



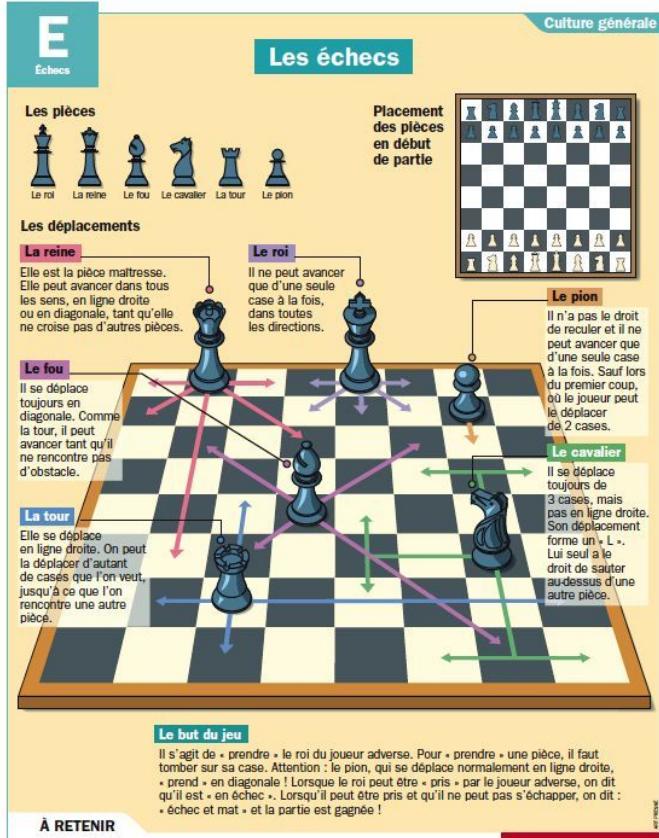
Chess game

Explication du jeux



Jeux d'échecs dans un terminal, chaque pièce aura une lettre attribuée.

Pion	Lettre	Déplacement
Roi	K	Dans toutes les directions. Une case à la fois.
Reine	Q	Dans toutes les directions. Plusieurs cases à la fois.
Fous	F	Seulement en diagonale. Plusieurs cases à la fois.
Cavalier	C	Déplacement en "L" (une case tout droit puis une case en diagonale).
Tours	T	Seulement en ligne droite (horizontale/verticale). Plusieurs cases à la fois.
Pion	P	Deux cases lors du premier tour uniquement puis une seule à la fois pendant le reste de la partie. Seulement en avant (ne peut pas reculer), SAUF si une pièce se situe proche de lui et à sa diagonale, il peut se déplacer et la manger.



Une pièce a le droit de se déplacer si :

- Sa case d'arrivée est conforme à son déplacement (ex, si je veux aller en face mais que je suis un fou, ce n'est pas possible).
- Aucune autre pièce ne se trouve sur le chemin (je ne peux pas sauter une pièce) .
- Aucune pièce de son équipe ne se trouve sur la case d'arrivée (je peux manger une pièce adverse, mais pas ma propre pièce, c'est évident...).
- Le déplacement ne met pas son roi en échec (si une reine menace mon roi mais que mon fou le protège, je ne peux pas bouger mon fou, parce que cela mettrait le roi en danger, et aux échecs, je n'ai pas le droit de mettre mon roi en danger !).

Fonctionnement du programme

Plateau :

1- Afficher le plateau

	A	B	A	B
1	(A1) Pion (0;0) blanc	(B1) Reine (1;0) noir	P	Q
2	(A2) Roi (0;1) noir	(B2) null (1;1) blanc	F	

Plateau

Affichage

~~ → nom de la case

~~ → pièce qui occupe la case
(null si case vide) = " " en affichage

(~~; ~~) → coordonnées de la case (determinées par le plateau ???)

