Fiche technique du Projet Tutoré

Azog-corp |

Auteur :

pIERRE MARIE COMBALBERT, Yanis Delmas, Yoan guiraud, maêl le folgoc pontis

2019

Table des matières

[1. Cahier des charges 4](#_Toc11095957)

[Description du jeu des grenouilles et des crapauds 4](#_Toc11095958)

[Description du casse-tête des grenouilles et des crapauds 5](#_Toc11095959)

[Application à développer 5](#_Toc11095960)

[Configurations initiales 6](#_Toc11095961)

[L’IA 7](#_Toc11095962)

[Remarque 7](#_Toc11095963)

[2. Spécification détaillée 7](#_Toc11095964)

[3. Diagramme de classe 9](#_Toc11095965)

[a) Partie 1 : 9](#_Toc11095966)

[a) Partie 2 : 11](file:///C:\Users\yanis\Documents\GitHub\Projet-tutore-S2\dossier%20du%20chef%20et%20de%20son%20secrétaire\ProjetTutoreS2_Combalbert_Delmas_Guiraud_LeFolgocPontis\DossiersTechnique.docx#_Toc11095967)

[4. Répartition des Taches 12](#_Toc11095968)

[5. Liste des objectifs 12](#_Toc11095969)

[6. Les codes 13](#_Toc11095970)

[7. Tests de outils : 13](#_Toc11095971)

[b) Test de recupType : 13](#_Toc11095972)

[c) Test de colonneValide : 14](#_Toc11095973)

[d) Test de ligneValide : 15](#_Toc11095974)

[e) Test de typeValide : 16](#_Toc11095975)

[f) 17](#_Toc11095976)

[g) Test verificationLettre : 17](#_Toc11095977)

[h) 18](#_Toc11095978)

[i) Test estNonVide : 18](#_Toc11095979)

[8. Résultat : 18](#_Toc11095980)

[j) Test de estValide : 18](#_Toc11095981)

[k) Test de recupType : 18](#_Toc11095982)

[l) Test de verificationLettre : 18](#_Toc11095983)

[m) Test de ligneEstValide : 19](#_Toc11095984)

[n) Test de colonneEstValide : 19](#_Toc11095985)

[o) Test de estNonValide : 19](#_Toc11095986)

[9. Test visuel Outils : 19](#_Toc11095987)

[p) RecupereLignePion, RecupereColonnePion : 19](#_Toc11095988)

[q) afficherConfigDispo : 19](#_Toc11095989)

[r) AttribuerNomJ1, attribuerNomJ2 : 19](#_Toc11095990)

[Personnalisé : 19](#_Toc11095991)

[Défaut : 20](#_Toc11095992)

[s) cordOk : 20](#_Toc11095993)

[Essai trop petit : 20](#_Toc11095994)

[Essai trop grand : 21](#_Toc11095995)

[t) Test configExiste : 21](#_Toc11095996)

[u) Test de placementPion : 21](#_Toc11095997)

[v) Test sauvegarder : 22](#_Toc11095998)

[w) Test supprimerConfig : 22](#_Toc11095999)

[x) Test suppressionPion : 23](#_Toc11096000)

[y) VerifVictoire : 24](#_Toc11096001)

[z) configIni : 24](#_Toc11096002)

[aa) placementPion : 24](#_Toc11096003)

[bb) configValide : 24](#_Toc11096004)

[cc) actualisationPlateau : 24](#_Toc11096005)

[10. Test InterfaceController : 25](#_Toc11096006)

[dd) Test showMVP (Pas de réel test) : 25](#_Toc11096007)

[ee) Test showRegle (Pas de réel test) : 26](#_Toc11096008)

[ff) Test showCreateur(Pas de réel test) : 28](#_Toc11096009)

[gg) Test showMenu(Pas de réel test)  : 28](#_Toc11096010)

[hh) Test showConfigurationPartie (Pas de réel test) : 29](#_Toc11096011)

[ii) Test showConfiguration (Pas de réel test) : 29](#_Toc11096012)

[jj) 30](#_Toc11096013)

[kk) Test verificationCheck, caseIaCheck, caseIaPasCheck, cacherJ2, cacherIA: 30](#_Toc11096014)

[ll) Test razIA: 31](#_Toc11096015)

[mm) Test razPartie : 32](#_Toc11096016)

[nn) Test verificationModeJeu, casseTeteConf, versusConf : 34](#_Toc11096017)

[oo) Test showGameBoard, recupNomEquipe : 35](#_Toc11096018)

[pp) Test actualiserJeu, rafraichirJeu : 36](#_Toc11096019)

[qq) Test configInitialisation : 38](#_Toc11096020)

[rr) Test ajoutPionConfig : 38](#_Toc11096021)

[ss) Test rafraichirConf: 40](#_Toc11096022)

[tt) Test deleteConfig: 41](#_Toc11096023)

[uu) Test enregistrerConfig : 42](#_Toc11096024)

[vv) Test showCreationConfig : 43](#_Toc11096025)

[ww) Test showSupprimerConfig : 44](#_Toc11096026)

[xx) Test showInitialisationConfig : 45](#_Toc11096027)

[11. Manuel d’utilisateur 45](#_Toc11096028)

[12. Bilan 47](#_Toc11096029)

# Cahier des charges

### Description du jeu des grenouilles et des crapauds

Le jeu oppose deux joueurs ou un joueur et une IA, et il consiste à déplacer des grenouilles et des crapauds sur un plateau quadrillé de N lignes et M colonnes. En début de partie, les grenouilles sont attribuées à l'un des joueurs, les crapauds à l'autre. Le détenteur des grenouilles commence, et ensuite les joueurs jouent à tour de rôle. Le premier qui ne peut plus avancer ses batraciens a perdu.

Les grenouilles avancent horizontalement de la gauche vers la droite, et les crapauds de la droite vers la gauche.

Si la case adjacente dans le sens de déplacement est libre, la grenouille (respectivement le crapaud) déplacée vient occuper cette case. Si, par contre, elle est occupée par un crapaud (respectivement une grenouille) et que la case suivante est libre, le batracien vient occuper cette case libre. Dans tous les autres cas, le batracien ne peut être déplacé.

**Exemple avec un plateau de 1 \* 5 :**

1. Plateau initial

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **G** | **G** |  | *C* | *C* |

1 2 3 4 5

1. La grenouille de la case 2 avance en 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **G** |  | **G** | *C* | *C* |

1. 2 3 4 5

1. Le crapaud de la case 4 avance en 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **G** | *C* | **G** |  | *C* |

1 2 3 4 5

1. La grenouille de la case 3 avance en 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **G** | *C* |  | **G** | *C* |

1. 2 3 4 5

1. Le crapaud de la case 5 avance en 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **G** | *C* | *C* | **G** |  |

1. 2 3 4 5

1. La grenouille de la case 4 avance en 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **G** | *C* | *C* |  | **G** |

1. 2 3 4 5

1. Le joueur qui détient les crapauds a perdu puisque ceux-ci ne peuvent plus avancer.

### Description du casse-tête des grenouilles et des crapauds

Les règles de déplacement sont identiques à celles du jeu. Cependant le joueur n’est pas contraint dans le choix des déplacements des grenouilles et des crapauds, il peut alors déplacer plusieurs fois le même pion. Le but de l'unique joueur est d'amener toutes les grenouilles à droite et tous les crapauds à gauche du plateau. Si cette disposition est atteinte, la partie de casse-tête est gagnée.

