Université Denis Diderot - Paris VII

Projet De Programmation Orientée Objet
Réalisation d'un Moteur de jeu Azul

Réalisé par :

- Mechouar Fella
- Su Li-Fang

Groupe : 05

Table de Matières:

I. Introduction : 3
A. Sujet du Projet : 3
B. Objectifs : 3
II. Conception : 3
A. Variantes du Jeu : 4
III. Aspects Graphiques : 5
A. Digramme des classes de la partie graphique : 5
B. Echantillons de la Vue graphique du projet : 6
C. Les Classes Graphiques : 7
IV. Problèmes rencontrés : 12
V. Outils de travail : 13
A. Environnement de développement : 13
B. Outil de gestion du Projet : 14
C. Outil de réalisation des diagrammes de classes 14
D. Outil de réalisation des images graphiques : 14
VI. Perspectives et Améliorations futures pour le Projet 15
VII. Conclusion 15

I. Introduction :

Dans le cadre du module de programmation orientée Objet du premier semestre de l'année universitaire 2019 - 2020, et après plusieurs semaines d'apprentissage sur les différentes notions de la programmation orientée objet et du langage Java à travers les séances de cours, TPs et Travaux dirigés. On est amené à réaliser un projet dans lequel on doit démontrer la bonne acquisition des notions rencontrées durant le semestre.

A. Sujet du Projet :

Le projet consiste à réaliser un moteur de jeu offrant deux modes de jeu possibles avec quelques règles qui changent en passant d'un mode à un autre, permettant à un certain nombre de joueurs chacun son tour d'interagir avec les différents composants du jeu, à travers plusieurs manches, pour essayer de collecter le maximum de points et tenter de gagner la partie.

L'interaction avec le jeu peut se faire soit en mode textuel ou en mode graphique.

B. Objectifs :

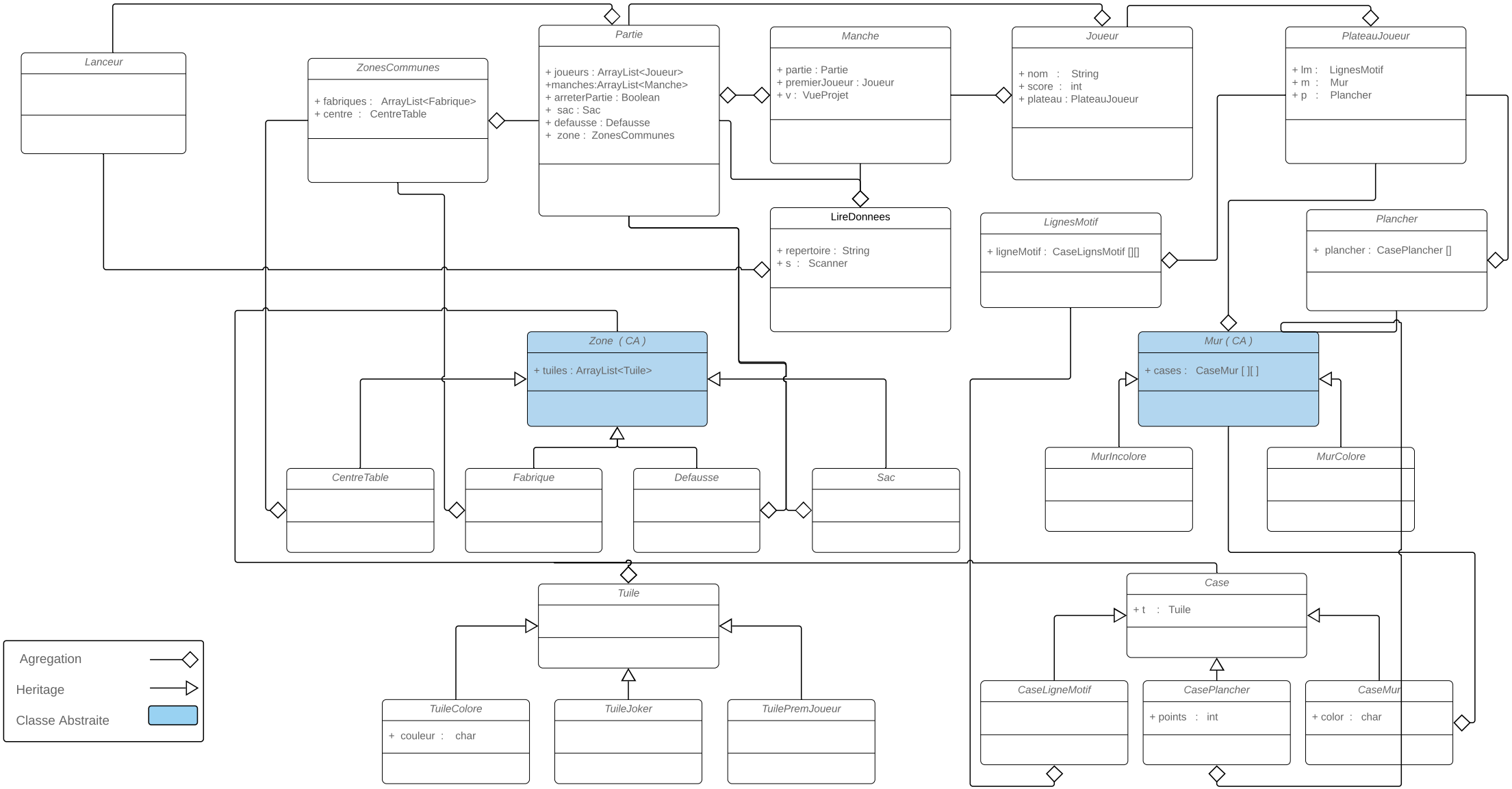
- Implémentation des règles du jeu tel que décrit dans l'énoncé du projet.
- Développement autour d'une architecture réutilisable, extensible et modulaire.
- Proposer deux modes de jeux différents avec des murs Colorés ou Incolores.
- Introduction des invariantes du jeu dans des endroits bien précis.
- Séparation des aspects graphiques du modèle de données.
- Permettre l'utilisation du jeu soit en mode graphique ou textuelle.

