CHRISTOPHE AUREGLIA

TANG Thanh Long

Rapport de projet

Jeu de Domino & Carcassonne

Licence 2 informatique

2022-2023

Projet informatique en POO

REMERCIEMENTS

Ressources utilisées :

Photos utilisées:

Gone Gaming: Anatomy of a Game: Carcassonne, Part Two: Balance & Tiles (boredgamegeeks.blogspot.com)

http://boredgamegeeks.blogspot.com/2006/04/anatomy-of-game-carcassonne-part-two.html

Generic Methods (The Java™ Tutorials > Learning the Java Language > Generics (Updated)) (oracle.com)

Nimbus Look and Feel (The Java™ Tutorials > Creating a GUI With Swing > Modifying the Look and Feel) (oracle.com)

How to Use Color Choosers (The Java™ Tutorials > Creating a GUI With Swing > Using Swing Components) (oracle.com)

SOMMAIRE

I. Introduction

Le jeu de DOMINO est un célèbre jeu de réflexion joué à plusieurs, se composant des tuiles sont à leur disposition dans un sac opaque mis en commun. Les tuiles sont des carrés où chaque côté porte trois chiffres. Au départ, une tuile (prise au hasard) est posée face visible. Chaque joueur à son tour va en piocher une dans le sac, et la déposer sur la table, à côté des autres (avec l'orientation de son choix), pourvu qu'il trouve une correspondance bord à bord avec celles qui y sont déjà. S'il ne le peut pas

il la défausse (sans la remettre dans le sac) et passe son tour. Lorsqu'un joueur pose une nouvelle tuile, il marque alors un certain nombre de points : le total des chiffres en contact avec ceux des tuiles voisines.

De la même façon que pour les dominos carrés, le jeu se joue à plusieurs joueurs placés en cercle autour d'une table. Chacun à leur tour, les joueurs piochent une tuile dans un sac puis la posent sur la table. La correspondance ne se fait plus avec des chiffres, mais avec des éléments graphiques : villes, champs, routes, etc. L'image ci-dessous vous donne une idée du paysage qui peut se former. Chaque joueur, après avoir placé sa tuile peut ensuite déposer un pion (partisan) de sa couleur sur un élément du paysage de cette tuile. En nombre limité, les partisans permettront à leur propriétaire de marquer des points de victoire plus tard. Par exemple, un partisan posé sur : — une ville, rapporte des points fonction de la surface de la ville ; — une route, rapporte des points en fonction de la longueur de la route ; — une abbaye, rapporte des points en fonction du nombre de tuiles adjacentes, etc.

Chacun des membres de ce groupe, Christophe Aureglia et TANG Thanh Long, appréciant le jeu de Domino et Carcassonne mais ne le connaissant que sous sa forme classique du plateau de bois. Il nous a donc été demandé de programmer un jeu de Domino en java en se basant sur les règles internationales puis ensuite développer Carcassonne et d'exploiter leurs points communs pour réaliser des prototypes pour les deux jeux, en réutilisant le maximum d'éléments.

Afin d'y parvenir, il nous a fallu analyser tout d'abord le sujet à l'aide de diagrammes UML de cas d'utilisation et de classes et nous organiser en nous répartissant les tâches à accomplir, du jeu sur console au jeu graphique voir partie analyse et conception, puis nous sommes passées à l'implémentation du programme à proprement parler, la description des algorithmes se trouve donc dans Réalisation et implémentation, enfin nous avons bien évidement rencontrés beaucoup d'obstacle qui seront listés dans Problèmes rencontrés et solutions apportées.

II. Analyse et conception

Analyse du travail à réaliser :

Un des objectifs principaux de ce projet a été de nous habituer au travail en groupe et aux notions de projet. Organiser son temps, établir les tâches sous la forme d'un diagramme de Gantt ou encore nous obliger à respecter notre cahier de charge.