~~ → background color de la case

Déplacement :

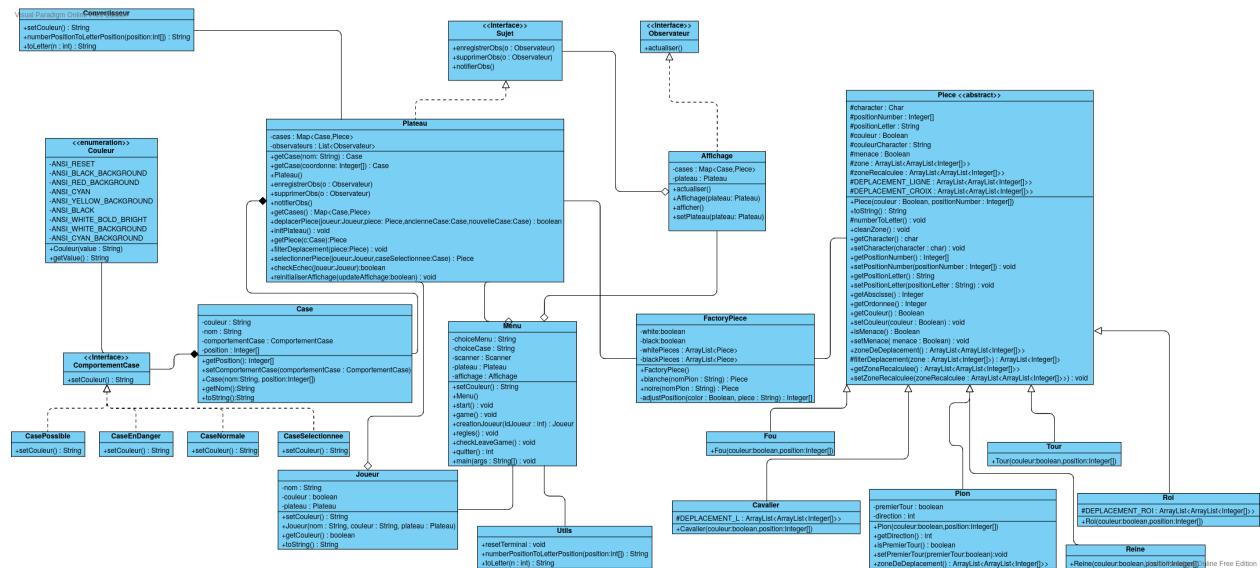
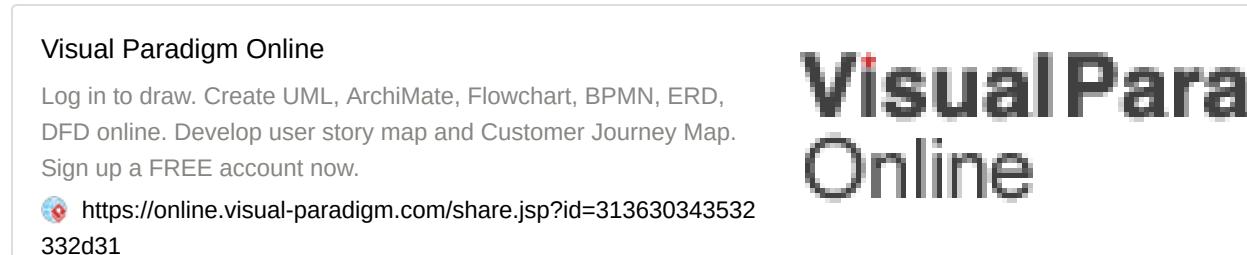
Pour le déplacement, on calcule toutes les cases sur lesquelles on peut aller, on met toutes ces cases possibles dans un tableau et on regarde si la case de destination est dans ce tableau, si oui on peut y aller. (cela permet de colorer toutes les cases sur lesquelles on peut aller pour mieux visualiser).

Pour savoir si le roi est en échec :

- D'abord, je regarde si ma pièce peut se déplacer (voir les 3 premières règles de déplacement plus haut)
 - Ensuite, je déplace temporairement la pièce en enregistrant sa position actuelle avant. Une fois déplacée, je regarde si mon roi est en échec. Si oui, je restaure la position de la pièce et je ne change rien. Si non, je déplace la pièce et je met à jour l'affichage.