II. Conception :

Dans cette partie on représente la hiérarchie complète des classes de notre projet, ainsi que les relations entre elles. Les objets ont été identifiés soigneusement à partir de la description détaillée du projet, tout en exploitant et en respectant les principes de la POO (héritage, encapsulation, abstraction).

Le diagramme ci-dessous, montre le modèle de données sur lequel se base l'architecture de notre projet.

Diagramme du Modele de Données



A. Variantes du Jeu :

Après une bonne analyse des composantes et des règles du jeu, il était tour d'introduire les variantes qui permettent d'offrir deux modes de jeu différents aux utilisateurs, un mode avec des murs colorés et un autre avec des murs incolores, on expliquera ci-dessous plus précisément nos choix.

1. Classe Partie :

La classe partie est la classe pivot qui a travers le choix du mode de jeu introduit par l'utilisateur conserve dans une variable booléenne permet de passer l'information aux autres parties du jeu sur le mode, et donc assurer l'initialisation et le déroulement du jeu selon les règles du mode sélectionné.

2. Classe MurColore et MurIncolore :

A partir de ces deux classes on peut distinguer deux types de murs dans le jeu qui héritent de la classe abstraite Mur , les deux classes ont les mêmes méthodes , mais chacune les implémentent différemment , par exemple pour déposer Tuile de la classe MurColore on a juste à la déposer dans la case qui lui a été réservé auparavant, par contre pour le mur Incolore , il faudra distinguer le cas d'une tuile joker d'une tuile coloré , ainsi que pour le calcul des points bonus de la fin de partie ou on doit prendre en considération l'existence des tuiles joker pour effectuer les calculs .

3. Classe Tuile :

A partir de la classe Tuile dérivent trois classes TuilePremJoueur, TuileColore et TuileJoker cette séparation permet de faciliter la distinction des types des tuiles lors du déroulement du programme et traiter chacune selon son type. Par exemple lors de la lecture de choix de récupération des tuiles, si un joueur en mode incolore préfère prendre que des tuiles Joker il suffit donc de les distinguer dans les méthodes de la zone de récupération par un opérateur (instanceof).

4. Classe Manche :

La classe Manche construit une partie importante du jeu, c'est la classe dans laquelle sont gérées les règles les plus complexes du jeu dans chaque phase de la manche ainsi que les interactions avec les joueurs.

Dans cette classe on distingue en premier lieu le mode coloré du mode incolore à travers un booléen récupérée dans les paramètres de ses méthodes (les phases de la manche). Ensuite, à partir de la valeur de ce booléen on choisit le bloc du code qui traite les règles convenables avec le mode sélectionné et permet la meilleure gestion de tous les cas possibles lors de l'interaction avec les joueurs.

5. Classe Joueur :

La classe joueur, à son tour elle prend dans son constructeur un booléen indiquant le mode du jeu qui permet d'initialiser le plateau du joueur avec le type de mur convenable au mode sélectionné.

