soit par d'anciens projets réalisés, ou alors par l'intermédiaire des cours que nous avons suivi tout au long de ce semestre.

Nous avons déjà bien avancé dans l'implémentation du programme, cependant il reste encore de nombreuses tâches à effectuer. Voici donc les principales tâches que nous avons identifiées :

- Finir les fonctions de bag
- Créer la fenêtre qui permet de choisir le nombre de joueurs (bord)
- Corriger l'erreur de calcul des points
- Mise en page du code déjà écrit (commentaires)
- Réaliser toute la partie view
- Coder les actions listener fonction qui gère les boutons

## 10/ MODIFICATIONS APRÈS L'IMPLÉMENTATION

Cette partie vient compléter le reste du rapport en apportant les modifications qui ont été faites durant la phase d'implémentation. Durant cette phase, nous avons procédé à plusieurs changement que ça soit dans la mécanique du jeu ou dans son architecture.

En ce qui concerne la mécanique, une règle a été modifiée afin de correspondre exactement à la version originale du jeu: à partir de la 2e manche d'une partie, le premier joueur de la manche est celui qui a pris la tile de malus à la manche précédente, autrement dit, c'est le premier joueur à avoir pris des tiles dans la Middle Pile. De plus, on précise ici que le nombre de Piles présentes dépendent du nombre de joueurs, en suivant la formule suivante :  $\text{nombreDePiles} = 2 * \text{nombreDeJoueurs} + 1$ .

Du côté de l'architecture du code, nous n'avons pas fait de changement majeur et nous sommes bien entendu restés sur l'architecture MVC choisie durant la phase de conception. Cependant, nous nous sommes rendus compte qu'il n'y avait pas assez de méthodes pour faire communiquer efficacement la vue et le modèle, nous avons donc rajouté plusieurs fonctions pour répondre à ce besoin.

Nous avons également ajouté un pop-up au moment de sélectionner la Line, cela permet d'agrandir le bord

du joueur qui est en train de jouer et donc d'avoir un affichage plus optimal. Cela nous permet aussi d'ajouter les boutons à côté de chaque ligne et de mettre le bouton malus.

De plus, quelques détails ont été ajoutés afin d'améliorer l'expérience du joueur et améliorer la cohérence par rapport au thème de l'UTBM. Un menu a été ajouté à la fin de la partie, il permet de choisir entre rejouer ou quitter le jeu. Ce menu annonce également les résultats en classant les joueurs selon leur score et en leur attribuant une lettre entre A et D. L'ajout de ces lettres permet de continuer les références au thème des UVs de l'UTBM comme le design des tiles ou encore les points qui sont nommés ECTS.