Exemple de disposition gagnante :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *C* | *C* |  | **G** | **G** |
|  |  | | |  |
| **G** | *C* | *C* |  | **G** |

Exemple de disposition perdante :

Optionnel :

Un chronomètre pourra être afficher pour montrer combien de temps le joueur va passer sur le casse-tête à la fin le temps pourra être sauvegarder dans un fichier texte pour être réafficher lors de la prochaine partie comme temps à battre.

### Application à développer

L’application sera dotée d’une interface graphique qui permettra à son lancement de proposer à l’utilisateur d’afficher une aide pour son utilisation.

Mais l’application devra permettre :

* À deux joueurs de s’affronter
* À un unique joueur de jouer contre l’ordinateur
* À un unique joueur d’utiliser le casse-tête
* De gérer des configurations initiales

Dans le cas d’une partie entre deux joueurs, ceux-ci devront saisir leur nom ce qui permettra ensuite à l’application de bien spécifier lequel des deux dois jouer. Les grenouilles et les crapauds seront attribués aux deux joueurs de manière aléatoire. En fin de partie, le nom du gagnant sera clairement affiché.

Dans le cas d’une partie contre l’ordinateur, ou bien lors de l’utilisation du casse-tête, l’application proposera à l’unique joueur d’entrer son nom. S’il n’en donne aucun, il sera désigné par un « joueur humain ». En fin de partie, un message indiquera le résultat de celle-ci et spécifiera le nom de l’unique joueur.

Après une partie, l’application affichera à nouveau le menu principal, ce qui permettra à l’unique joueur ou aux deux joueurs de lancer, éventuellement, une nouvelle partie.

Optionnelle :

Une musique douce pourra alors être mis durant la partie de jeu.

### Configurations initiales

Une **configuration** est la description d'un plateau dans un état précis. Elle indique donc la taille du plateau (nombre de lignes et de colonnes) et la disposition des grenouilles et des crapauds présents sur le plateau. La taille du plateau aura une limite supérieure à fixer (liée aux contraintes d’affichage).

Pour spécifier la configuration initiale d’une partie, l'utilisateur choisira l'une des deux alternatives suivantes :

* Entrer au clavier cette configuration.

* Sélectionner une configuration préétablie. Par conséquent, le programme devra permettre de gérer un ensemble de configurations préétablies que l'utilisateur pourra modifier à son gré par insertion ou retrait d'une configuration. En particulier, lorsque l'utilisateur aura saisi une configuration initiale, le programme lui demandera s'il souhaite l'inclure dans l'ensemble des configurations préétablies.

Le placement des batraciens sur le plateau est libre. Toutefois, l’application vérifiera qu’il y a autant de batraciens de chaque catégorie et que la configuration entrée n’est pas gagnante. Si ce n’est pas le cas, la configuration sera considérée comme invalide.

Exemple de configuration initiale :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **G** |  |  | *C* |  | *C* |
|  | *C* | **G** |  | **G** | **G** |  |
| **G** | **G** |  |  | *C* | *C* |  |
|  | **G** | *C* | **G** |  | *C* | ***G*** |
|  | **G** | *C* |  | *C* | *C* |  |

Les configurations préétablies devront être stockées en mémoire permanente (On utilisera un fichier texte pour stocker les configurations préétablies) afin d’être retrouvées lors du prochain lancement de l’application.

### L’IA

Un joueur pourra affronter une IA en 1 contre 1, cette IA devra réaliser des taches les plus simple :

* Déterminer où se situe ses pions
* Déterminer les cases vide autour d’un pion
* Avancer le pion sur une case vide

Il y aura une IA simplifier

* Facile : IA réalisant les taches simple vue au-dessus de façons aléatoires.

Optionnelle :

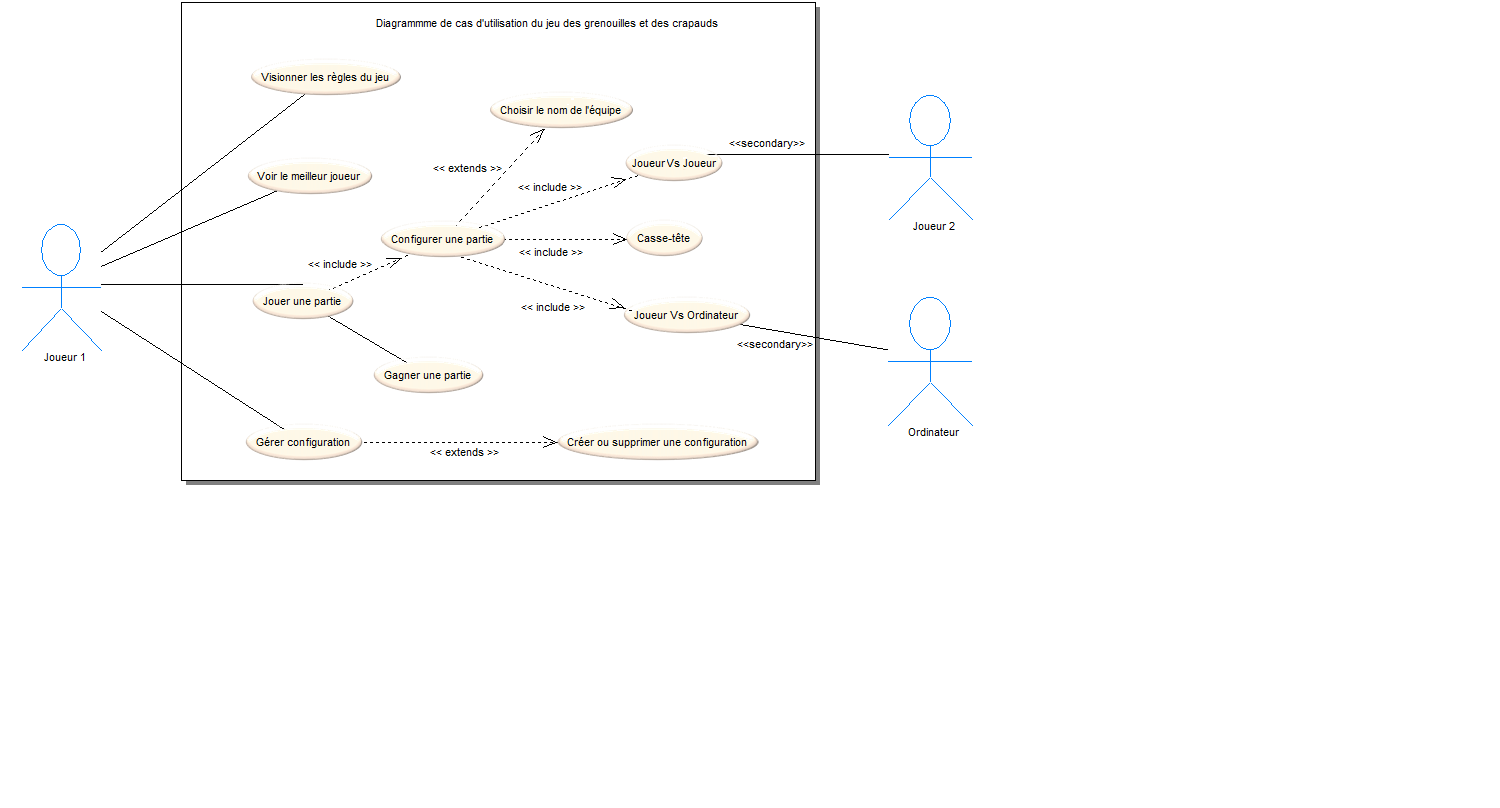
* Moyenne : IA réalisant les taches simple vue au-dessus de façon moins aléatoires.

