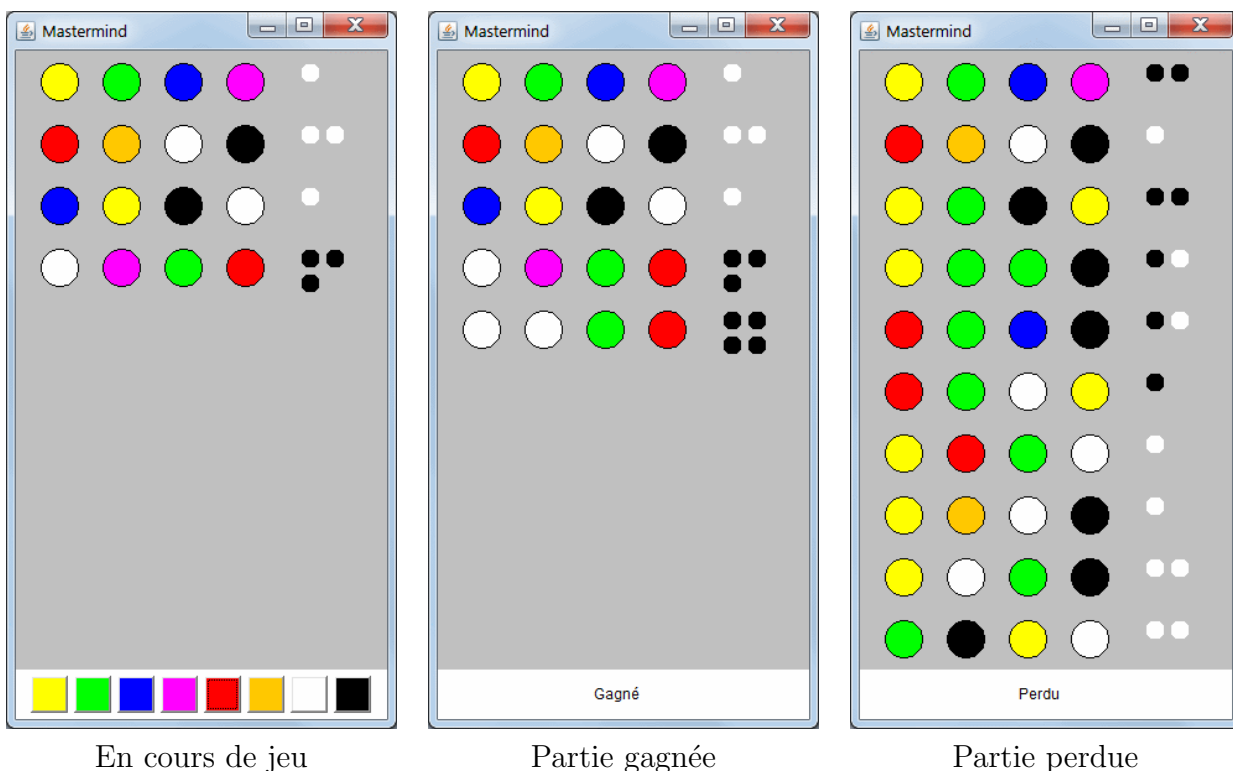


# Mastermind

## Exercice de synthèse


Le Mastermind est un jeu de réflexion classique dans lequel le joueur doit deviner une combinaison tirée aléatoirement par le jeu. Pour cela, le joueur fait des propositions, qui sont évaluées par le jeu qui indique au joueur le nombre de jetons de sa proposition qui appartiennent bien à la combinaison, bien placés ou non.

L'illustration ci-dessous montre 3 étapes du jeu. Tant que le jeu n'est pas terminé, le joueur fait des propositions (des rangées de  $n$  jetons de couleur) que le jeu évalue. Lorsque le joueur a trouvé la bonne combinaison (les  $n$  jetons corrects et bien placés), il gagne. Si le joueur ne devine pas la combinaison au bout de  $t$  tentatives, il perd. Les illustrations suivantes montrent le jeu avec les paramètres suivants :  $n = 4$  et  $t = 10$ .





## Exercice 1 – Modèle du jeu


La première étape consiste à modéliser les données du jeu.

