Introduction :

Le projet de Morpion Solitaire a été réalisé en binôme par Houssam et Maxime. Houssam s'est occupé de la partie front-end, c'est-à-dire de la création de l'interface graphique et de la connexion avec le back-end. Maxime s'est quant à lui occupé de la partie back-end, c'est-à-dire de la logique du jeu en elle-même.

Nous avons choisi cette répartition des tâches car nous souhaitions mettre en avant nos compétences respectives et nous orienter vers les aspects du développement qui nous intéressaient le plus. De plus, en travaillant en binôme, nous avons pu profiter de l'aide et de l'expertise de l'autre pour progresser plus rapidement et efficacement sur nos parties respectives.

Méthodologie :

Avant de débuter le développement du projet, nous avons pris le temps de bien réfléchir à la manière dont nous allions structurer notre code. Nous avons échangé sur les différentes classes que nous allions devoir créer et sur la manière dont elles allaient interagir entre elles. Nous avons également discuté de la connexion entre le front et le back. En prenant le temps de bien planifier notre projet, nous avons pu démarrer le développement en ayant une vision claire de ce que nous voulions réaliser.

Avant de débuter la réalisation du projet, nous avons pris le temps de bien réfléchir à la structure du code et à la manière dont nous allions nous répartir les tâches. Nous avons décidé de nous fixer des points de rendez-vous réguliers afin de pouvoir avancer de manière efficace sur le projet. Nous avons également veillé à bien segmenter le code pour pouvoir identifier rapidement les éventuels problèmes et les corriger rapidement. Nous avons également discuté de la connexion entre le front et le back. En prenant le temps de bien planifier notre projet, nous avons pu démarrer le développement en ayant une vision claire de ce que nous voulions réaliser.

Méthodologie back :

Pour coder le back du projet, nous avons décidé de travailler de manière itérative, en commençant par déterminer les classes essentielles et leur fonctionnalité. Nous avons également défini les entrées et les sorties de chaque classe afin de pouvoir bien connecter le front et le back. Nous avons également accordé une importance particulière à l'ordre de création des classes, car certaines classes étaient plus essentielles que d'autres et devaient être mises en place avant d'autres.

Méthodologie front :

Houssam a choisi de commencer par implémenter l'interface graphique avant de se concentrer sur la logique de jeu afin de s'assurer que l'interface soit fonctionnelle et agréable à utiliser. Houssam a utilisé des objets de type Point2D pour envoyer les coordonnées du point cliqué au back, qui pouvait alors traiter cette information et renvoyer une réponse au front. Enfin, Houssam a travaillé sur la gestion des scores, en utilisant un fichier texte pour enregistrer les scores de chaque partie et afficher ces scores dans une table à l'aide de la bibliothèque JavaFX.

Comment marche le projet :

Front :

La classe principale du back du projet est la classe "CreateGraphiqueBoard". Cette classe s'occupe de gérer l'affichage graphique du jeu de Morpion Solitaire, en affichant une grille de cercles représentant les intersections de lignes sur lesquelles le joueur peut placer ses pions. Elle inclut également un label indiquant l'état actuel de la partie et des boutons permettant de jouer et d'obtenir des indices.

Pour créer l'interface graphique du jeu, nous avons commencé par ajouter des lignes horizontales et verticales sur la grille. Ensuite, nous avons calculé les intersections entre ces lignes afin de pouvoir placer des cercles sur chacune d'elles. Ces cercles sont clicables et permettent au joueur de sélectionner une intersection. Une fois cette base mise en place, nous nous sommes occupés du score et des autres éléments de l'interface tels que les boutons et les labels.

Back :

Pour le back, la classe principale est GameMechanics. Elle contient les fonctions nécessaires pour mettre en place le jeu, comme reset qui réinitialise le plateau et la liste des mouvements et lignes, ou encore isHorizontalPossible, isVerticalPossible et isLeftDiagPossible qui permettent de déterminer si il est possible de poser une ligne sur le plateau selon une direction donnée. Il y a également la fonction canAdd qui vérifie si un mouvement est valide, et la fonction makeMove qui ajoute un mouvement à la liste et met à jour le plateau. Enfin, la fonction gameWon vérifie si le jeu est terminé et la fonction getHint donne une suggestion de mouvement au joueur.

Pour le back, nous avons décidé de créer deux classes distinctes pour gérer les règles de jeu 5D et 5T. Cependant, la logique de base était assez similaire entre les deux. Nous avons commencé par coder la version 5D, qui était la plus simple, avant de passer à la version 5T.