### Remarque

L’application fonctionnera en mode console ou bien sera dotée d’une interface graphique, au choix du groupe réalisant le projet.

# Documents d’analyse

Voici le diagramme des cas d’utilisation :



# Spécification détaillée

**Identification**

**Nom** :  Le jeu des grenouilles et des crapauds

**Objectif** :  Détailler les étapes et vérifications qui se succèdent quand un joueur lance le jeu des grenouilles et des crapauds

**Acteur** **principal** :  Le joueur

**Date** : 08/06/19

**Responsable** : Yoan Guiraud

**Version** : 3.0

**Séquencement**

Le cas débute lorsque le joueur lance le jeu des grenouilles et des crapauds et arrive dans le menu du jeu :

**Précondition** : L'utilisateur possède le jeu des grenouilles et des crapauds et lance l’exécutable du jeu (executable.bat).

**Enchaînements** **nominaux** : Après que l’utilisateur ait ouvert le jeu des grenouilles et des crapauds.

1. Le joueur arrive dans le menu principal du jeu.

2. Il clique sur le bouton “Jouer”.

3. L’application affiche l’écran “Configuration partie ”.

4. Il choisit le mode de jeu Joueur contre Joueur (mode de jeu par défaut).

5. Il choisit un nom d’équipe pour lui et l’autre joueur, choisit une configuration sauvegardée dans l’application et valide.

6. L’application affiche la partie dans l’écran “Déroulement partie”.

7. Il arrive dans la partie créée selon sa personnalisation et peut commencer à jouer.

8. Le joueur gagne la partie et accède à la fenêtre VICTOIRE avec la visualisation de son nom d’équipe en dessous de « victoire ».

9. Il clique sur le bouton “Retour menu” et revient au menu de l’application.

**Enchaînements** **alternatifs :**

**Point 2 de la séquence nominale :** Le joueur clique sur le logo en forme de triangle dans un cercle du menu principal. Cette icône permet au joueur d’accéder ou de revenir dans le menu principal. Il arrive donc au cas N°1.

**Point 2 de la séquence nominale :** Il clique le bouton “Comment jouer ?” du menu principal. Puis la fenêtre détaillant les règles du jeu s’ouvre, il peut appuyer sur le bouton “OK” ou la petite croix en haut à droite de la fenêtre. Retour au cas N°7.

**Point 2 de la séquence nominale :** Il clique sur le bouton “Quitter” ou sur la croix en haut à droite du menu principal et l’application se ferme.

**Point 2 de la séquence nominale :** Il clique sur l‘engrenage situé dans la partie haute du menu principal.

- L’écran de “Que voulez-vous faire ?” s’affiche.

Première possibilité :

- Le joueur clique sur le bouton “Ajouter”.

- L’application affiche l’écran de “Création configuration”.

- L’utilisateur doit choisir le nom de la configuration à créer et la taille du plateau en entrant le nombre de lignes et de colonnes qui le compose.

- Le joueur clique le bouton “OK”.

- L’application affiche l’écran de “Placement pion”.

- Le joueur place par la suite les pions sur le plateau en entrant la ligne et la colonne de chaque pion, ainsi que leur types (C ou G). Il peut aussi contrôler le placement des pions grâce aux boutons “Ajouter” et “Supprimer”.

- Enfin l’utilisateur clique sur le bouton “Valider”.

- L’application affiche un message box indiquant que la modification a été enregistrée.

- Le joueur clique sur le bouton “OK” ou la croix en haut à droite.

- Retour au cas N°1.

Deuxième possibilité :

- Le joueur clique sur le bouton “Supprimer”.

- L’application affiche l’écran de “Suppression configuration”.

- Le joueur choisit la configuration à supprimer en entrant son numéro d’emplacement.

- L’application affiche une message box en indiquant que la configuration a été supprimée.

- Le joueur clique sur le bouton “OK” ou la croix en haut à droite.

- Retour au cas N°1.

**Point 2 de la séquence nominale :** Il clique sur la coupe située dans la partie haute du menu principal. Puis la fenêtre montrant le meilleur joueur de l’application s’ouvre. Retour au cas N°7.

**Point 2 de la séquence nominale :** Il clique sur le bonhomme situé dans la partie haute du menu principal. Puis la fenêtre des créateurs du site qui détaillent les acteurs de cette application s’ouvre. Retour au cas N°1.

**Point 4 de la séquence nominale :** Il choisit le mode de jeu Casse-tête. Le joueur peut choisir le nom de son équipe et choisir une configuration et valide. Puis il arrive au cas N°6.

**Point 4 de la séquence nominale :** Il choisit le mode de jeu Joueur Vs Ordinateur. Le joueur peut choisir le nom de son équipe, choisir une configuration et valide. Puis il arrive au cas N°6.

**Point 4 de la séquence nominale :** Le joueur choisit une configuration par défaut dans l’application et valide. Puis il arrive au cas N°6.

**Point 6 de la séquence nominale :** Le joueur gagne la partie et accède à la fenêtre VICTOIRE avec la visualisation du nom de l’équipe de l’ordinateur en dessous de « victoire ». Puis il arrive au cas N°9.

**Enchaînements d'exception :**

**Point 4 de la séquence nominale** : Dans l’écran “Configuration partie”, le joueur doit valider sa configuration ajoutée ou supprimer grâce au système de sauvegarde de configuration, mais il peut aussi l’annuler et retourne au menu principal.

Il doit aussi entrer des coordonnées valides dans la sélection de la ligne et la colonne d’un pion, sinon une erreur se produit.

La limitation maximale des lignes et des colonnes dans un plateau est de 20. Une erreur se produit, si on entre un nombre de lignes et de colonnes invalides, si on entre le nombre d’une configuration invalide.

**Post condition :**

La navigation sur les pages qui s’effectue sera toujours vrai.