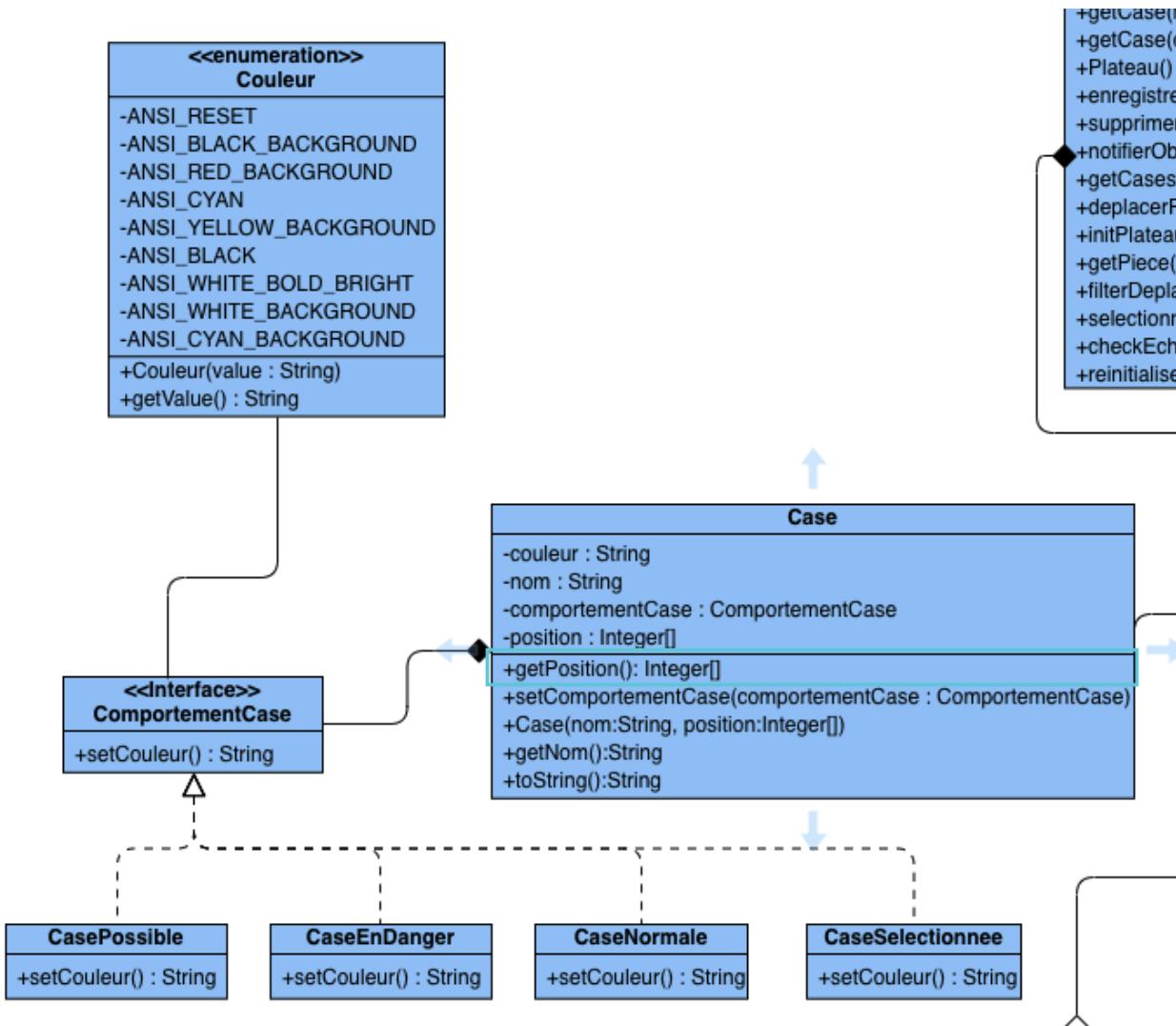
Il faut, un attribut casesPossibles qui est un tableau de toutes les cases sur lesquelles chaque pièce peut aller. À chaque déplacement, toutes les pièces mettent à jour leur tableau (Pattern observateur).