6. Classe Sac :

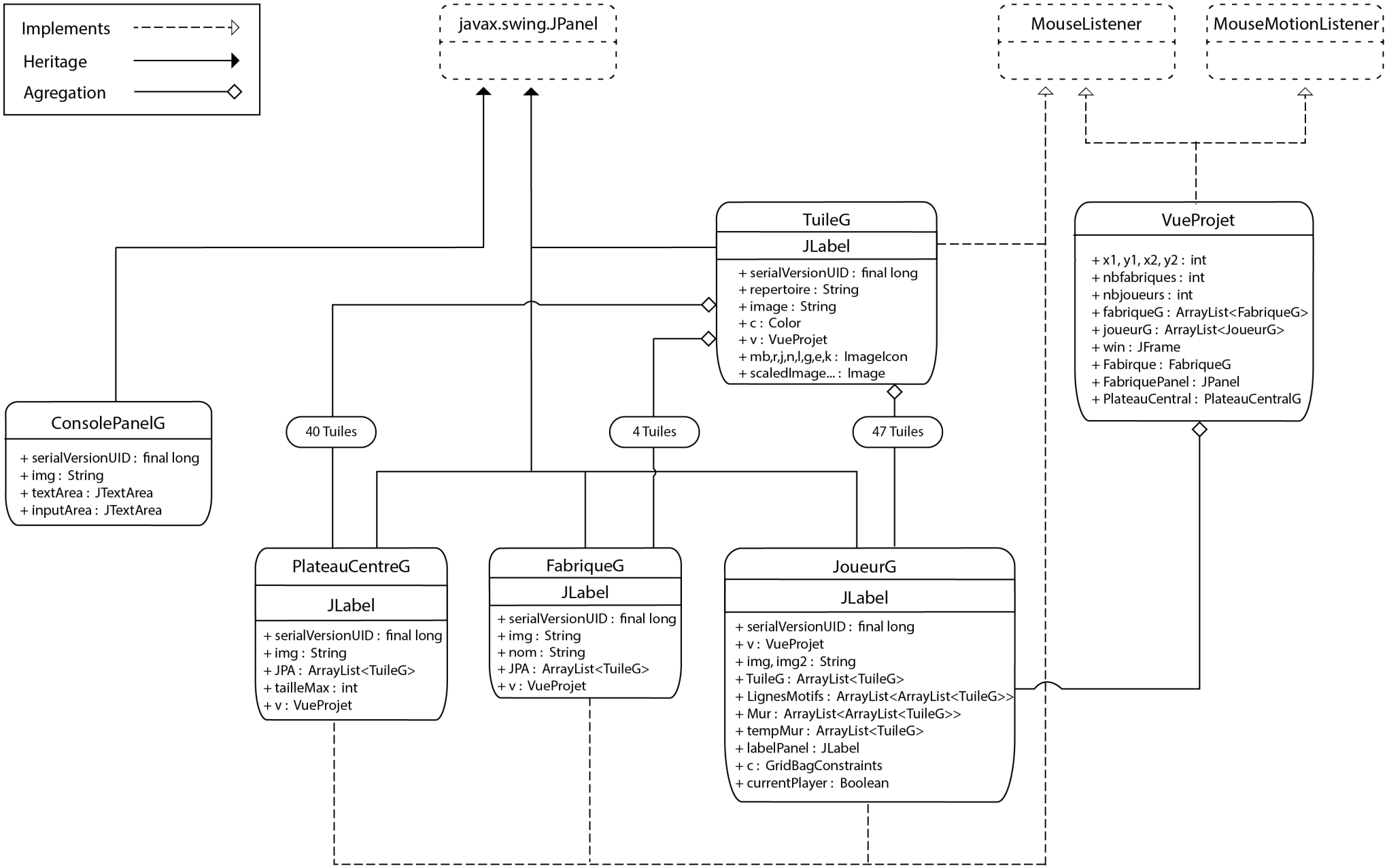
La classe sac, comporte deux constructeurs le premier permet d'initialiser le sac dans le mode coloré, et un deuxième pour initialiser le sac avec un mélange entre les tuiles colorées et les tuiles Joker en fonction du nombre de joueurs de la partie, le choix d'initialisation se fait au lancement de la partie en fonction de la valeur du booléen

Indiquant le mode du jeu.