Nous avons ensuite et dans un premier temps travaillé ensemble sur les premières classes à établir (notre premier diagramme de classe en UML est donné en annexe anx1) afin d'établir un socle commun au sein de notre groupe en codant en dure

Voici notre planning prévisionnel :

TACHE									
Analyse du sujet	Les deux								
Etablissement des différents diagrammes			Les deux						
Rédaction des fonctions principales				Thanh					
Codage du début de la classe plateau				Christophe					
Codage des autres classes					Les deux				
Codage des fonctions pour des domino & Carcassonne						Les deux			
Tests			Christophe						
Codage des classes externes						Les deux			
Regroupement partie graphique/ Plateau							Les deux		

Mise en place d’une IA	XXX							
Dernier tests						Thanh		
Test sur plusieurs systèmes d’exploitation								Christophe
Rédaction rapport								Les deux

Notre but était de commencer à élaborer un programme sous forme console en déterminant les algorithmes qui nous donneraient les choix de déplacements des cartes de domino tout en respectant les règles du sujet.

Il nous a également été demandé d'utiliser java AWT et SWING comme langage informatique pour programmer ce jeu. Les avantages fournis par ce langage sont nombreux. La disparition des pointeurs nous a notamment simplifié la tâche et nous a permis de programmer plus vite sans rencontrer de problème.

Par la suite, il nous a fallu définir les étapes à accomplir afin de mener à bien ce projet. Il a donc été décidé d'implémenter premièrement une version console du jeu de Domino, pour après programmer une interface graphique, un moyen de pouvoir quitter et sauvegarder une partie en cours de jeu, et enfin de créer une intelligence artificielle capable de jouer contre un joueur seul. Après plusieurs premiers tests et après avoir imaginé de nombreux cas de figures, nous sommes arrivés à une configuration du jeu à travers deux schémas :

Schéma des cas d'utilisation :