Le nombre de crapauds et grenouilles sur le plateau seront toujours équivalents.

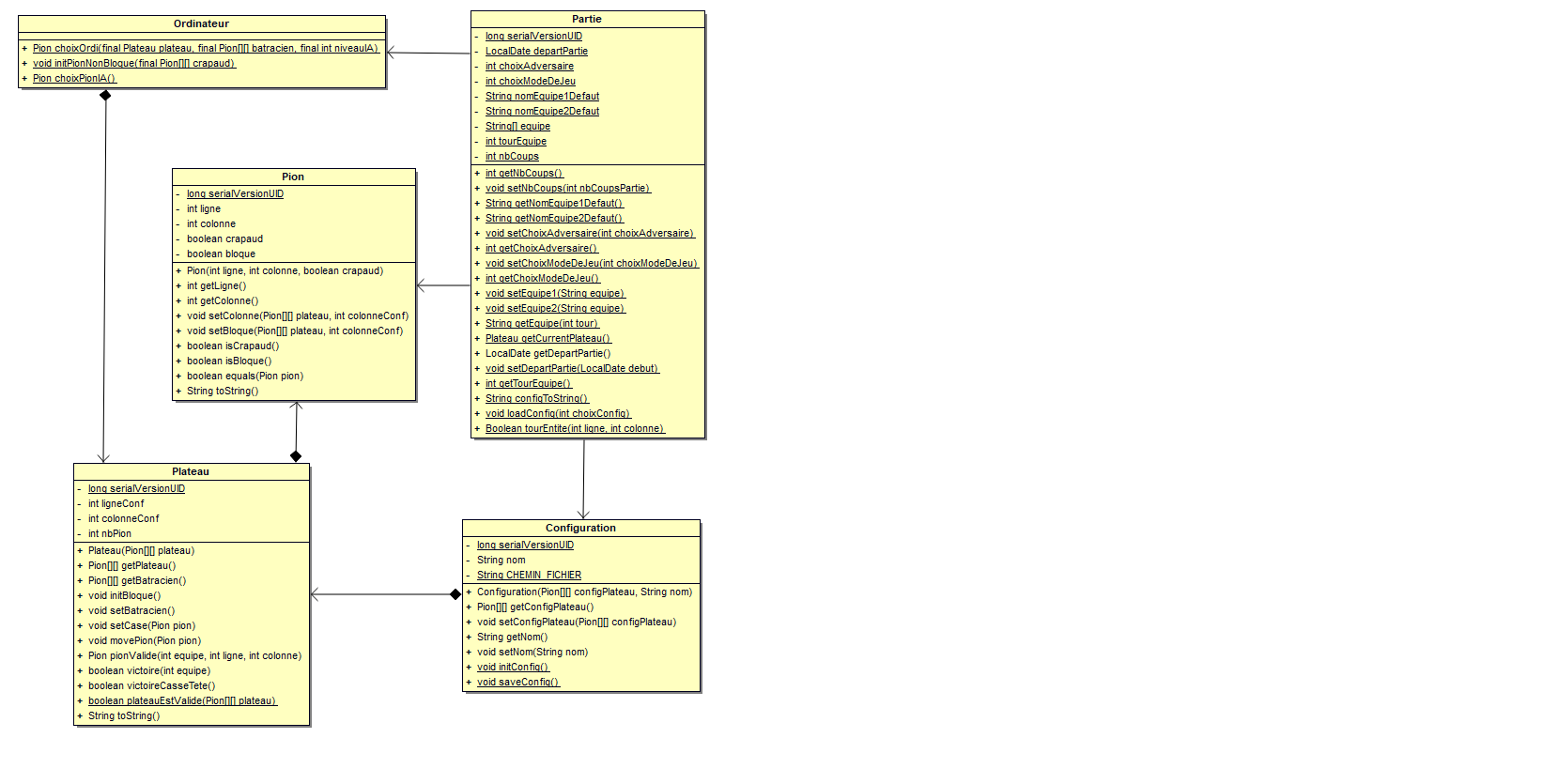
C’est toujours l’équipe des grenouilles qui commence.

**Remarque :**

Le choix de l’interface pour les résultats du jeu et de la création d’autres boutons dans le menu principal sont possibles de changer dans le futur, ainsi que l'agencement de certaines pages entre certaines actions

# Diagramme de classe

## Partie 1 :



Explication des principaux éléments du diagramme de classe :

* La classe Ordinateur est compose de la classe plateau si celle si venait à disparaitre la classe Ordinateur ne serait plus fonctionnel car celle-ci se base intégralement sur la copie et la recherche de solution dans des plateau
* La classe plateau est composé de la classe Pion en effet la classe Plateau créer un tableau de Pion pour définir un plateau de jeu
* Une configuration est un plateau enregistre donc la classe Configuration est composée de la classe Plateau
* La classe Partie fait appel fait appel à la classe Configuration (lors de du chargement d’un plateau au commencement d’une partie), Pion (lors de la création d’une configuration) et la classe Ordinateur (lorsque le joueur joue contre l’ordinateur la classe Ordinateur est appelle pour joueur).



## Partie 2 :

Explication :

* Nous avons séparé le diagramme de classe en 2 parties distinctes pour le rendre plus facile à la compréhension
* La classe GestionPartieInterface fait appel a la classe Partie lorsque l’interface se lance et que l’utilisateur souhaite faire une partie / créer une configuration
* La classe GestionPartieInterface fait appel a la classe InterfaceAppliController pour gérer le fonctionnement de l’interface.
* Cette même classe fait appel a la classe Outils qui est une classe compose de l’ensemble des fonctions de vérification et de procedure (ex : enregistrer la config, mettre a jour le plateau …)

# Répartition des Taches

|  |  |  |
| --- | --- | --- |
| Nom | Rôle | Taches |
| Pierre marie Combalbert | Gestionnaire de configuration | Ces tâches sont :   * Choisis les applications et configurations de celle-ci qui devront être utilisés dans le projet * Programmation de l’interface de notre jeu vidéo. * Programmation du code permettant de gérer l’interface |
| Maël Le folgoc pontis | Contrôleur qualité | * Programmation du code des objets du jeu. * Programmation du code pour les parties du jeu. * Vérification de la qualité de notre code. |
| Yoan Guiraut | Secrétaire | * Tenir à jour les documents écrits. * Faire les comptes rendus pour chaque réunion. * Aide dans la rédaction de documents |
| Yanis Delmas | Chef de projet | * Coordonnées l’équipe. * Rédiger un plan projet * Aide dans la programmation |