III. Aspects Graphiques :

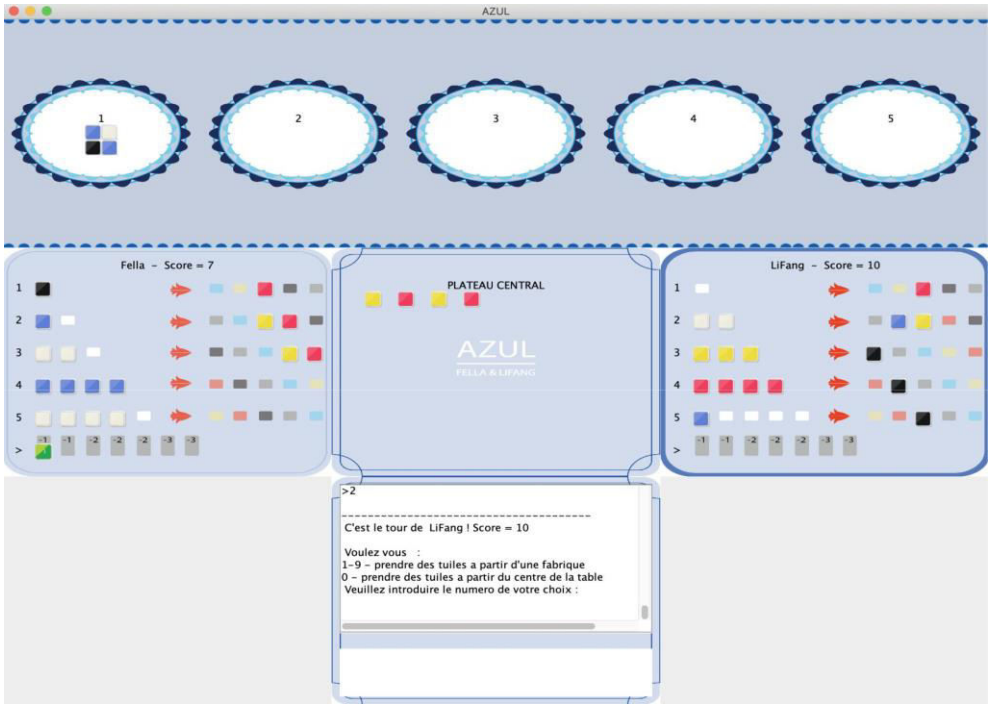
A. Digramme des classes de la partie graphique :

Diagramme des Classes Graphiques

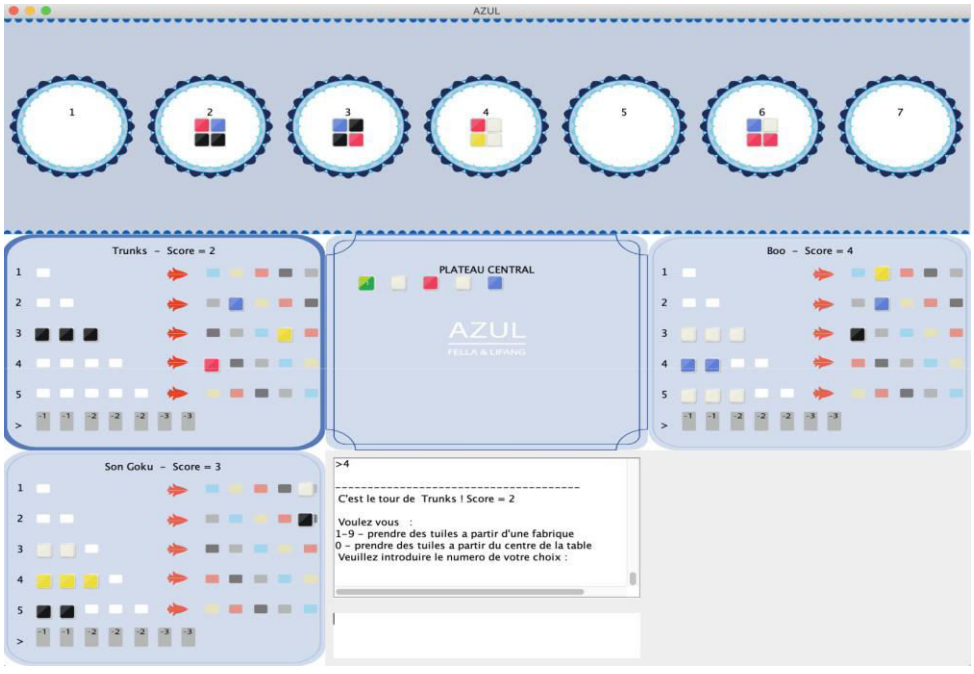


B. Echantillons de la Vue graphique du projet :

- Deux Joueurs, Cinq Fabriques



- Trois Joueurs, Sept Fabriques



C. Les Classes Graphiques :

1. VueProjet

- Cette classe orchestre toute l'interface graphique.