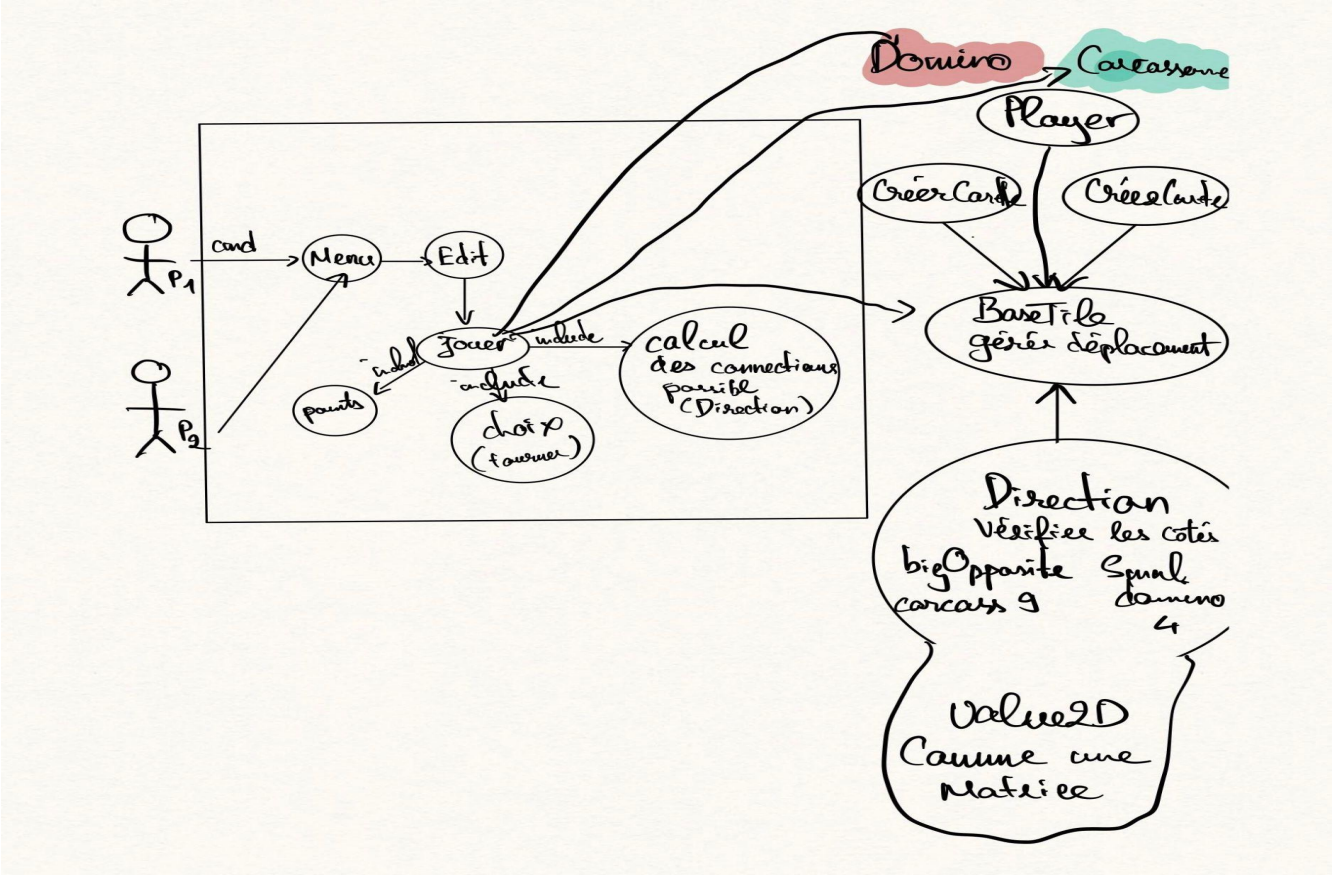
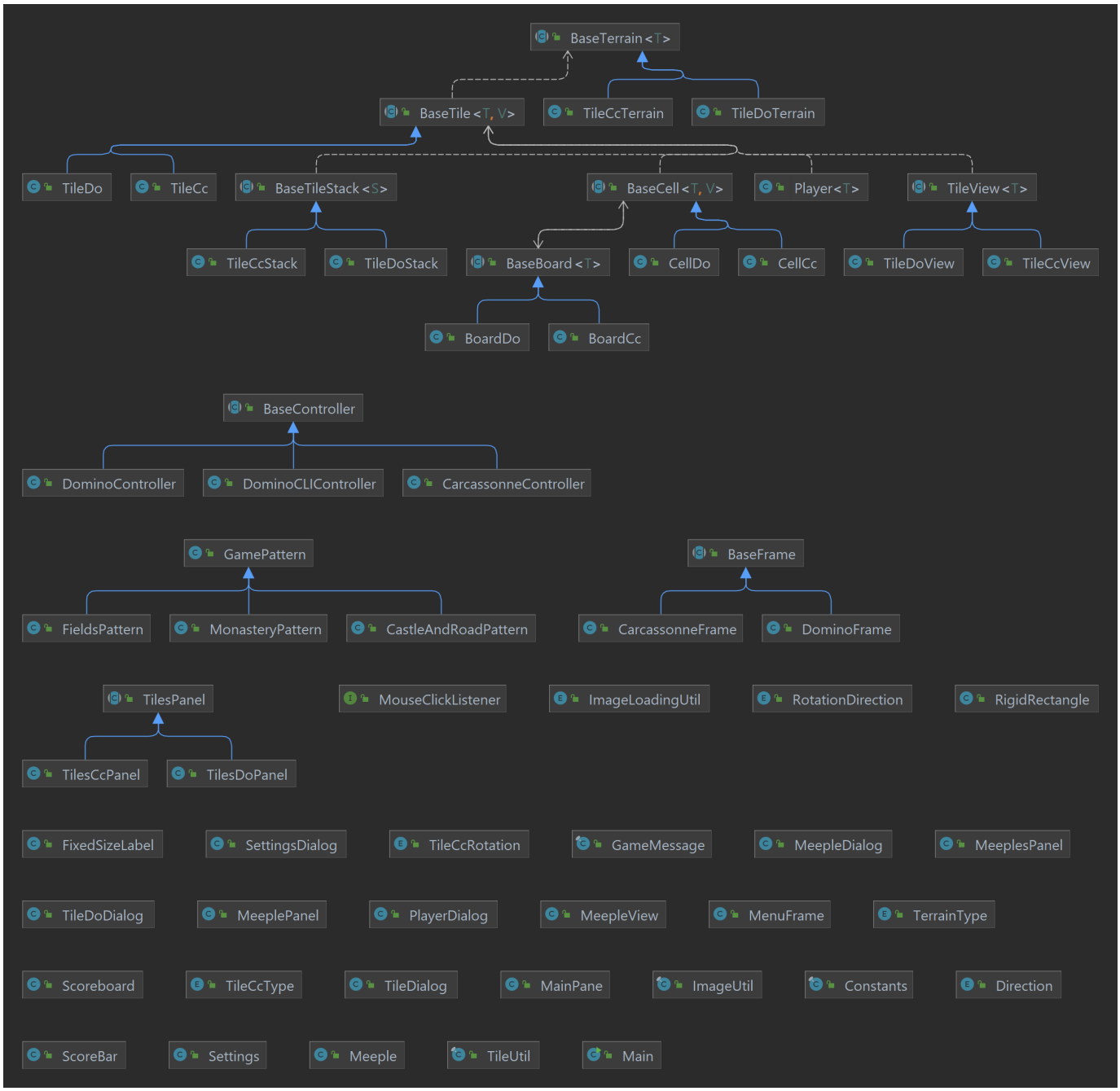


