Architecture du projet:

L'architecture du projet est faite en pyramide, les 2 classes mère sont board(le terrain du jeu) et player(le joueurs et ses possesions).
Les champs de board sontr le nombre de joueur afin d'initialiser correctement le jeu, le total des jetons restants (la banque), les quatres nobles en jeu(classe TileBoardLine), les 3 niveaux decks(classe deck) et les 12 cartes sur le terrains classés dans une map selon leurs niveau(classe CardBoardLine). Cette classe est censé représenter tout ce qu'il y a sur le terrain et ses méthode manipulent ces éléments(ex : ajouter ou retirer des jetons). La classe player possède un qui permet l'identification du joueur via son nom, un champ qui donne les jetons bonus (grace aux cartes) et les jetons possèdé par le joueur , un champ qui represente les cartes achetés et les cartes réservés et un champs qui indique le prestige total.
La classe GameAction permet tous les mécanisme du jeu(ex : acheter des cartes ou piocher des jetons), elle contient uniquement des méthode static qui agissent sur board et player ou qui vérifie un état de l'un ou l'autre.

Partie Graphique :
Pour commencer la partie graphique il nous d'abord fallut comprendre le fonctionnement de Zen5. Zen5 utilise un context qui nous permet d'afficher et de récupérer des saisie il est donc nécessaire de le passer en parametre de chaque fonction dessinant sur la fenetre.

Changements soutenance beta:

Suite à la soutenance beta nous avons retirés les classes inutiles comme celle qui contenait une pile de jetons ou les deck noble nous avons également retirés les méthodes inutiles (grace au raccourcie ctrl shift g).
Nous avons ajouté la classe GameAction qui avec des méthodes

static implement les fonctionalités du jeu. Et enfin modifié la manière de représenter le prix des cartes(directement pas une tokenBank).La gestion d’erreur des saisies a aussi été modifié pour redemander une saisie tant qu’elle soit mauvaise.