# Cahier des charges :

## Descriptif du projet :

Le produit fini sera un jeu de Scrabble multijoueur (2 joueurs maximum).

## Règle :

Le jeu se déroule en tour par tour et pour désigner le premier joueur, chaque joueur tire une lettre, celui possédant la plus proche du « A » tire en premier les 7 lettres et commence. Le but est de former des mots avec ses lettres pour marquer un certain nombre de points (cf. plus loin).

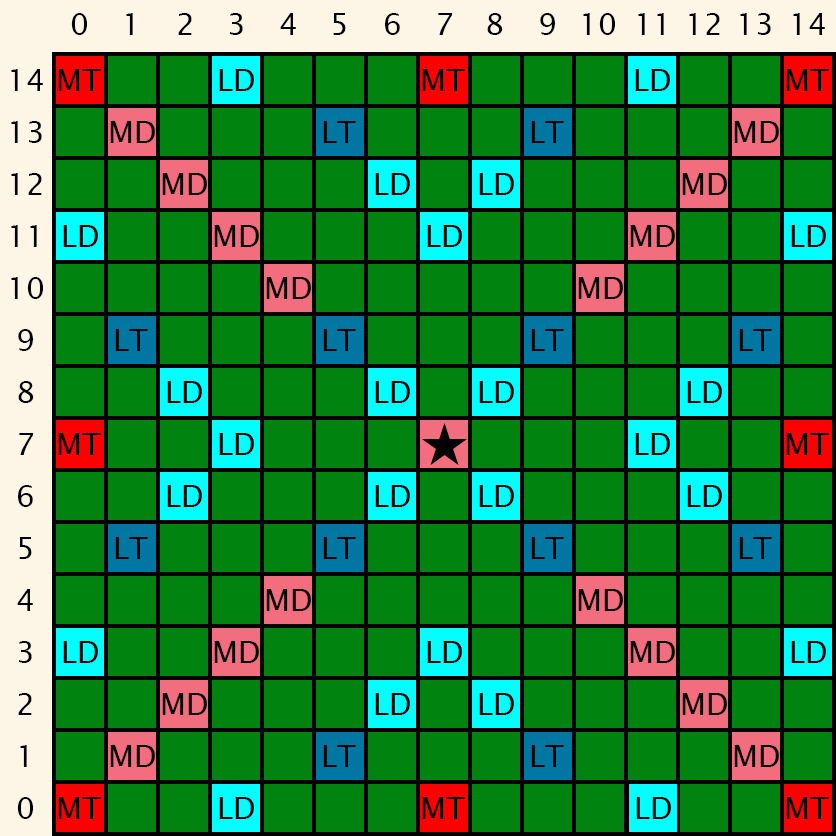
Les points sont calculés en fonction de la valeur de chaque lettre et de possible bonus dispersé sur le plateau. Le jeu se termine lorsqu’un joueur n’a plus de lettre ou lorsqu’il lui est impossible de former un mot.

## Implémentation :

### Le jeu sera composé de :

* Un plateau de jeu (15x15) avec certaines cases bonus dispersées sur ce dernier suivant le pattern suivant :

Le premier mot doit être posé sur l’étoile au centre du plateau.



* Les joueurs commenceront avec 7 lettres aléatoirement choisies dans la sélection suivante :
  + *0 point* : **Joker** × 2
  + *1 point* : **E** ×15, **A** ×9, **I** ×8, **N** ×6, **O** ×6, **R** ×6, **S** ×6, **T** ×6, **U** ×6, **L** ×5
  + *2 points* : **D** ×3, **M** × 3, **G** ×2
  + *3 points* : **B** × 2, **C** × 2, **P** × 2
  + *4 points* : **F** ×2, **H** ×2, **V** ×2
  + *8 points* : **J** × 1, **Q** × 1
  + *10 points* : **K** ×1, **W** ×1, **X** ×1, **Y** ×1, **Z** ×1

Lorsqu’une lettre est piochée, son nombre d’instances dans « le sac » est décrémenté.

Chaque lettre correspond à un certain nombre de points (voir liste ci-dessus).

Lors de son tour, un joueur peut, soit poser un mot sur le plateau (de haut en bas et de gauche à droite), soit remettre l’entièreté ou une partie lettres dans « le sac », soit passer son tour.

Lorsque le joueur réussit à poser toutes ses lettres, il réalise un « Scrabble » et marque immédiatement 50 points bonus.

