Rendu 1 : Rapport

Nous avons respecté l'architecture MVC, avec MastermindGame représentant le modèle, MastermindGameDisplay la vue, et RoundController le contrôleur.

Composants principaux

Combinaison et tentatives : Dans le jeu, une combinaison représente une ligne de pions. Nous avons modélisé cela avec la classe Combination. Cette classe est utilisée pour deux objectifs principaux : représenter la combinaison secrète définie par l'ordinateur et les combinaisons tentées par le joueur. Chaque tentative est donc une instance de la classe Combination.

Indices comme attribut : Au lieu d'utiliser une classe distincte pour les indices, nous avons opté pour les intégrer en tant qu'attributs au sein de la classe Combination. Chaque combinaison contient un attribut qui stocke les indices correspondants, indiquant si un pion est bien placé, mal placé ou absent. En mode numérique, le premier indice du tableau est spécialement formaté pour afficher le nombre de pions bien placés et mal placés, fournissant ainsi une information concise et directe au joueur.

Manche : Une manche dans notre application est définie comme l'ensemble des tentatives ayant permis ou non de trouver une combinaison secrète. Cela est représenté par la classe Round, qui contient une collection de tentatives (instances de Combination). Chaque Round représente donc une série d'essais du joueur pour deviner la combinaison secrète, avec les indices correspondants fournis après chaque tentative.

Design patterns utilisés

Strategy :

- Nous avons adopté le design pattern Strategy pour la génération des indices, adaptée selon le mode de jeu. Ce pattern permet de définir une série d'algorithmes (différentes manières de générer des indices) et de les rendre interchangeables sans modifier la classe principale.

- Dans notre cas, la classe Combination référence l'interface GenerateCluesStrategy. Cette interface est implémentée par plusieurs stratégies comme GenerateCluesEasy, GenerateCluesClassic, et GenerateCluesNumerical, chacune correspondant à un mode de jeu différent. Ainsi, la stratégie de génération des indices peut être aisément modifiée selon les besoins du jeu ou les choix du joueur.

Observer :

- Nous avons choisi d’observer la manche plutôt que le jeu entier, car c’est la manche qui est affichée et interagit avec l’utilisateur.

MVC (décrit dès les premières lignes)

Annexes

Diagramme UML
Code source.