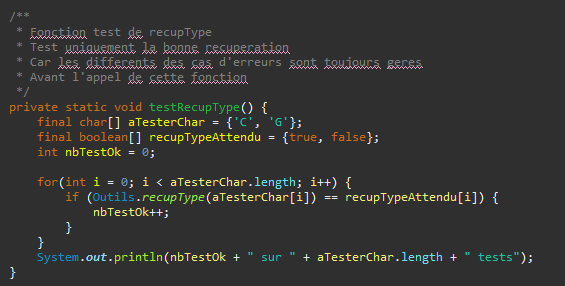
# Liste des objectifs

|  |  |  |
| --- | --- | --- |
| Parti | Objectif | Atteint |
| Générale |  |  |
|  | Impossibilité qu’un crapaud puise sauter au-dessus d’une grenouille et vice versa | Oui |
|  | Le déplacement vers la droite ou vers la gauche suivant le type de pion. | Oui |
|  | Début des parties toujours par les grenouilles | Oui |
|  | Vérification de victoire | Oui |
| Joueur contre joueur |  |  |
|  | Choix du nom des joueurs | Oui |
| Joueur contre ordinateur |  |  |
|  | Création d’un ordinateur | Oui |
| Casse-tête |  |  |
|  | Déplacement des pions libres | Oui |
| Configuration |  |  |
|  | Configuration de base | Oui |
|  | Ajouter une configuration | Oui |
|  | Supprimer une configuration | Oui |
|  | Vérification configuration gagnante | Oui |
| Interface |  |  |
|  | Interface interactif | Oui |
| Supplément |  |  |
|  | Ajout musique | Non |
|  | Ajout de 3 ordinateurs | Non |
|  | Pause entre joueur et ordinateur | Non |

# Les codes

## Tests de outils :

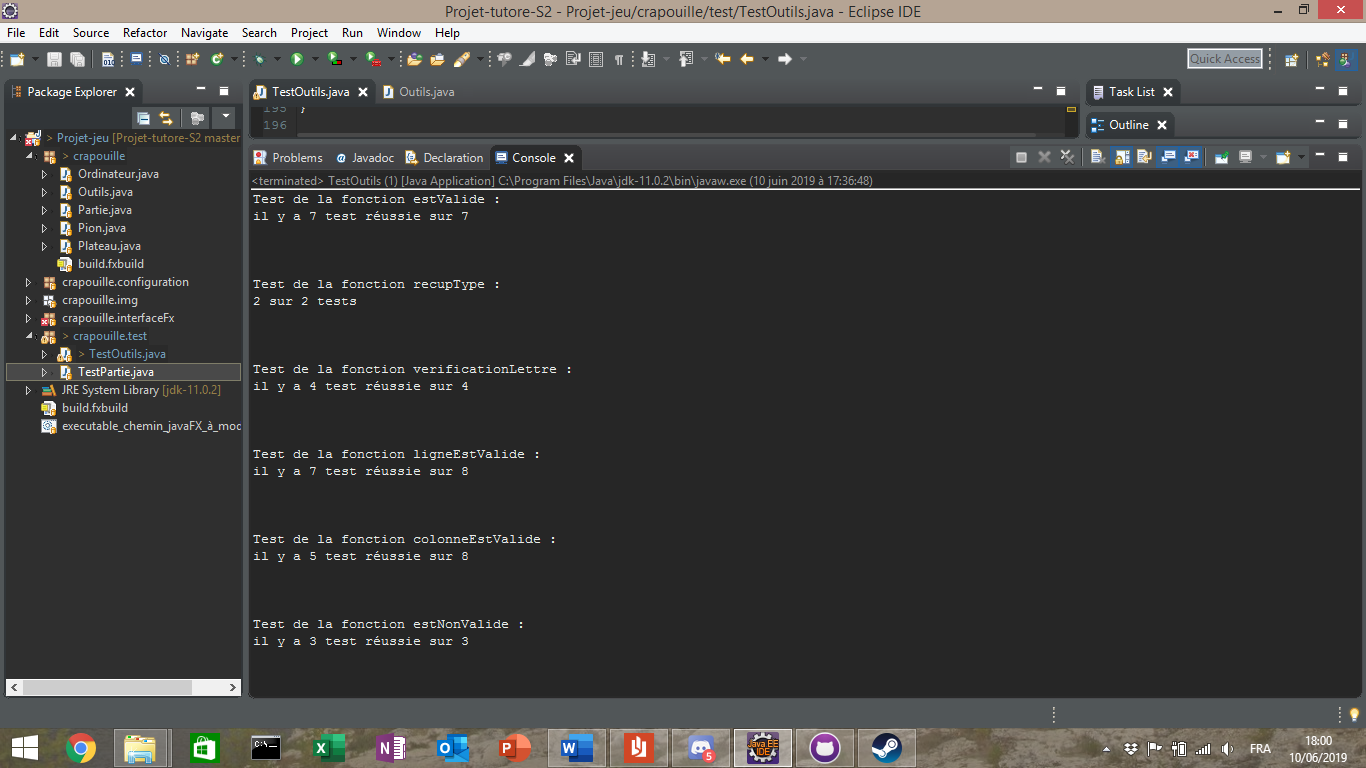
### Test de recupType :



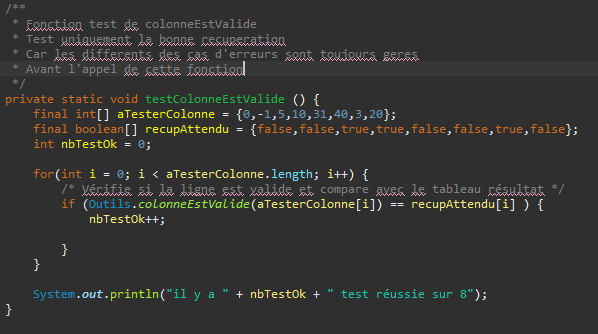
Ce test doit juste servir à vérifier que notre fonction est bien capable de récupérer le type du pion est que celle-ci retourne bien le bon résultat en fonction du type.

Ce test contient bien l’ensemble des possibilités requises à tester (C, G) et vérifie bien que la fonction retourne la bonne valeur attendue (Les différents cas d’erreurs étant teste avant)

Résultat :



