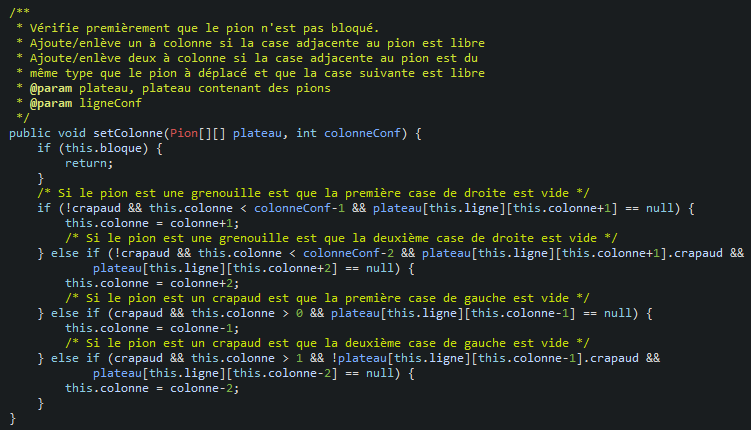
# Modification de la colonne



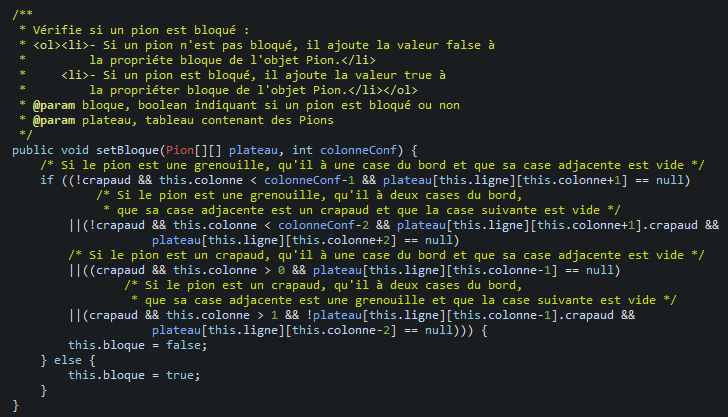
# Explication :

Cette fonction est appelée pour déplacer un pion théoriquement non bloqué. Cependant, cette fonction vérifie quand même ce boolean premièrement pour les tests unitaires et deuxièmement pour éviter de potentiels cas d’erreur et des vérifications inutiles. Cette fonction vérifie premièrement le type de pion ainsi que la colonne sur laquelle ce trouve le pion. Cela est nécessaire pour éviter les erreurs de déplacements si le pion se trouve à une extrémité du plateau. Ensuite, cette fonction ajoute ou enlève 1 ou 2 à la valeur de la colonne du pion en fonction de si sa case adjacente et libre ou non ou que sa case adjacente est du même type que celui-ci.

On remarque que les arguments sont tous deux issues d’une instance d’un objet plateau. Les vérifications sur l’initialisation de ces propriétés étant déjà faite préalablement, il n’y a aucun risque d’erreur de null pointer exception en appelant cette fonction.

Nous avons choisi cette structure car elle permet d’éviter un maximum de cas d’erreur et qu’elle est facilement maintenable. Par exemple, nous avions au début codé cette fonction pour qu’un pion puisse passer par-dessus un pion de type différent. Notre professer référent nous ayant indiqué qu’un pion ne pouvez chevaucher qu’un pion du même type, il ne nous à fallut que quelques minutes pour corriger ce problème. Cette fonction est donc facilement maintenable.