- Elle contient le JFrame principal, construit les JPanels correspondant à l'ensemble des éléments graphiques :

  (1) Un JPanel FabriquePanel contenant un nombre variable de FabriqueG, qui contiennent elles même 4 TuilesG.

  (2) Un JPanel PlateauJoueursPanel contenant :

  - Les 2 premiers JPanels JoueursG pour les 2 premiers joueurs.
    Par JoueurG on compte : 15 TuilesG pour les LignesMotifs, 7 TuilesG pour le Plancher et 25 TuilesG pour le Mur.
  - Le PlateauCentralG comptant 40 TuilesG sous format 5*5.

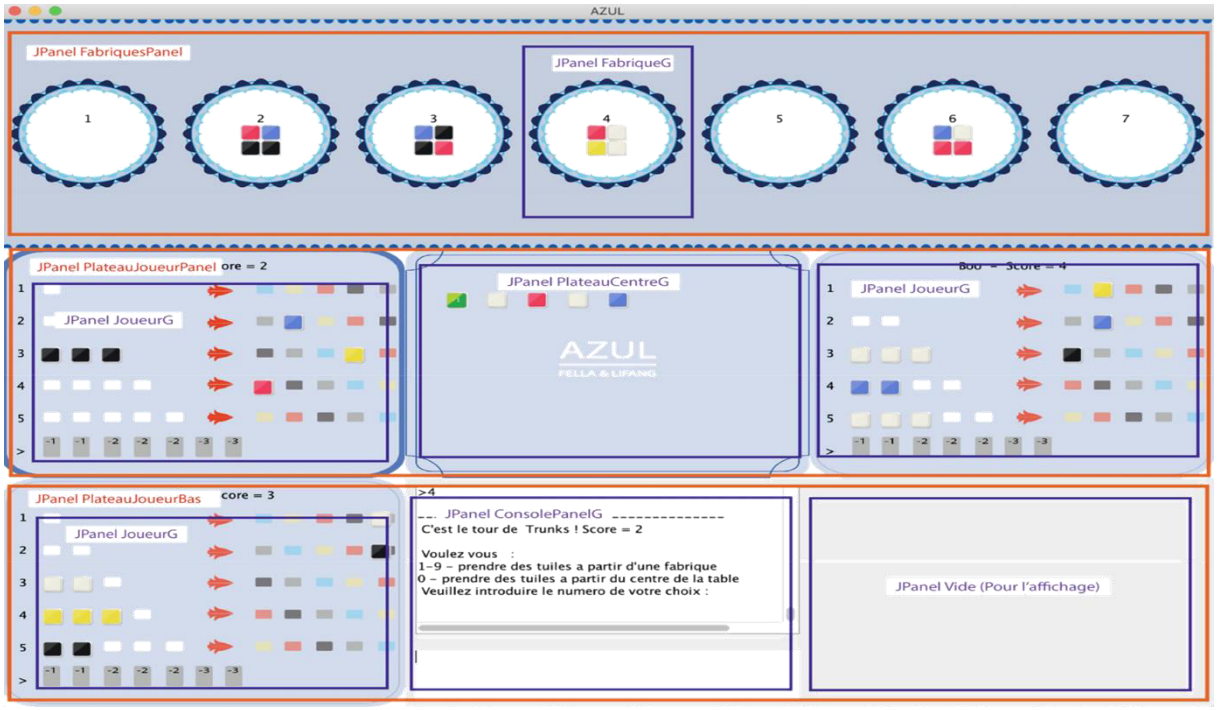
  (3) Un JPanel PlateauJoueurBas contenant :

  - Au centre Le ConsolePanelG pour l'interaction utilisateur.
  - Optionnellement 1 ou 2 autres plateaux de joueurs JPanel JoueurG.

- Elle comprend une méthode de « refresh » qui met à jour la vue avec l'ensemble des éléments du jeu.

- Elle crée un thread qui aura pour but de repeindre l'interface lors des changements / des clicks sur les différents JPanels.

- Cette classe implémente MouseListener et MouseMotionListener afin d'écouter les clicks souris, et de mettre en place du drag & drop dans une version future.

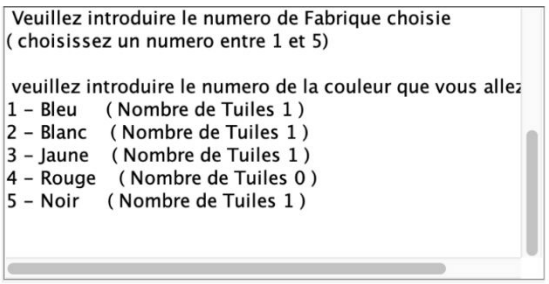


2. TuilesG :

- Cette classe étendant JPanel permet l'affichage des tuiles présentes selon leur couleur, ou leur type (Premier Joueur, Joker…).