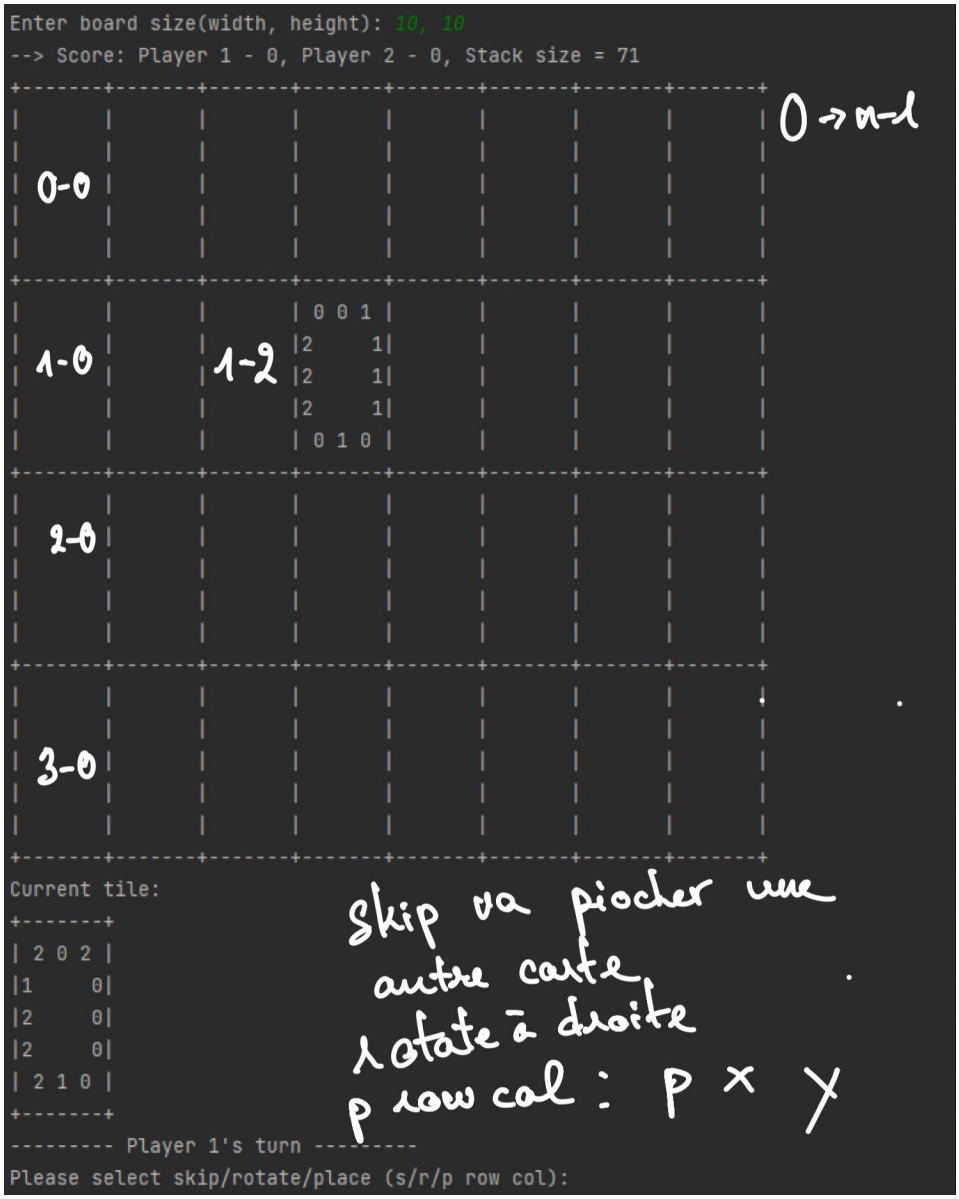
Diagramme de classe, structurant le projet:



III. Réalisation et implémentation

- **Implémentation des mécaniques de jeu:**

Terminal affichage :



Structure du projet :

Il y a trois principales qui permettent au jeu de se dérouler de différentes manières (il y a donc 3 class princiles qui sont DominoController, CarcassonneController, DominoCLIController qui ont BaseController en Extends. La classe PartieConsole fait marcher le jeu sous console et la partie graphique de Domino et Carcassonne ont le même type d'interface (demontration dans Implementation de l'interface graphique).

BaseTile :

La classe indispensable à notre programme. La classe possède un champ membre : un objet Map appelé terrain. Cette carte stocke le mappage des directions vers les objets de type T. La carte est initialisée dans le constructeur de la classe BaseTerrain à l'aide de l'expression new HashMap<>(). BaseTerrain fournit une implémentation de base pour un terrain qui peut être tourné et qui possède une carte de directions vers des objets de type T. Il s'agit d'une classe abstraite, ce qui signifie qu'elle ne peut pas être instanciée seule et doit être sous-classée afin de créer une implémentation concrète du terrain. Si les types de terrain sont égaux, la méthode renvoie true, indiquant que les deux tuiles peuvent être connectées. Si les types de terrain ne sont pas égaux, la méthode renvoie la false, indiquant que les deux tuiles ne peuvent pas être connectées. Par example, sur le plateau il y a déjà un, la CanConnectTo va prendre la direction de la carte, par exemple le domino qui est déjà sur le XXX, si nous voulons coller le côté nord au côté sud d'une carte. Il vérifiera les égalités pour le côté opposé de notre domino en main. Pareil pour Carcassonne mais 9 côtés/direction

Calculer score pour Carcassonne : GamePattern

Disburse : La méthode disburse est utilisée pour débourser, ou distribuer, le score d'un motif. Elle vérifie d'abord si le motif est complet en utilisant le champ boolean complet. Si le motif est complet, la méthode appelle la méthode distributePatternScore pour distribuer le score, puis retire tous les meeples des tuiles où ils sont placés à l'aide de la méthode removeMeeple. Elle efface également la carte involvedPlayers, qui stocke les joueurs impliqués dans le motif et leurs scores correspondants.

Fordisburse : La méthode forceDisburse est similaire à la méthode disburse, mais elle ne vérifie pas si le motif est complet. Au lieu de cela, elle appelle toujours la méthode distributePatternScore pour distribuer le score, même si le motif n'est pas complet. Donc les deux joueurs ont quand même les points.

Constants : L'une des class principales du jeux qui pourra definir la taille du plateau ainsi que de declarer le nombre minimum & maximum de joueurs.