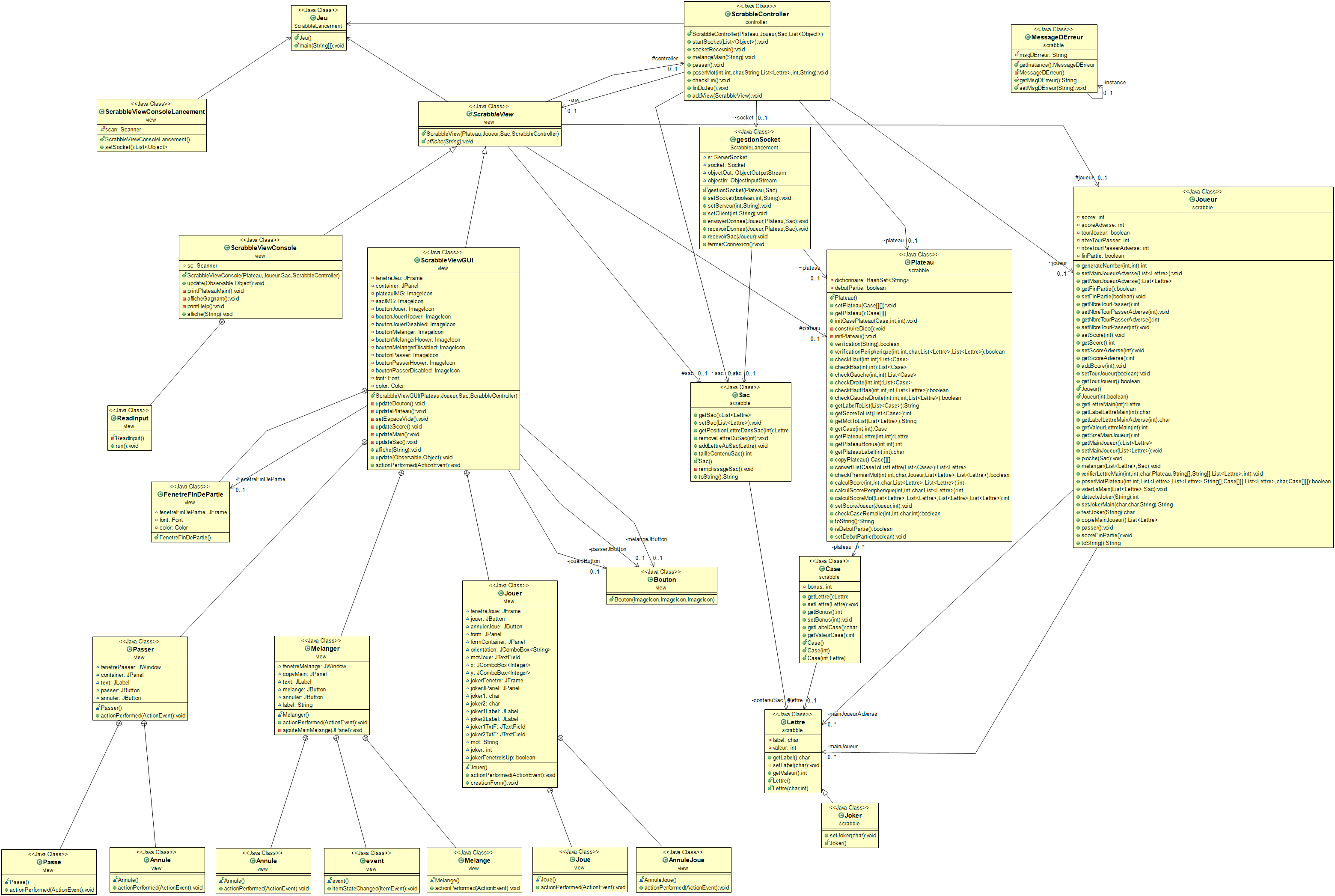
Les points sont comptés en fonction de la valeur des lettres (voir liste ci-dessus) et, éventuellement des bonus du plateau. Un bonus est activé si le mot est posé sur ce dernier. Un mot peut être posé sur plusieurs bonus, les effets sont cumulables mais les bonus ne peuvent être utilisés qu’une seule fois.

Un mot doit utiliser une lettre d’un mot déjà présent sur le plateau ou le complétant, cette lettre est comptée dans les points.

Si un mot en colle un autre, les points de tous les nouveaux mots sont comptés.

Les mots sont vérifiés grâce à un dictionnaire, les noms propres, les sigles et abréviations, les expressions composées et les symboles chimiques sont invalides à contrario, les verbes conjugués et les onomatopées sont autorisés. Les nouveaux mots formés au placement d’un mot sont aussi vérifiés.