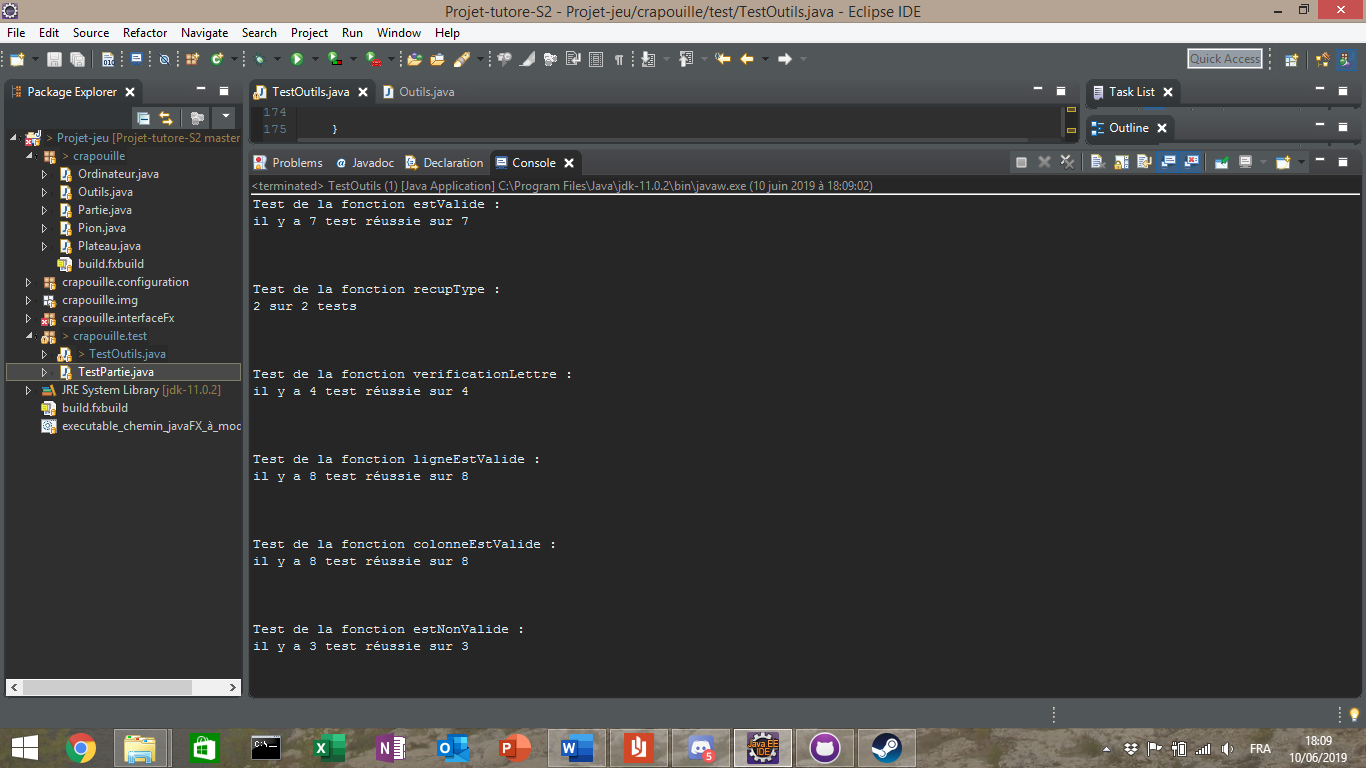
### Test de colonneValide :



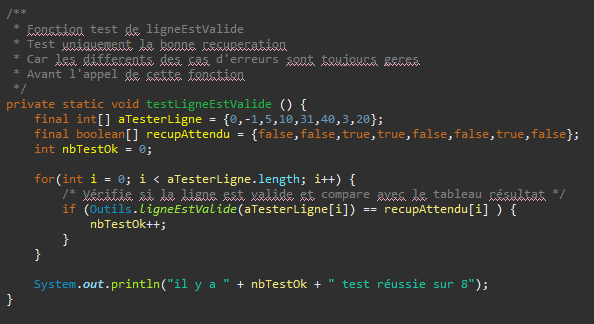
Ce test vérifie que notre fonction est capable d’identifier si la colonne rentrée par l’utilisateur est valide.

Pour cela on test un l’ensemble des cas possible dans cette fonction : chiffre négatif, Trop grand, trop petit et dans l’intervalle valide. Ce qui nous permet de tester tous les cas possibles et vérifier que le fonctionnement de notre fonction de vérification est Valide.

Résultat :



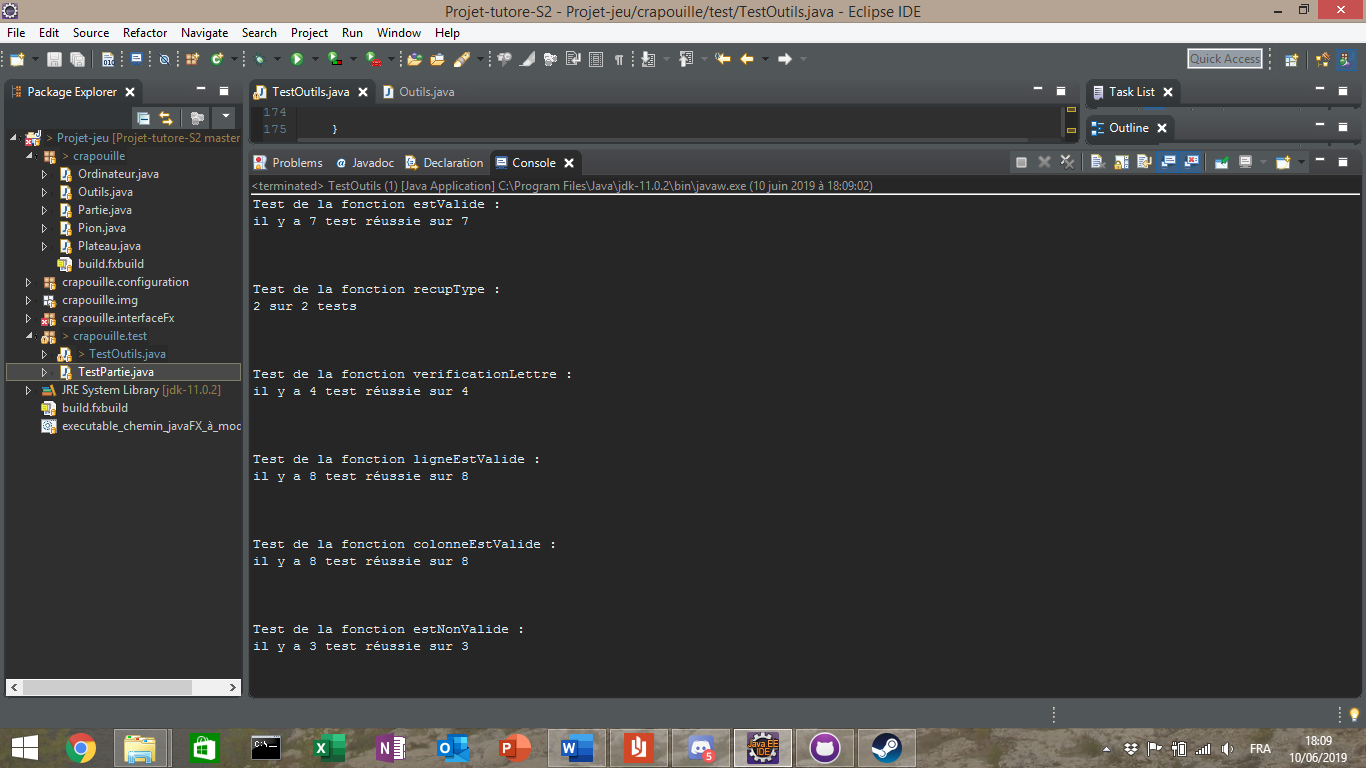
### Test de ligneValide :



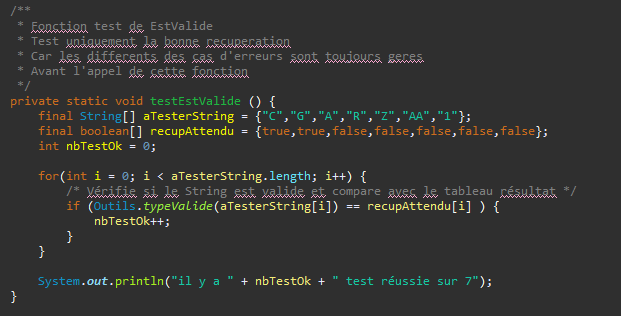
Ce test vérifie que notre fonction est capable d’identifier si la ligne rentrée par l’utilisateur est valide.