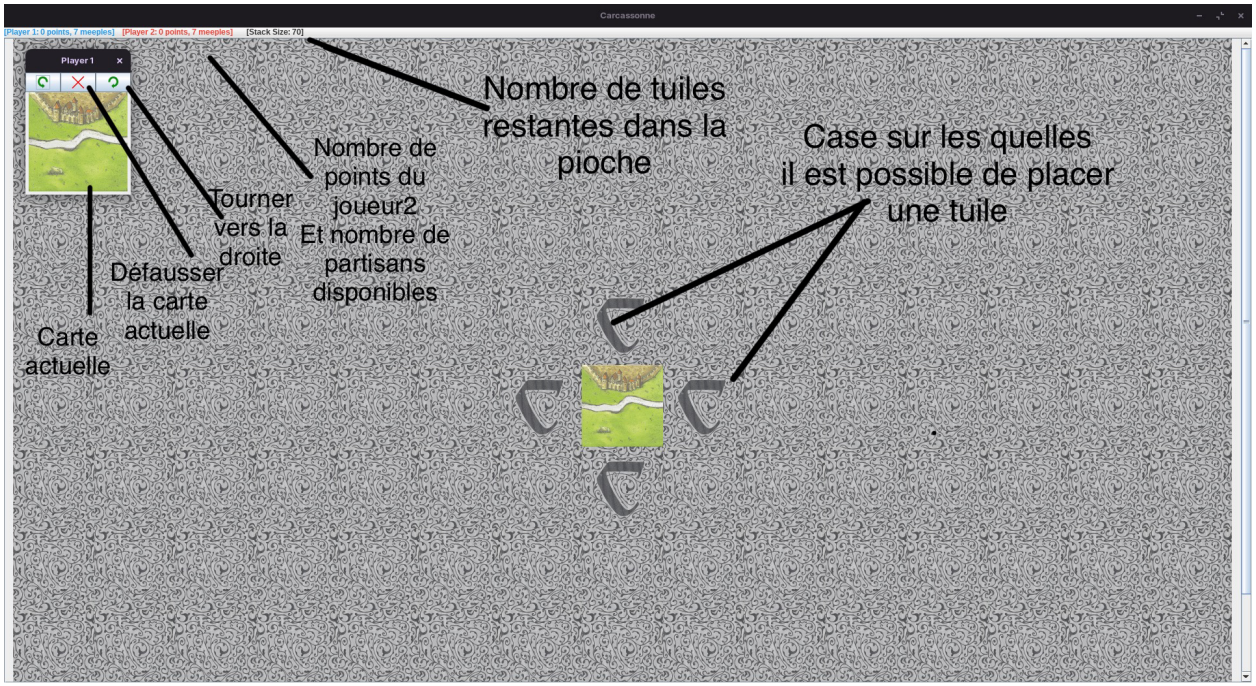
Fenêtre:

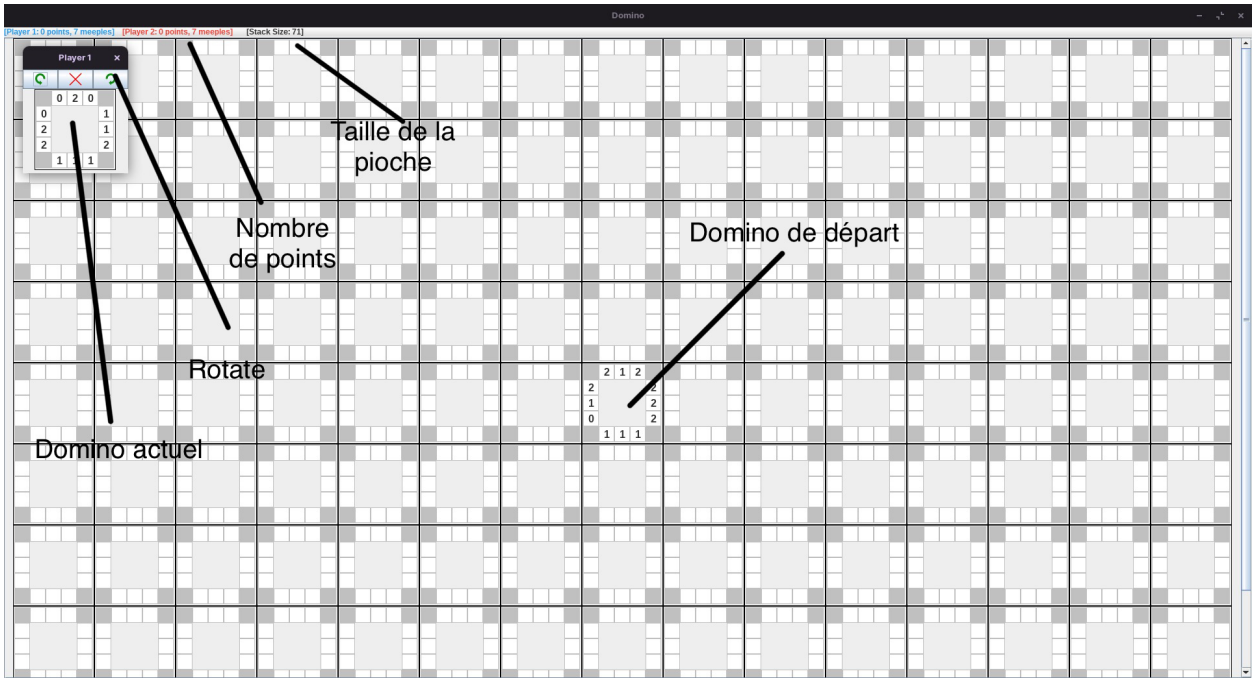
La classe indispensable de l'interface graphique. Elle hérite de Jframe est implémente les interfaces ActionListener et KeyListener. C'est donc ici aussi que les interactions entre la souris ou le clavier et notre interface graphique pourront avoir lieu. Elle est ordonnée grâce à des Jpanel et des Layout.