- L'affichage des tuiles se fait sur certaines cases prédéfinies sur le plateau central, les Fabriques ainsi que sur le plateau des joueurs.

- Cette classe implémente MouseListener afin de positionner le numéro de la tuile dans l'input de la consolePanel lors des clicks. Cela renvoie le numéro de la tuile lors de la question :



3. PlateauCentralG

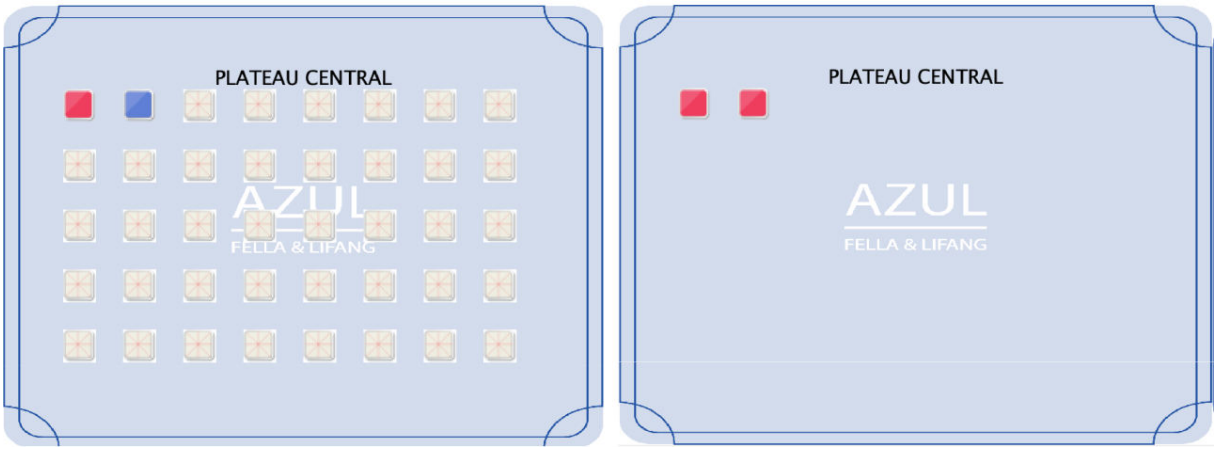
- Elle comprend Un JLabel présente le nom de l'élément.
- Les cases sont des TuileG centrées avec une BorderFactory.
- Elles ont représenté ici avec une image spécifique invisible dans une partie normale, sous format de tableau a 2 dimensions de 5 lignes 8 colonnes, mises en place avec un gridBagLayout.
- Lors du remplissage du plateau, les tuiles invisibles sont remplacées par des tuiles visibles respectant cette même disposition.
- Cette classe implémente MouseListener afin de positionner un texte spécifique dans l'input de la ConsolePanelG lors des clicks.

Cela répond « 0 » a la question :

```
Debut de la manche 1

---------------------------------------
C'est le tour de  Raditz ! Score = 0

Voulez vous   :
1-9 - prendre des tuiles a partir d'une fabrique
0 - prendre des tuiles a partir du centre de la table
Veuillez introduire le numero de votre choix :
```



4. FabriqueG

- Comme pour le plateau central et joueur, le layout se fait via un gridbagLayout permettant un bon centrage des tuiles et du JLabel présentant le numéro de fabrique.

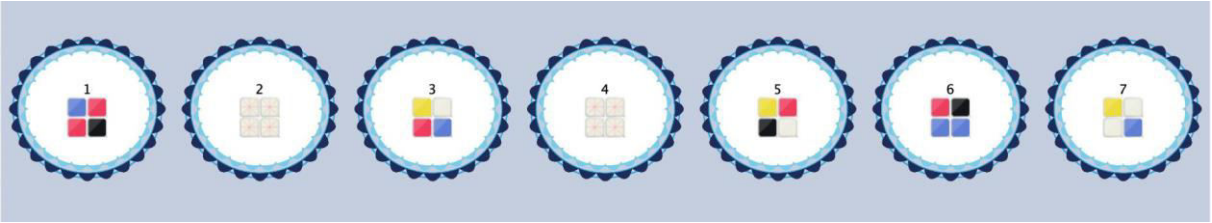
- Cette classe implémente MouseListener afin de set le numéro de la fabrique choisie dans l'input de la ConsolePanelG lors des clicks, ce à 2 reprises via un thread.

