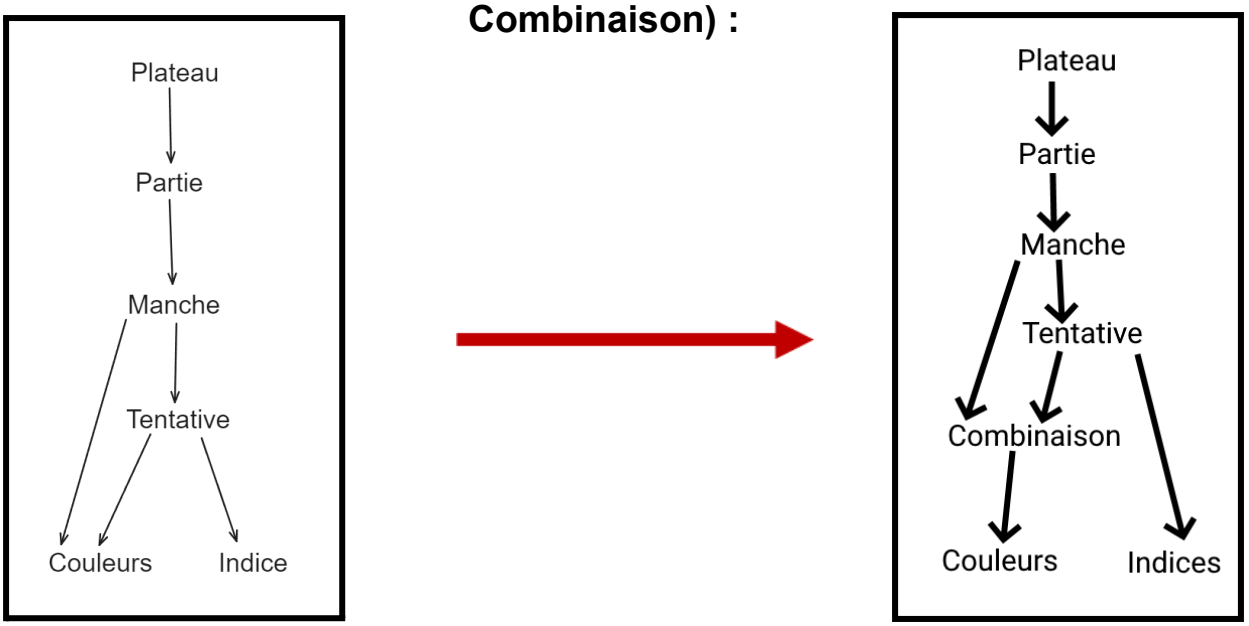
Projet Mastermind A31 : Nouveaux choix de conception

Eliott Schott - Sylvain Schaefer

Nouvelle organisation des classes du modèle (ajout de Combinaison) :

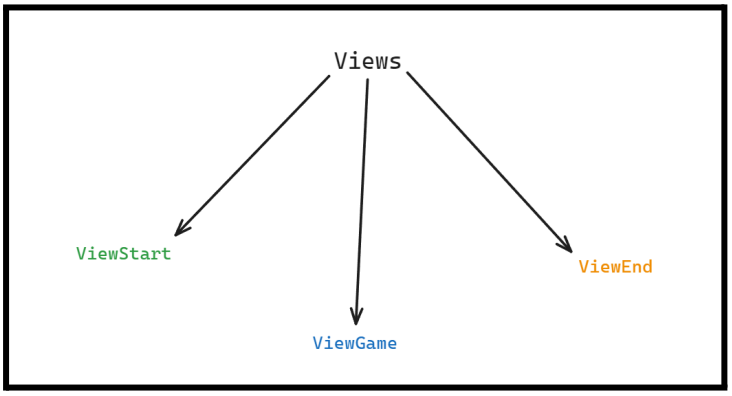


Ajout de la classe Combinaison:

- On a décidé de créer cette classe pour éviter à la **Manche** d'avoir trop de responsabilités (la classe **Combinaison** permet de générer la combinaison secrète).
- Possède d'autres méthodes pour obtenir la longueur de la combinaison et pour placer une couleur à un index donné.

Classe abstraite Views :

Grâce à cette classe abstraite, chaque vue possède la bonne icône et peut changer la police des composants grâce aux méthodes de cette classe abstraite. Sans celle-ci nous aurions dû écrire du code répétitif.