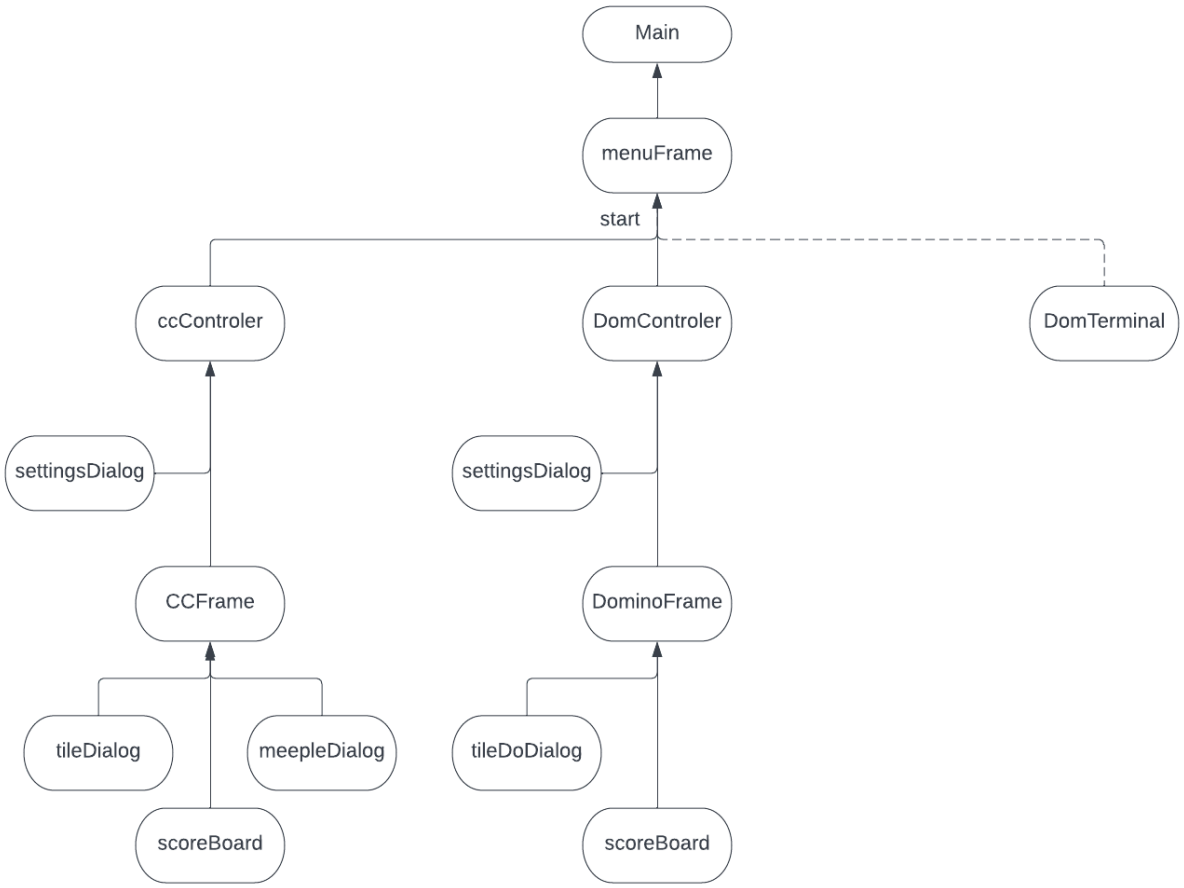
PartieView :

Cette classe a les mêmes fonctionnalités que la classe partieConsole mais sous forme graphique, donc dans une fenêtre (pour Domino).