Pour cela on test un l’ensemble des cas possible dans cette fonction : chiffre négatif, Trop grand, trop petit et dans l’intervalle valide. Ce qui nous permet de tester tous les cas possibles et vérifier que le fonctionnement de notre fonction de vérification est Valide.

Résultat :



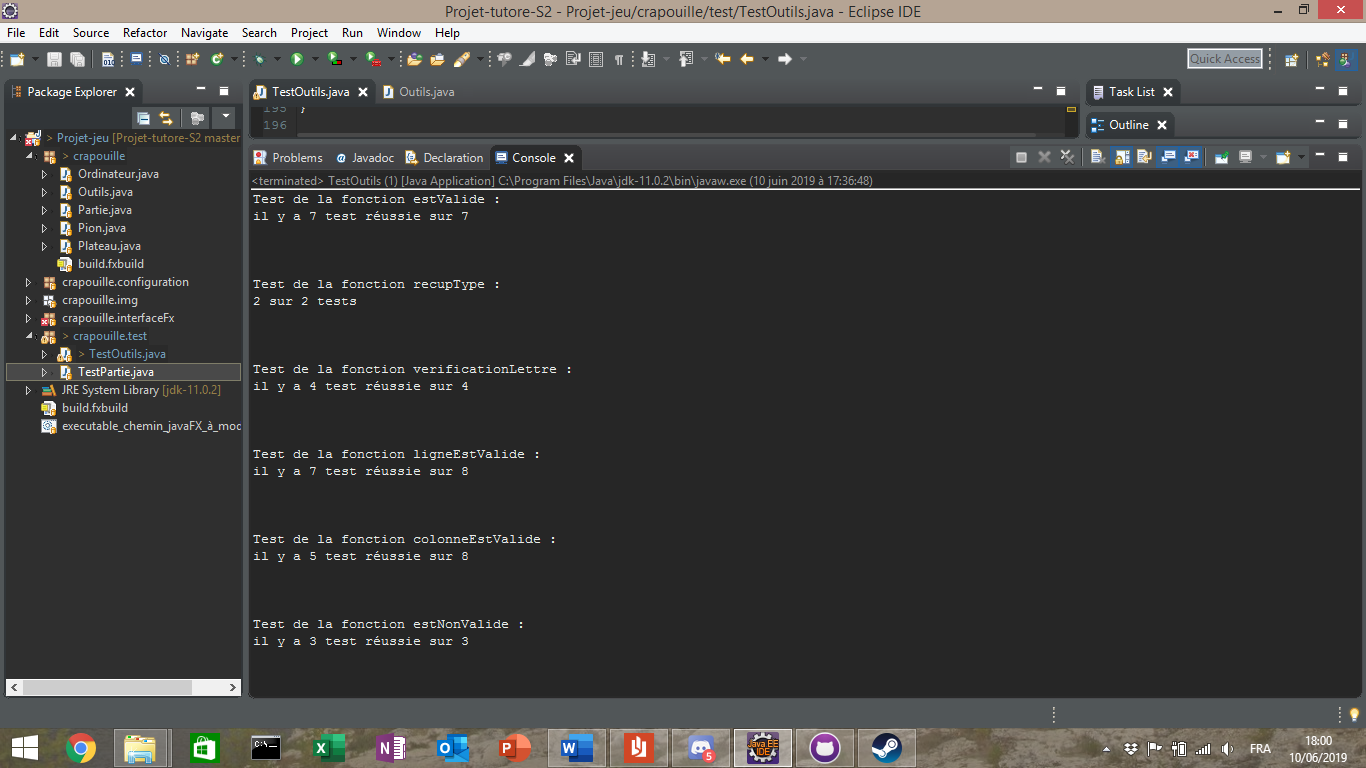
### Test de typeValide :



Ce test vérifie que notre fonction est capable d’identifier si l’utilisateur a rentre un type non valide de pion.

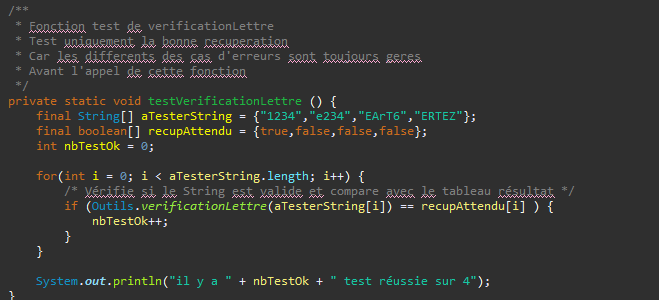
Pour cela on test tous les cas possibles : chiffre, lettre non valide (ne correspondant pas à un type), plus d’un caractère, et le cas ou le caractère est valide. On vérifie que le résultat attendu correspond à celui obtenu et si tous les tests correspondent cela signifie que notre fonction est capable de gérer tous les cas d’erreurs.

Résultat :