# Diagramme UML final :



# Choix d’implémentation :

*Nous avons dû prendre plusieurs choix durant la conception du projet :*

La plateau de jeu est une matrice à double entrée, où chaque case contient un objet Case. Cet objet contient, lui, un éventuel objet Lettre.

Nous avons utilisé des ArrayLists pour la main du joueur et le sac de lettre car les méthodes disponible avec les listes était utiles pour gérer les mouvements de lettre de l’un vers l’autre (.add, .remove, .removeAll).

Le dictionnaire, se trouvant dans la classe Plateau est une HashSet qui charge à l’initialisation un fichier .txt de ± 400 000 mots. Nous avons pris une HashSet pour supprimer instantanément les éventuel doublons. La méthode .contains nous a évidemment été très utile pour rechercher des mots.

Nous avons choisi d’utiliser un singleton pour le message d’erreur //TODO

Pour l’utilisation des sockets, nous avons dû implémenté la sérialisation afin que les classes Lettre, Case et Joker puissent être envoyée avec le plateau.

La classe ScrabbleViewConsoleLancement permet de demander à l’utilisateur les données nécessaires à l’initialisations des sockets. Cette classe sort légèrement du pattern MVC, mais c’était plus agréable de pouvoir lancer l’application sans devoir définir les paramètres avant le lancement.

# Difficultés rencontrées :

## Modèle :

### Plateau :

* Vérification des mots adjacents nouvellement créés
* Calcul du score et bonne gestion des bonus

*Résolution des problèmes :*

Une des grosses interrogation au début du programme était de trouver un moyen de vérifier les mots posé ainsi que les nouveaux mots formés.

Nous avons trouvé la solution ensemble, en écrivant des algorithmes de vérification en pseudocode. Ensuite, en se basant sur ces algorithmes et après de nombreux essais, la vérification étaient implémentée. Elle sera modifiée encore de nombreuses fois, au fil des bugs rencontrés grâce aux tests et aux JUnits.

Nous avons suivi le même mode opératoire pour l’élaboration du calcul de score; en se basant sur des méthodes écrite en pseudocode.

## Interface graphique :

* Tentative d’implémentation du drag’n drop finalement abandonné après avoir réussi à rendre les pièces déplaçables.

Le problème était que je n’avais pas écrit les méthodes utilisées par manque de connaissance et malgré le succès de la première partie, je n’arrivais pas à empêcher les pièces de se superposer ce qui avait pour conséquence de détruire la lettre.

J’ai donc abandonné cette possibilité car le projet était encore loin d’être fini et j’ai trouvé préférable de faire une interface graphique fonctionnelle plutôt que d’essayer pour finalement ne pas finir l’intégralité des consignes demandées. Les seuls choses qu’il restait à faire pour ce drag’n drop étaient d’empêcher les lettres de se superposer et de lire le layout pour récupérer les modifications une fois que l’on appuyait sur jouer.

# Améliorations éventuelles :

## Global :

* Permettre de jouer à trois et quatre joueurs.
* Permettre de jouer dans d’autres langues.
* Permettre une sauvegarde du jeu pour pouvoir reprendre la partie plus tard.

## Interface console :

* Ajout de fonctionnalité pour permettre de relancer directement une partie à la fin d’une, abandonner la partie et pouvoir sauvegarder la partie.

## Interface graphique :

* Utilisation du Drag’n Drop pour rendre l’interface graphique plus agréable à l’utilisation.
* Ajout d’un menu pour pouvoir relancer directement une partie lorsque l’on en termine une, abandonner la partie et permettre la sauvegarde de la partie.

# Conclusion personnelle :

Simon Fauconnier :

Personnellement, j’ai beaucoup appris grâce à ce projet. Nous avons réussi à résoudre de nombreuses interrogation grâce aux nombreux tutoriels disponible sur le net (ArrayList, Socket,…)

Le fait de recréer un jeu physique déjà existant nous a imposé des contrainte, en effet nous devions rendre un jeu de scrabble fonctionnel, avec toute les règles officielles implémentées. C’était donc un défi d’arriver à faire fonctionner l’application avec toutes les fonctionnalités décrite dans le cahier des charges.

Nous avons délégué le travail dès le début du projet. GitHub s’est révélé très pratique pour suivre l’évolution du projet en restant attentif à la progression de l’autre. Nous avons beaucoup discuté ensemble des problèmes rencontrés afin des trouver des solutions adaptées.