# Vérification du boolean bloque



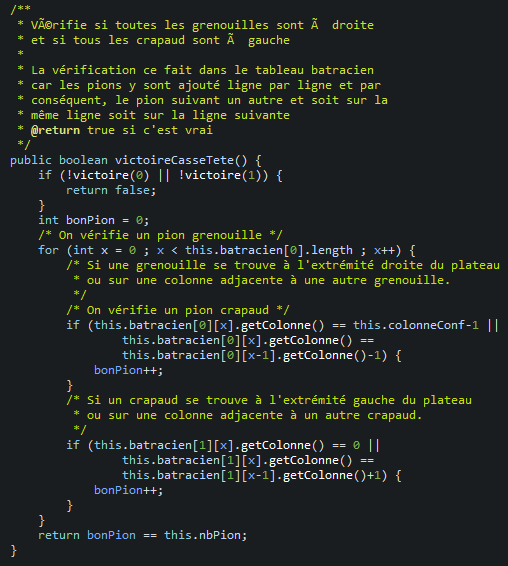
# Explication :

La structure de ce code est très semblable à celle de la fonction précédente à la différence qu’il n’y à qu’une seule boucle if. Cette boucle vérifie tous les cas dans le quel un pion de n’importe quel type est bloqué ou non.

Comme pour la fonction précédente, les arguments de cette fonction sont supposés non null pour éviter les cas d’erreur.

Nous avons choisi cette structure pour les mêmes raisons que la fonction précédente afin d’éviter un maximum d’erreur et pour la facilité de modification.

# Vérification de la victoire casse-tête



# Explication

Cette fonction est appelée après chaque action d’un joueur en mode casse-tête. Elle vérifie le bon emplacement des pions sur le plateau en fonction des pions contenu dans le tableau batracien de l’instance du plateau en cour. Cela se justifie par le fais que lors de l’initialisation de ce tableau, celle-ci parcours le plateau ligne à ligne ce qui signifie qu’un pion de ce plateau est forcément sur la même ligne ou sur la suivante.

Cette fonction ne peut pas générer d’erreur puisque le tableau batracien est initialisé chaque fois qu’une instance de Plateau est créé.

Cette fonction montre l’intérêt d’avoir une copie des pions du plateau dans un tableau à part. Cette fonction aurait été beaucoup plus longue elle aurait fait les vérifications directement sur le plateau.

# Affichage du plateau



# Explication :

Cette fonction permet de créer une représentation visuelle d’un plateau de pion qui sera ensuite définit en plateau. Elle affiche premièrement une ligne avec les numéros de colonne puis, dans une double boucle for, elle affiche toutes les lignes du plateau avec leur numéro et dans chaque case non vide, un C ou un G indique la présence d’un pion.

Les variables des boucles for se basant sur les dimensions du tableau, le seul cas d’erreur possible serait d’appelé la fonction sans l’initialiser. Or, dans notre code celui-ci est toujours initialisé avant l’appel de la fonction donc elle ne peut pas générer d’erreur.

La structure de ce code est très similaire à celui de la fonction toString de classe plateau. Elle est simple est l’affichage est équilibré.

# Création d’une copie des pions



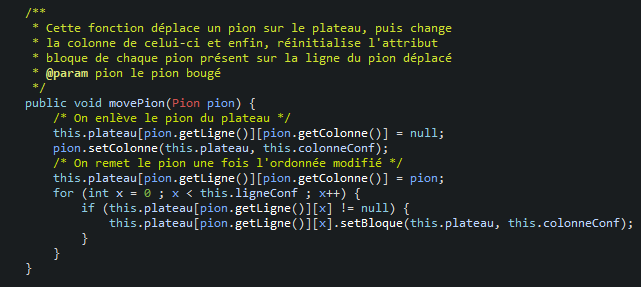
# Explication :

Cette fonction créé un plateau de pion contenant toutes les instances de pions avec leur caractéristique. Ce tableau permet d’effectuer des opérations plus rapidement sur les pions du plateau.

Cette fonction ne peut pas créer d’erreur car nbPion est toujours pair et s’il est égal à 0, les boucle for ne se lancerons pas.

On a décidé de créer ce plateau justement pour simplifier les opérations sur les instances de la classe Plateau comme les vérifications de victoire ou de la validité d’un pion.