  Cela répond « 1-9 » à la question :

  « 1-9 - prendre des tuiles a partir d'une fabrique »

  Par un thread, La réponse est doublée et identique à la question :

  « Veuillez introduire le numero de Fabrique choisie"



5. JoueurG :

- Un JLabel présente en haut le nom du joueur ainsi que son score, il sera mis à jour pendant la partie.

- Les cases des lignes motifs, du plancher ainsi que du mur sont des TuileG ayant des positions définies précisément avec un gridBagLayout, en accord avec le background image. La disposition est conservée par rapport à l'image en cas de redimensionnement de la fenêtre.

- Les background images varient selon le mode Mur Colore ou Mur Incolore.

- Cette classe implémente MouseListener afin de set le numéro de la ligne motif choisie dans l'input de la ConsolePanelG lors des clicks, ce à 2 reprises via un thread.

Cela répond à la question :

```
Voulez vous deposer vos tuiles
1-5 - Sur Une Ligne Motif
0- Sur le plancher
```

Et cette question :

```
Veuillez introduire le numero de la ligne
( Introduisez un numero entre 1 et 5 )
```

- Selon le joueur qui joue, la background image change, permettant de visualiser rapidement le joueur actuel, et une évolution simple des graphiques.

Image du plateau joueur avec mur coloré :

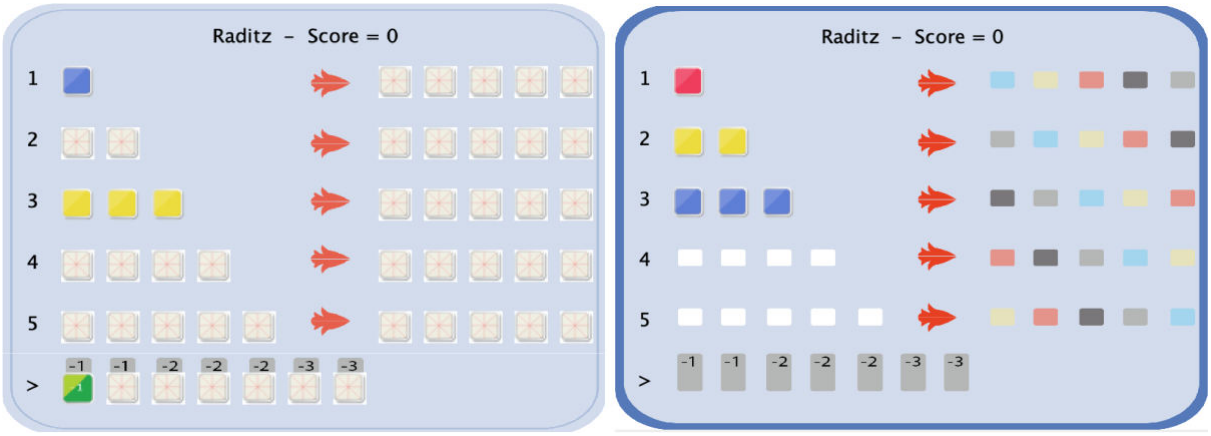
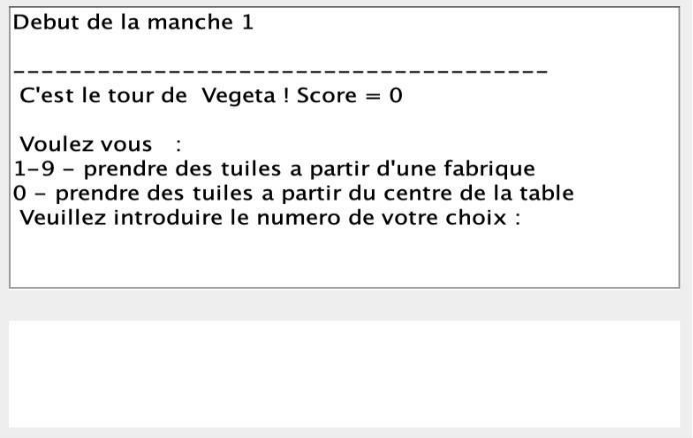


Image du plateau joueur avec mur incolore :