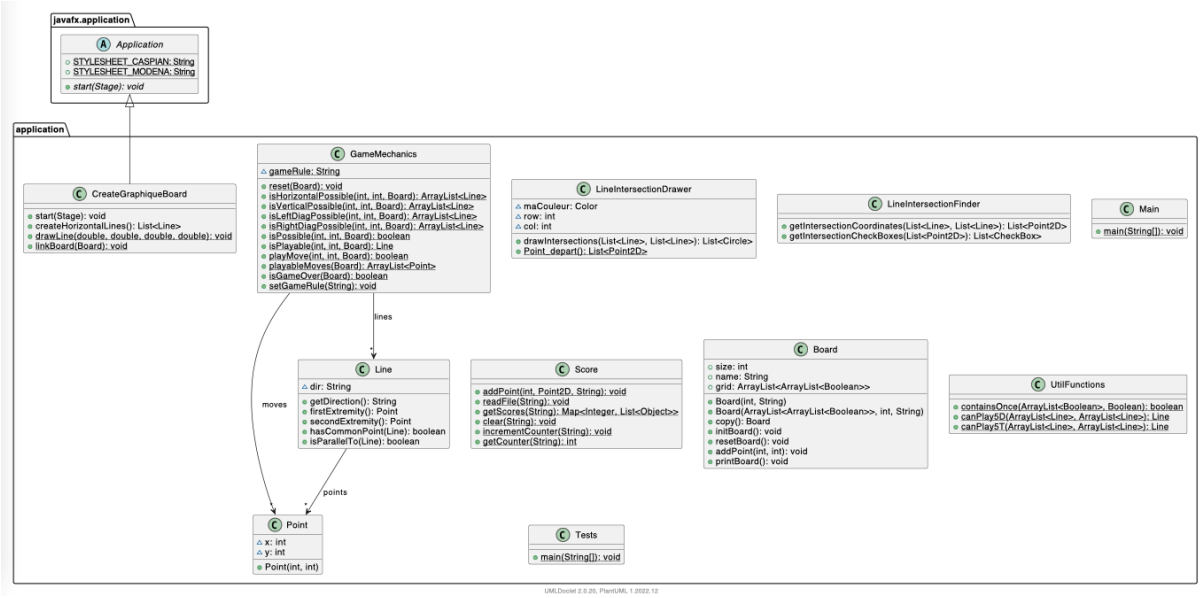
Interconnexion Front – Back :

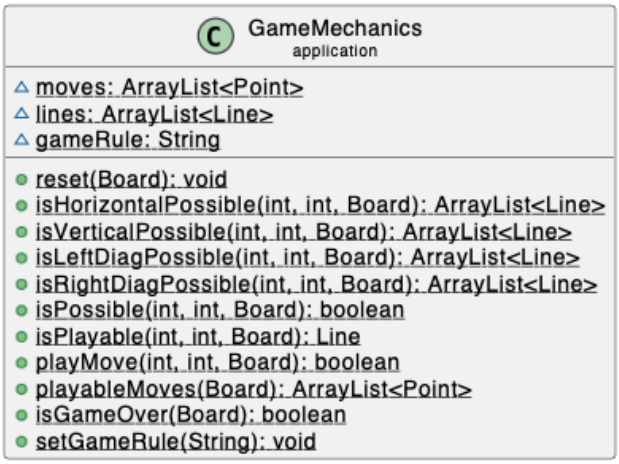
Pour le premier moment d'interconnexion, lors de la création de la grille graphique, nous avons décidé de créer une instance de la classe Board qui représente la grille logique. Cette instance était alimentée par les points ajoutés sur la grille graphique par l'utilisateur en cliquant sur les cercles. Cela nous a permis de synchroniser les deux grilles et de mettre à jour la grille logique en temps réel.

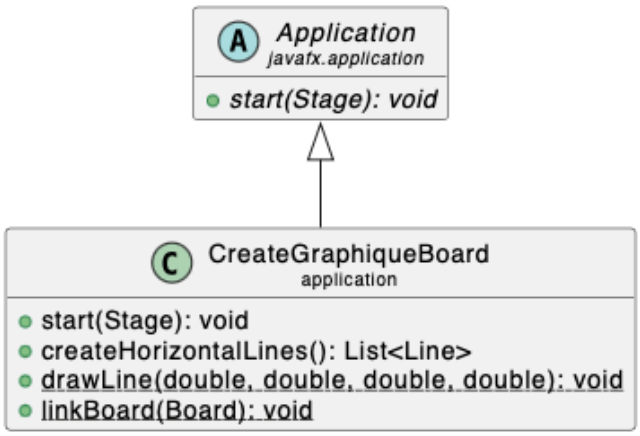
Pour le second moment d'interconnexion, nous avons implémenté la logique de création de lignes dans la classe GameMechanics. Cela nous a permis de détecter si une ligne pouvait être créée à partir des points sélectionnés sur la grille. Si une ligne pouvait être créée, nous avons ajouté cette ligne à la liste de lignes de la classe Board et avons mis à jour la grille graphique en ajoutant une ligne entre les points correspondants.