# Déplacement d’un pion



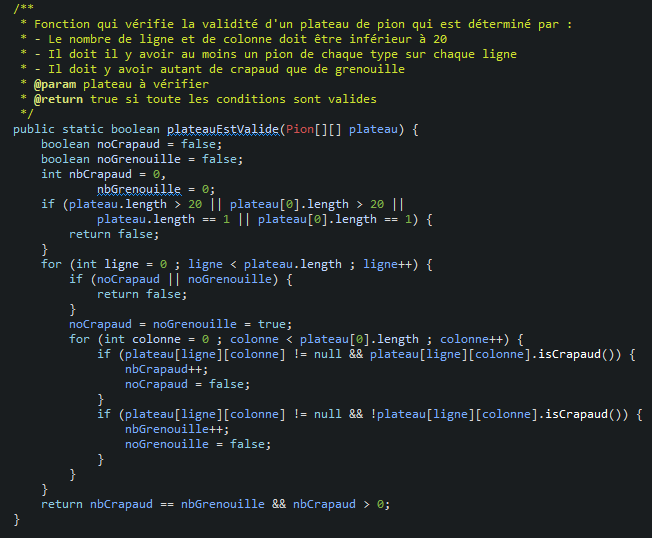
# Explication :

Cette fonction est appelée avec comme argument un pion qui a été préalablement été vérifié comme non bloqué. Elle change la position d’un pion avec la fonction setColonne que l’on a vu précédemment et qui modifie ainsi le plateau ainsi que la propriété bloque de chaque pion présent sur la ligne du pion bougé.

Cette fonction ne peut pas générer d’erreurs car le pion placé en argument à été vérifié préalablement et ces coordonnées sont comprises dans le plateau.

Nous avons choisi cette structure car elle est utilisable pour tous les types de pion et qu’elle permet facilement de de vérifier l’attribut bloque de la ligne modifiée.

# Vérification d’un plateau



# Explication :

Cette fonction vérifie si un tableau de pion placé en argument est valide pour crée une instance de Plateau. Elle vérifie les dimensions ainsi que le nombre de pion sur le plateau, qu’il y a autant de crapaud que de grenouille et que chaque ligne contient au moins un pion. Elle vérifie également qu’il y a plus d’un pion sur le tableau.

Cette fonction ne peut pas générer de cas d’erreur car le tableau placé en argument est toujours initialisé.

Nous avons créer cette fonction pour éviter les cas d’erreur lors de la création d’un Plateau s’il n’y a pas autant de pion crapaud et grenouille.

# Faire une action



# Explication :

Cette fonction permet d’effectuer une action en prenant en compte le mode jeu et la présence ou non de l’ordinateur. Elle prend en argument deux coordonnées qui seront vérifié et éventuellement validé pour un déplacement.

Cette fonction ne génère pas d’erreur car currentPlateau est toujours initialisé avant l’appelle de cette fonction.

Nous avons voulu faire une fonction polyvalente pour le déplacement des pions pour ne pas avoir à en faire plusieurs. De plus, elle est facilement maintenable si l’on veut rajouter des modes de jeu ect…

# Suppression d’une configuration

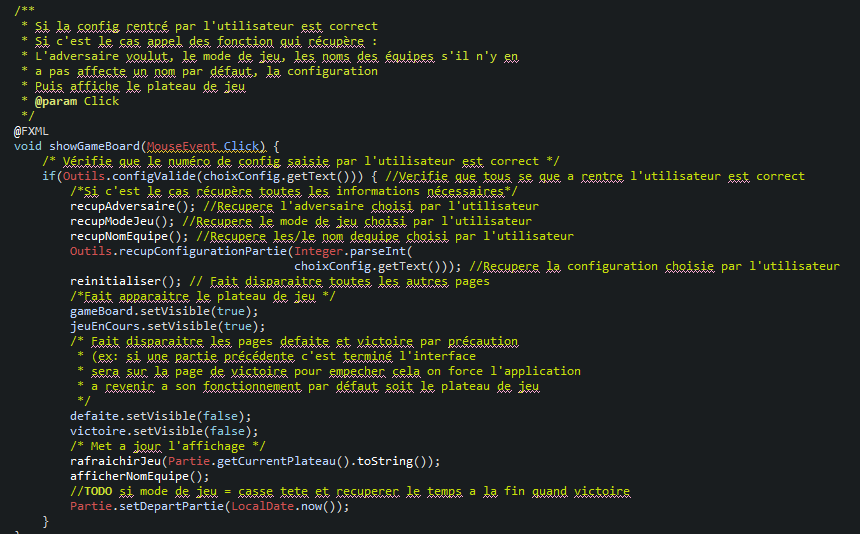
# Explication :

Exemple de fonction type de la classe outil :

Cette fonction est plutôt simple mais son fonctionnement général (Architecture) et est assez similaire a beaucoup d’autres fonctions de la classe outil.