Les vues :

- **ViewStart** permet au joueur de définir les paramètres comme son nom, le nombre de manches, le nombre de pions disponibles, le nombre de pions dans les combinaisons, le nombre de tentatives par manche et le type d'indices.
- **ViewGame** permet de jouer les manches : choisir sa combinaison, afficher les indices correspondant
- **ViewEnd** permet d'afficher le nom du joueur et les statistiques de la partie

Les tableaux tabScores[] et tabTenta[] :

La classe **Partie** contient maintenant des tableaux pour stocker les statistiques de la partie. Ces tableaux permettent de stocker le score et le nombre de tentatives de chaque **Manche**. Ils sont contenus dans la **Partie**, car elle est "parent" des différentes **Manche**.

Type d'indice :

La classe **Partie** et la classe **Manche** contiennent la variable typeIndice. Cela permet aux observers de savoir ce qu'il doit afficher dès que le joueur valide sa tentative (il est notifié), et à la manche de calculer

l'indice correspondant à la tentative du joueur et le type d'indice sélectionné au début de la partie.

Nous n'avons pas décidé d'appliquer le design pattern Strategy car le choix d'affichage des indices se fait au début du jeu et ne peut être changé durant la partie.
En effet, le joueur choisit uniquement ce paramètre en début de partie, on a donc pas besoin de modifier ce paramètre dynamiquement.
(un entier : 0 pour "facile", 1 pour "classique" ou 2 pour "numérique").

Méthode newManche :

Nous avons ajouté une méthode "newManche()" dans l'interface ***MastermindObserver*** afin d'afficher une fenêtre qui indique si la manche est gagnée ou perdue avec le nombre de tentatives correspondant.

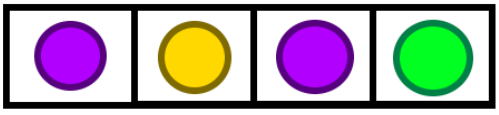
Abandon :

Afin de gérer le cas de l'abandon d'une manche, nous avons ajouté une méthode "giveUp()" dans **GameController** qui sera appelée par la vue lors de l'appui sur le bouton "Abandonner". **GameController** appellera ensuite la méthode "giveUp()" située dans **Manche** et, grâce à une méthode "notify" **Manche** va appeler la méthode "newManche()" de ***MastermindObserver*** décrite précédemment.

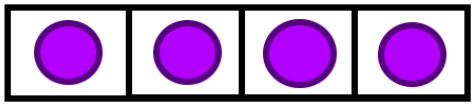
Dictionnaire dans la classe Manche :

Ajout d'un dictionnaire dans la classe **Manche**, qui facilite la méthode de calcul d'indices.
Notre fonction de calcul ne donnait pas les bons indices dans certains cas. Voiçi un schéma pour mieux expliquer le problème :

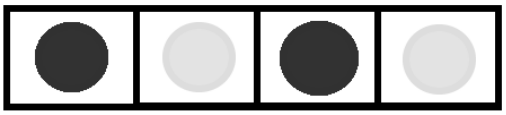
La combinaison générée par la classe **Combinaison** est la suivante :



Si la tentative du joueur est la suivante :

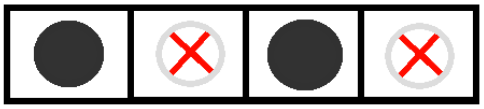


l'ancien mode de calcul affichera ces indices :



Ce n'est pas faux de dire que le violet de la 2ème case et de la 4ème case se trouvent dans la combinaison secrète, mais cela peut porter à confusion en laissant croire au joueur qu'il y a plus de 2 violets.

Après avoir corrigé cette fonction, l'indice affiché sera le suivant :



Ici, plus de confusion, car le joueur saura qu'il n'y a que 2 violets dans la combinaison secrète.

Le dictionnaire permet de stocker le nombre d'occurrences de chaque couleur dans la combinaison secrète.
Dans la fonction de calcul d'indices, un premier tour de boucle pose les indices noirs, et incrémente un compteur temporaire pour chaque couleurs trouvées.
Le deuxième tour de boucle pose les indices "blancs" si la couleur est présente dans la combinaison, sauf si toutes les occurrences ont déjà été trouvées (on compare la valeur du compteur temporaire et du dictionnaire qui possède le nombre d'occurrences de chaque couleur).

Conclusion :

Depuis le premier rendu, à part l'ajout de la classe **Combinaison**, les principaux changements ne concernent pas le modèle mais plutôt les vues.