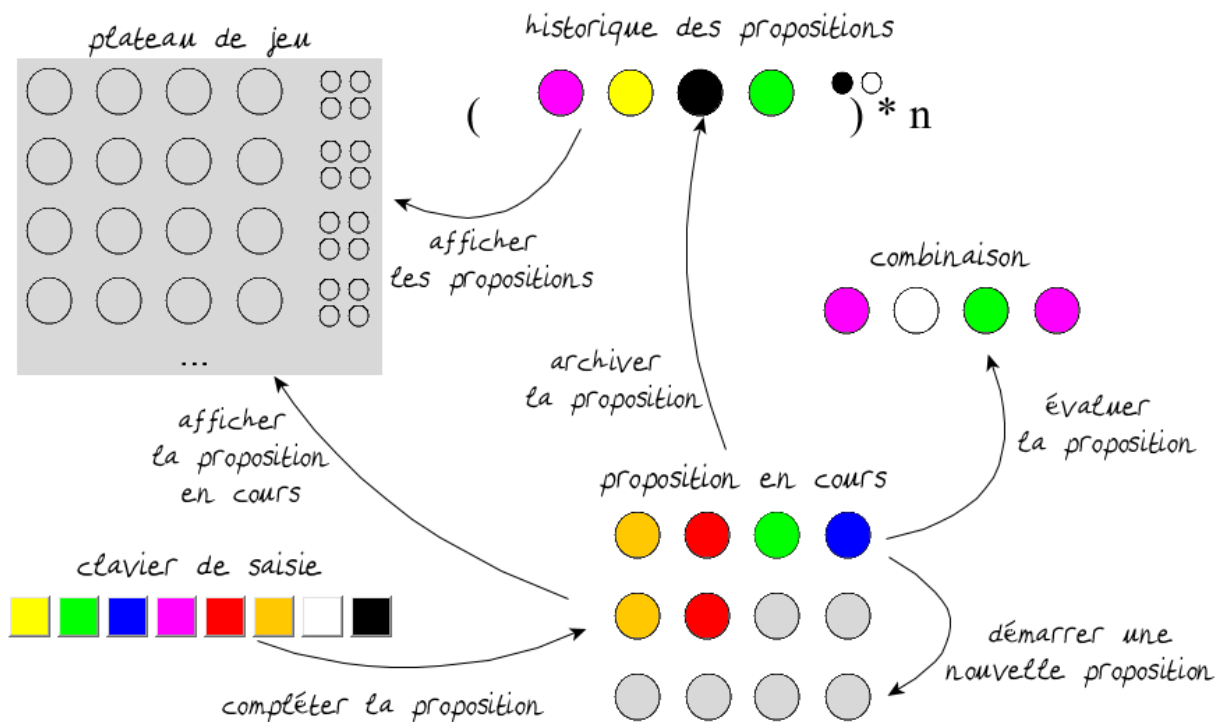
 On commence par définir dans une classe **Modèle** les constantes qui composent le modèle du jeu :

nom	type	description
COULEURS	tableau de <b>Color</b>	Ce tableau contient les constantes couleurs suivantes : Jaune, Vert, Bleu, Magenta, Rouge, Orange, Blanc, Noir
N_TENTATIVES	entier	Permet d'indiquer le nombre de tentatives autorisées (10)
DIFFICULTE	entier	Permet d'indiquer le nombre de jetons de la combinaison (4)
Etat	enumération	Représente les différents états du jeu : EN_COURS, GAGNÉ et PERDU.

 On définit ensuite dans une classe **Rangée** le concept de rangée. Une rangée est un tableau **jetons** de jetons (couleurs) de taille **Modèle.Difficulté**. Un entier **indiceJeton** permet d'indiquer si la rangée est complète (**indiceJeton** = **Modèle.Difficulté**), si elle est en cours de complétion ( $0 < \text{indiceJeton} < \text{Modèle.Difficulté}$ ) ou bien si elle n'a pas encore été entamée (**indiceJeton** = 0). Enfin, à une rangée est associée un résultat, qui se présente sous la forme d'un couple d'entiers **noirs** et **blancs**, qui correspondent aux nombres de jetons noirs (dans la combinaison et bien placés) et blancs (dans la combinaison mais mal placés) associés à une rangée complétée.


 On définit enfin dans la classe **Modèle** les variables qui constituent le modèle du jeu. Le jeu est d'abord caractérisé par un **état**, de type **Etat**, par une **combinaison** de type **Rangée** qui sera tirée aléatoirement en début de partie, par un tableau de **Rangée** nommé **propositions** qui conserve les propositions du joueur, et enfin par un entier **tentative** qui représente le numéro de la tentative en cours. On considérera que la tentative en cours correspond à la rangée **propositions[tentative]**.

 Dans les classes **Modèle** et **Rangée**, on créera toutes les méthodes présentant une utilité (voir graphique ci-dessous).



## Exercice 2 – Interface graphique du jeu

On s'intéresse ensuite à l'interface graphique du jeu.


 L'application graphique se compose d'une fenêtre (**Frame**) qui contient deux panneaux :

nom	type	description
VueClavier	Panel	Ce panneau contient autant de boutons qu'il existe de couleurs dans la constante <b>COULEURS</b> .
VuePropositions	Canvas	Ce canvas permet d'afficher les propositions passées et présente du joueur, ainsi que les résultats (pour les propositions passées). La taille du canvas sera calculée en prenant en compte le nombre de tentatives (hauteur de la zone d'affichage) et le nombre de jetons dans un combinaison (largeur de la zone d'affichage).


 Mettre en place le jeu des observateurs/observés entre les éléments de l'interface et le modèle de l'application.

## Exercice 3 – Contrôleur

Dans cette dernière partie, on écrit la partie contrôle de l'application.

 Dans une classe **Contrôleur**, destinée à traiter les événements de l'application, écrire le code permettant de régir l'évolution du jeu. Lorsque le jeu commence, une combinaison est tirée et une nouvelle proposition est créée. Lorsque le joueur clique sur un bouton, la proposition en cours est complétée par la couleur du jeton cliqué. Lorsque la proposition est complète, celle-ci

est corrigée et son résultat est affiché. Si la proposition correspond à la combinaison, le jeu se termine par la victoire du joueur (*Gagné* est affiché à la place des boutons). Sinon, en fonction du nombre de tentatives restantes, soit le jeu se poursuit par une nouvelle tentative (nouvelle proposition démarrée en dessous de la précédente), soit le jeu se termine par l'échec du joueur (*Perdu* est affiché à la place des boutons).

 Ajouter le contrôle suivant : à chaque fois que le joueur double-clique dans le panneau des propositions, le dernier jeton est retiré (annulé), et ainsi de suite jusqu'à ce que la proposition en cours soit vide. Une proposition complétée (et corrigée) ne peut en aucun cas être annulée.