Cette fonction effectue une série de test et affiche les messages d’erreurs correspondant sur l’interface pour informer l’utilisateur de son erreur. Si toutes les informations sont valides cette fonction retourne un boolean true qui permet par la suite d’executer la suite du code qui supprime la configuration.

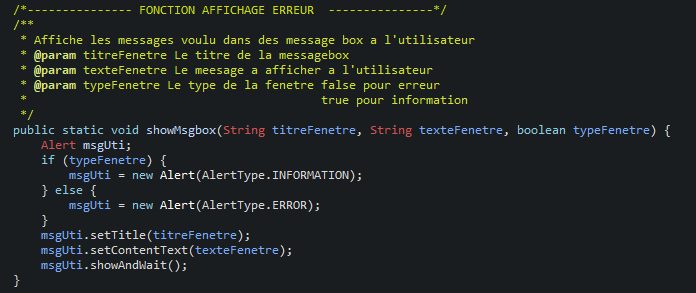
# Création pion



# Explication :

Son fonctionnement est assez simple une série de test simple sont effectués si ceux-ci ne sont pas valide alors on affiche à l’utilisateur un message d’erreur. Si tous sont correct alors on exécute le reste du code dans ce cas-là un pion est créé et retourne un boolean true signifie que le plateau a bien été initialisé dans la création d’une configuration) et que le programme peut donc afficher la page suivante

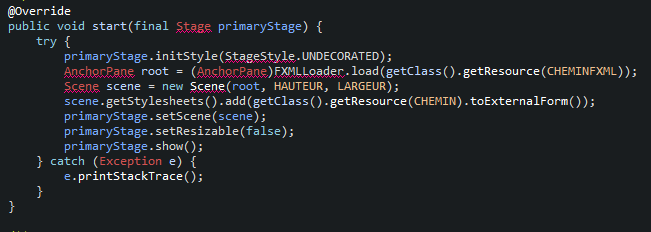
# Affichage des erreurs



# Explication :

Fonction qui permet d’afficher a l’utilisateur une message box lui indiquant son erreur / son avancement dans sa navigation. Cette fonction est très importante car elle représente un grand principe de l’IHM la rétroaction. Pour cette fonction nous avons choisi de l’appelé avec les paramètres définissant le titre de la message box, son texte et son type. Cette optimisation permet de rendre l’appel de cette fonction bien plus simple selon la situation le texte étant souvent le même on peut donc le définir comme constante sur le haut de la page. De plus comme notre l’application de nécessite que 2 types de message box l’utilisation du boolean simplifie encore plus l’utilisation de notre fonction.

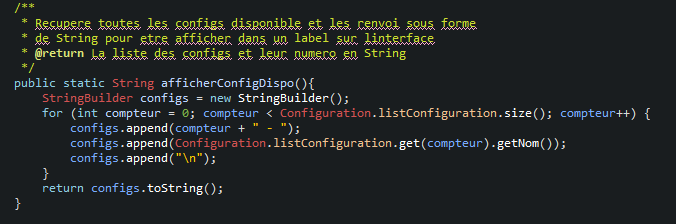
# Lancement application



# Explication :

Cette fonction est au cœur de notre application c’est elle qui permet de lancer notre application, d’appeler notre interface et ces ressources.

# Affichage des configurations disponibles



# Explication :

Cette fonction est très simple elle se contente de retourner le nom des configurations disponibles et leurs numéros sous la forme d’un String pour être afficher. Celle-ci n’est pas compliqué mais est essentiel dans la compréhension de l’interface et des possibilités disponible a l’utilisateur de l’application .Elle représente une caractéristique importante pour le fonctionnement d’une bonne IHM.