Diagramme UML

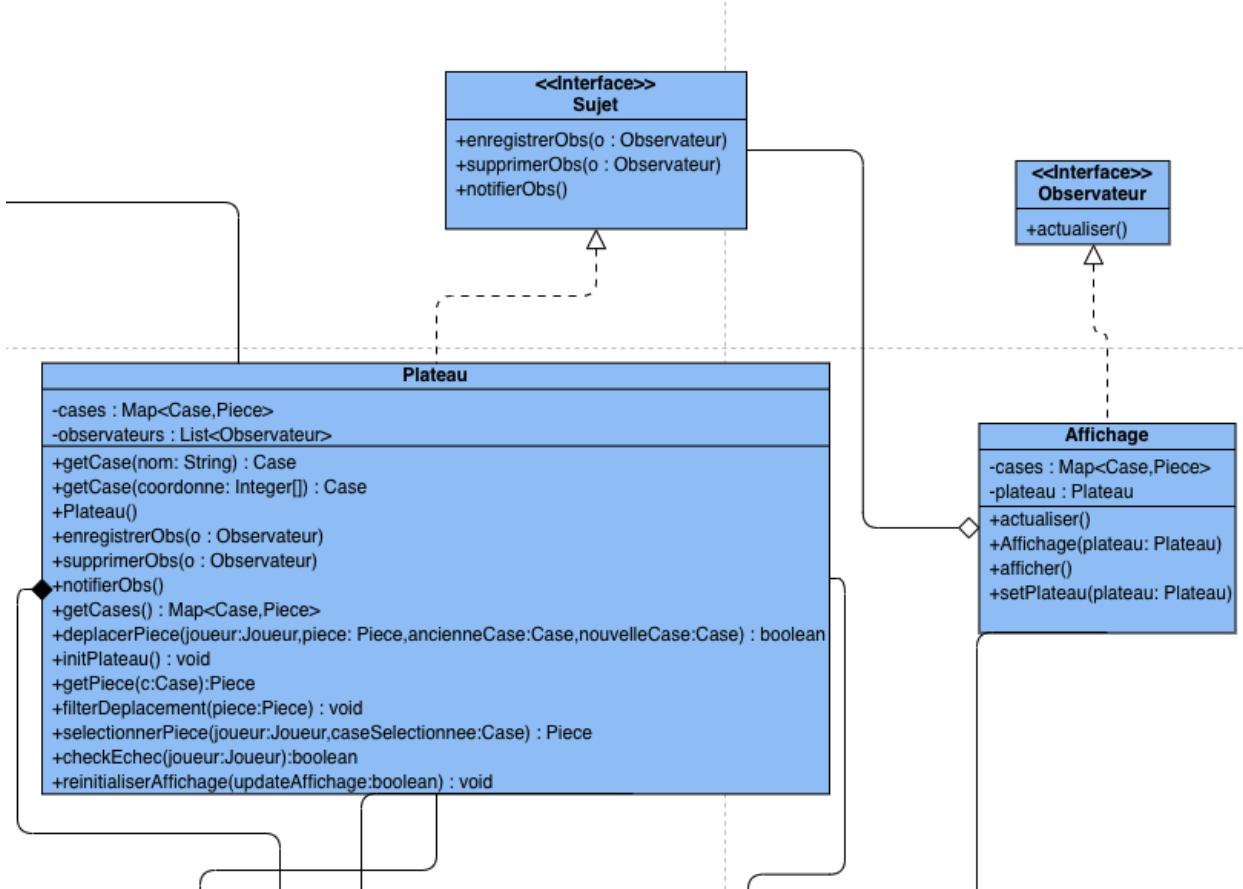


Patterns retenus

- Le pattern strategy pour modifier la couleur des cases en fonction de leur état (case normale, case en danger, case atteignable par une pièce). Par exemple, si ma pièce peut manger une autre, la case sur laquelle se situe cette dernière devient en danger et se colore en rouge.



- Le pattern d'observation met à jour l'affichage à chaque déplacement d'une pièce. C'est le plateau qui envoie les infos de ses cases, et l'affichage enregistre le nouvel état du plateau pour ensuite l'afficher.



- Pour la création des pièces nous utilisons le pattern factory qui sera divisé en deux méthodes : une pour fabriquer des pièces blanches, et l'autre pour fabriquer des pièces noires.