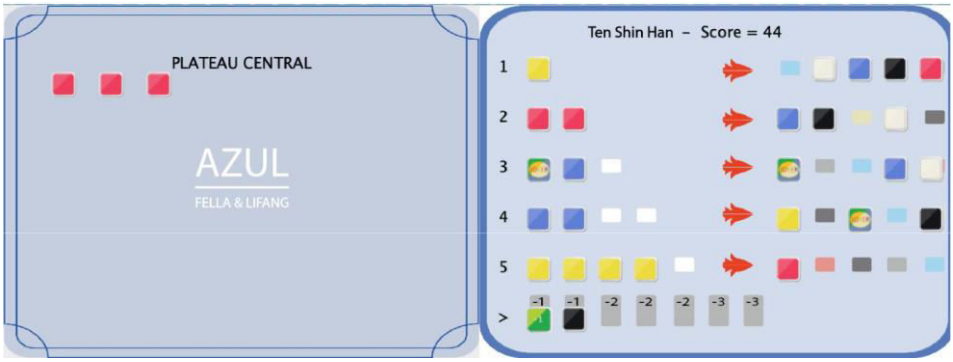
6. ConsolePanelG :

- Cette classe contient 2 JTextArea permettant l'interaction avec les utilisateurs.
- La première renvoie le retour de la console Eclipse, et donc les choix à faire.
- La 2e est une case d'input ou entrer les réponses aux choix proposés.
- Chaque mouseclick sur les éléments vus précédemment rempli cette case d'input, et l'information est ensuite traitée

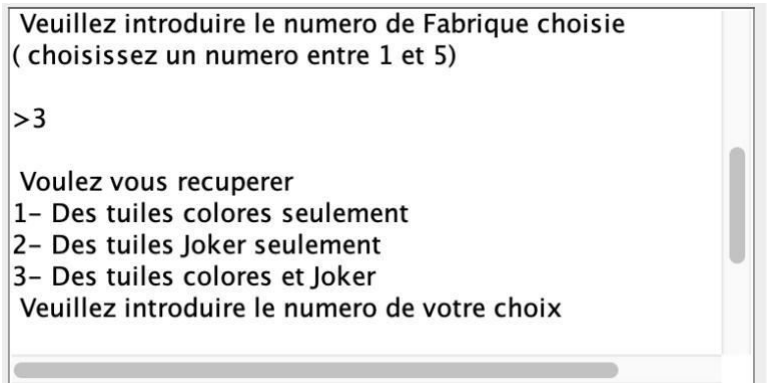


IV. Problèmes rencontrés :

- Fin du jeu avec la tuile Joker :
  Dans ce type de situation, le joueur Ten Shin Han est bloqué : Il doit poser la tuile jaune de la ligne 1 colonne 1 car aucune autre place n'est pas disponible, mais il y a déjà un jaune en ligne 4, donc il est totalement bloqué.



- Gestion du choix des types des Tuiles en mode coloré “Par Click”, pour le moment l’utilisateur doit entrer l’input manuellement dans la JTextArea .



V. Outils de travail :

A. Environnement de développement :

Eclipse est un projet, décliné et organisé en un ensemble de sous-projets de développements logiciels, de la fondation Eclipse visant à développer un environnement de production de logiciels libre qui soit extensible, universel et polyvalent, en s'appuyant principalement sur Java.

B. Outil de gestion du Projet :

GitLab est un logiciel libre de forge basé sur git proposant les fonctionnalités de wiki, un système de suivi des bugs, l'intégration continue et la livraison continue.

LIEN DE NOTRE DEPOT GITLAB :
https://gaufre.informatique.univ-paris-diderot.fr/fel_mch/rojo.git

C. Outil de réalisation des diagrammes de classes

Lucidchart est un service d'informatique en nuage qui permet de travailler collaborativement afin de créer des ordinogrammes, organigrammes, diagrammes UML, cartes heuristiques, schémas de classification, cartes conceptuelles, diagramme fonctionnel[1] et d'autres types de diagrammes.

D. Outil de réalisation des images graphiques :

Adobe Illustrator est un logiciel de création graphique vectorielle. Il fait partie de la gamme Adobe, peut être utilisé indépendamment ou en complément de Photoshop, et offre des outils de dessin vectoriel puissants.

VI. Perspectives et Améliorations futures pour le Projet

1. Ajouter une Intelligence artificielle avec analyse prédictive des divers scénarios de partie possibles en fonction des tuiles positionnées, et restantes dans le sac et la défausse.

2. Implémenter le MouseMotionListener afin de faire drag and drop des tuiles entre les fabriques/plateau central et les lignes Motifs/lancher.

3. Ajouter du son lors des actions.

VII. Conclusion

A travers ce projet, on a pu avoir une vue concrète sur l'application des principes de la programmation orientée objet, qui nous permet de modéliser n'importe quelle forme de problème en plusieurs objets ayant connexion entre eux à travers des relations, d'héritage, implémentation et agrégation. Pour enfin, pouvoir exploiter la hiérarchie établie et mettre en œuvre une solution fonctionnelle permettant de répondre aux besoins des utilisateurs et atteindre les objectifs tracés .