- **<u>Implémentation de l'interface graphique</u>**




Domino
[Player 1: 0 points, 7 meeples]
[Player 2: 0 points, 7 meeples]
[Stack Size: 71]
Player 1
Taille de la pioche
Nombre de points
Rotate
Domino actuel
Domino de départ


Main
menuFrame
start
ccControler
DomControler
DomTerminal
settingsDialog
settingsDialog
CCFrame
DominoFrame
tileDialog
meepleDialog
tileDoDialog
scoreBoard
scoreBoard

Problèmes rencontrés et solutions apportées

- ***Les Ressources (Images)***

Un autre problème notamment lié à l'interface graphique a été l'utilisation d'images. Donc si on lançait le dossier avec Intelji, il n'y avait pas de bug. Mais quand on lance avec eclipse ou vscode en utilisant des commandes, on reçoit des erreurs car les tuiles ne pouvaient pas récupérer des images. Pour régler ce problème, on a appris que c'est seulement Intelji qui met en place le dossier ressources dans path.
Afin de lancer le jeu, on doit déplacer toutes les photos dans le dossier src car pour compiler, faut qu'on soit dans les

- ***Implémentation du MVC***

C'est le point le plus important du projet car une fois avoir la structure mvc, on doit se poser la question comment relier la partie :

- util qui sert à définir le plateau, le nombre de joueur et les partisan pour chaque joueur
- view qui gère toute les Dialog par exemple le petit onglet pour les cartes (Domino ou Carcassonne)
- Model qui va générer les tuiles ainsi les cartes etc

Pour régler ce problème, on a décidé de rajouter un dossier Controller où il y a la class BaseController qui contient des méthodes que Domino et Carcassonne peuvent réutiliser ainsi de tout lancer (Domino Terminal,... ensuite un Dialog Setting pour pouvoir choisir la couleur des players) ainsi de partager aussi les fonctions importantes comme placement des tuiles et abandonner la tuile qu'on a dans la main.

- ***Relancer l'application/bouton retour***

Une fois l'application "finie" nous nous sommes rendu compte qu'il manquait des boutons retour en arrière, et que ajouter des boutons au milieu de notre jeu n'était pas une manière de faire très élégante, une fois l'interface structuré, pour schématiser nous avons donc décidé de remplacer "Exit on close" par "dispose on close"

Pistes d’extensions pas encore implémentées et cahier des charges

CAHIER DES CHARGES DOMINO

- langage utilisé AWT et SWING
- plusieurs joueurs 2-5
- tuile carrée avec ou chaque cote porte 3 chiffre de 0 à 2 (ou 0 - n -> peut être changer dans les TileDoTerrain dans Tile Dom)
- Tuile pris au hasard posée au centre
- Pioche dans le sac
- Orientation au choix. Rotate droite gauche correctement
- Marque des points quand on pose le domino. Donc c’est la somme des chiffres en contact avec d’autre tuile
- à régler -> On ne passe pas notre tour quand on défausse car sinon trop de temps sans jouer
- autre “bug” quand on joue au domino et qu’on tourne le domino le “sens de lecture” reste le même, par exemple : si en haut (nord) on a la sequence {1,0,0} en faisant 2 rotates en bas nous aurons aussi la séquence {1,0,0} ROTATE LEFT RIGHT FUNCTION

DOMINO TERMINAL

Mode de jeu texte pour le domino avec toutes les fonctionnalités comme sauter, tourner, placer en respectant les coordonnées.

CAHIER DES CHARGES CARCASSONNE

- Tuile pris au hasard posée au centre
- pioche dans le sac
- orientation au choix
- marque et compte les points
- contrainte de placement d’un partisan respectée
- à la différence de Domino quand l’on défausse une carte, la pioche ne se vide pas
- tour ne passe pas quand on défausse

PISTE EXTENSIONS

- Sélectionner “humain” ou “IA” pour chaque joueur
- Affiche le vainqueur à la fin de la partie
- pop-up de tips/tutoriel