Pour le troisième moment d'interconnexion, nous avons connecté les boutons du front à des fonctions du back. Par exemple, lorsque l'utilisateur clique sur le bouton "Refresh", le plateau de jeu est entièrement réinitialisé et toutes les lignes créées sont supprimées de la liste. Cela a été mis en place en utilisant une fonction du back appelée "reset", qui remet à zéro le plateau de jeu et les listes de lignes et de mouvements. Cette fonction a été appelée lorsque le bouton "Refresh" du front était cliqué, permettant ainsi à l'utilisateur de recommencer une partie à tout moment.

Diagramme de classe :







Difficultés rencontrées :

Nous avons rencontré plusieurs difficultés au cours de la création de notre projet. Tout d'abord, il était difficile de se lancer dans le développement du jeu et de savoir par où commencer. Nous avons dû réfléchir à toutes les classes nécessaires avant de pouvoir commencer à coder, ce qui a été assez compliqué. Nous avons également dû penser aux interconnexions entre le front et le back avant même de commencer à écrire du code, ce qui a été une tâche assez difficile.

Une autre difficulté que nous avons rencontrée a été de devoir apprendre un package d'interface graphique en étant seuls. Nous n'avions jamais utilisé de JavaFX auparavant et il a fallu un certain temps pour comprendre comment l'utiliser correctement. De plus, il a été difficile de trouver de la documentation en français pour certaines parties de JavaFX, ce qui a ralenti notre progression.

Une autre difficulté a été de comprendre le code de l'autre personne, car Houssam s'occupait du back-end et Maxime du front-end. Il a fallu prendre le temps de se familiariser avec le code de l'autre pour pouvoir travailler efficacement en binôme.

Fonctionnalités implémentées :

Nous avons ajouté une interface graphique à notre jeu afin de permettre à l'utilisateur de jouer de manière interactive. Nous avons également inclus les deux versions du jeu, 5D et 5T, en utilisant les règles spécifiées pour chaque version. En outre, nous avons intégré un algorithme de recherche aléatoire qui permet au jeu de suggérer des coups à l'utilisateur lorsqu'il est bloqué. Grâce à ces ajouts, notre jeu est maintenant complet et prêt à être joué. Cela ajoute une dimension supplémentaire au jeu et permet à l'utilisateur de s'entraîner ou de découvrir de nouvelles stratégies de jeu.

Fonctionnalités supplémentaires facultatives implémentées :

En plus des fonctionnalités obligatoires, nous avons également ajouté des fonctionnalités facultatives afin de rendre le jeu plus agréable pour l'utilisateur. Nous avons ajouté un tableau des scores qui permet à l'utilisateur de voir ses performances précédentes et de se mesurer aux autres joueurs. Nous avons également ajouté un système d'indices qui permet à l'utilisateur de demander une suggestion de coup s'il se trouve dans une impasse. Ces fonctionnalités facultatives ont été ajoutées afin de rendre le jeu plus accessible et plus amusant pour tous les joueurs.

Tests Unitaires :

Nous avons effectué plusieurs tests sur notre application. Tout d'abord, nous avons testé chacune des fonctionnalités individuellement pour nous assurer qu'elles fonctionnaient correctement. Ensuite, nous avons testé l'application dans son intégralité pour nous assurer que toutes les fonctionnalités fonctionnaient de manière cohérente et qu'il n'y avait pas de bugs. Nous avons également testé l'application avec différents types d'entrées pour être sûrs qu'elle gère correctement les erreurs. Enfin, nous avons testé l'application avec différentes tailles de grille pour nous assurer qu'elle fonctionne de manière optimale dans tous les cas.

Nous avons également réalisé une série de tests unitaires afin de s'assurer que chaque fonctionnalité du jeu fonctionne correctement de manière indépendante. Cela nous a permis de détecter et de corriger rapidement les éventuels bugs ou erreurs de code. Nous avons également testé le jeu en utilisant différentes combinaisons de fonctionnalités pour vérifier que l'ensemble du jeu fonctionne de manière cohérente. Enfin, nous avons demandé à plusieurs personnes de jouer au jeu et de nous faire part de leurs retours pour nous assurer que le jeu était facile à comprendre et à utiliser.