## 

### Test verificationLettre :

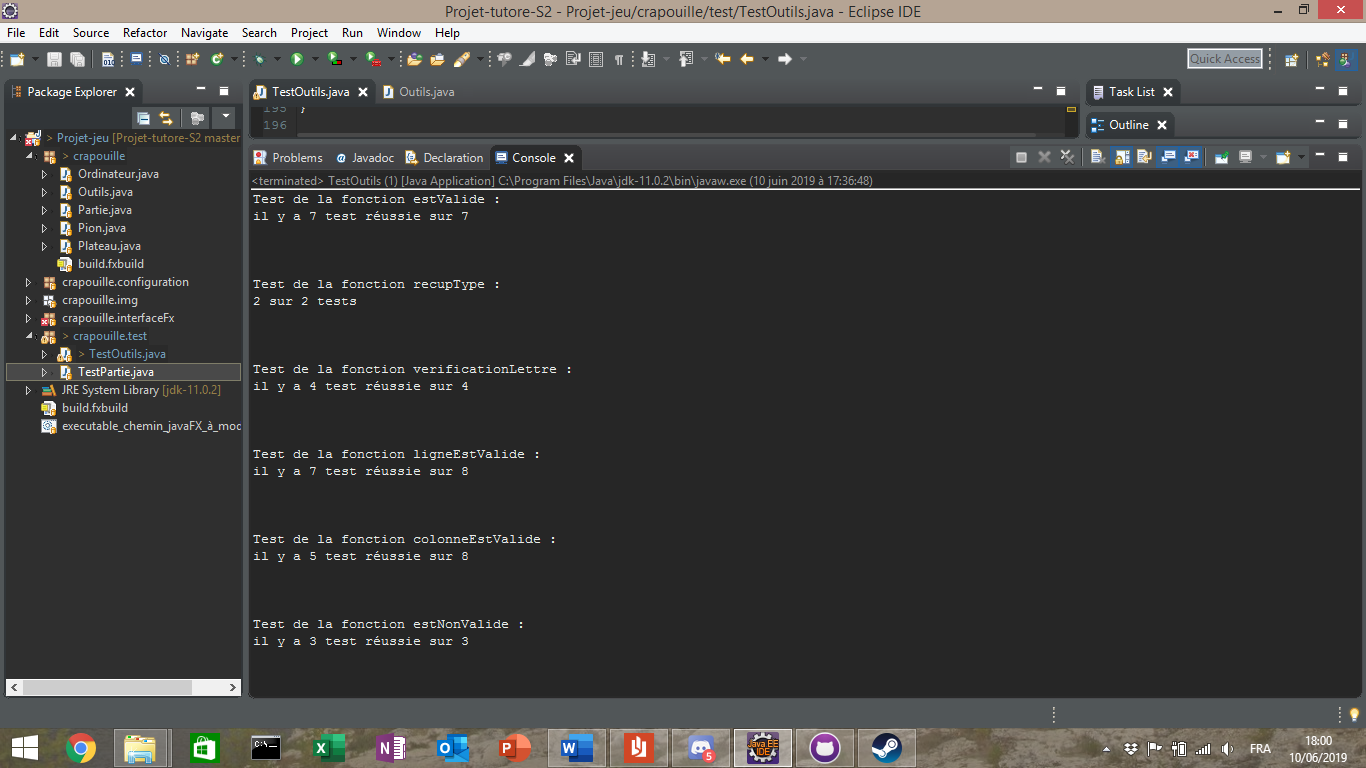


Ce test vérifie que notre fonction est bien capable d’identifier une chaine de caractère avec et sans lettre. Pour cela on test un ensemble de cas possible :

Pas de lettre, que des lettres, combinaison de lettre et de chiffre

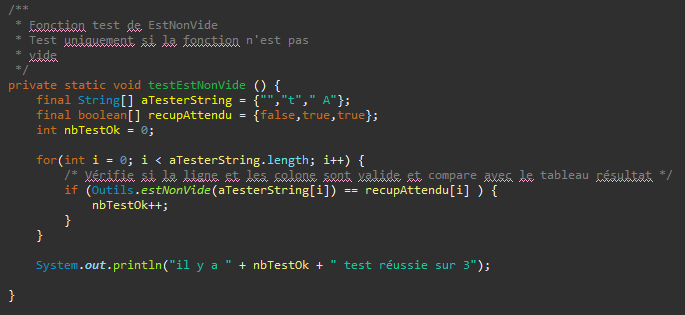
Puis on vérifie que les résultats attendus lors des tests de ces chaines de caractères correspondent a celui attendu pour tester l’ensemble des cas possibles

Résultat :



## 

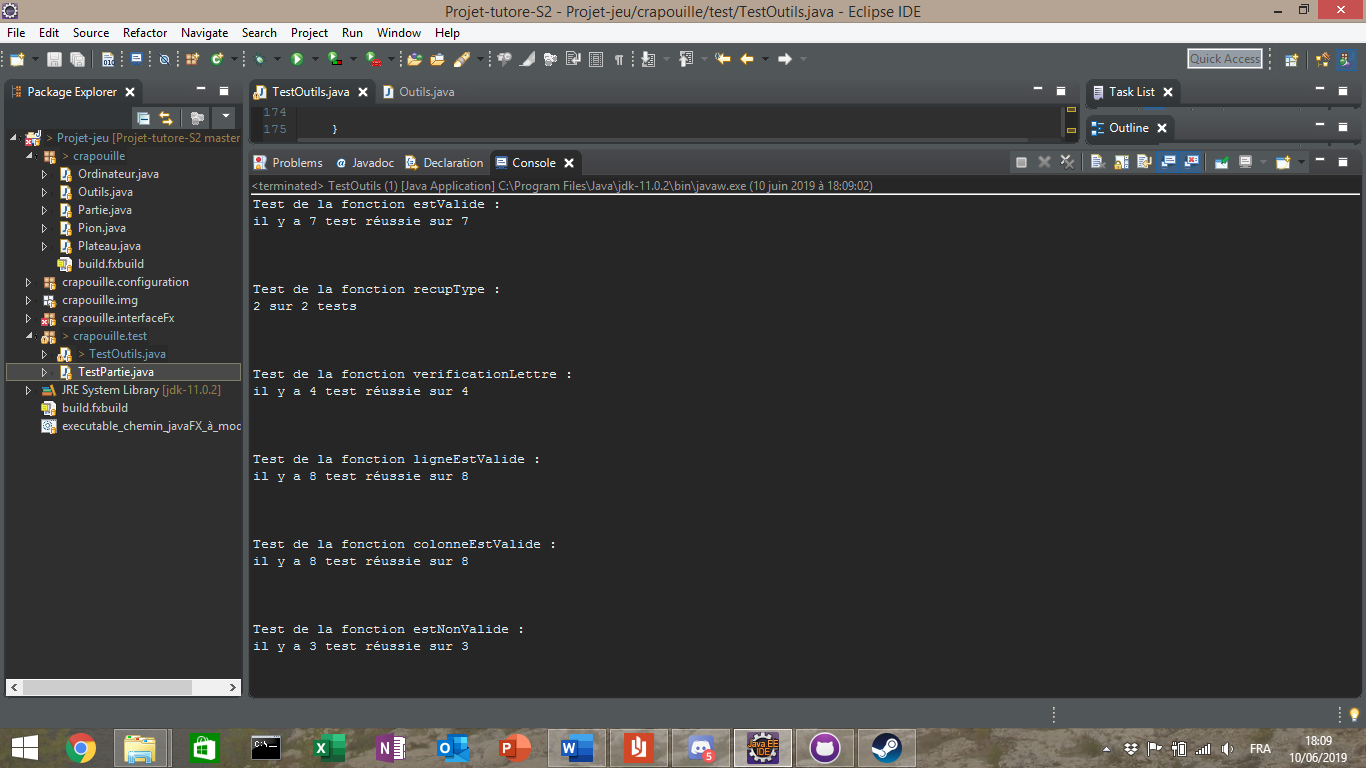
### Test estNonVide :



Fonction qui vérifie si une chaine de caractère est vide ou non. Pour cela on test un ensemble de possibilité : Non vide, vide, test avec un premier caractère vide

Puis on test que tous nos tests renvoient le résultat attendu.

Résultat :



## Test visuel Outils :

### RecupereLignePion, RecupereColonnePion :

(Pas vraiment de tests ou de cas d’erreur a testé cette fonction celle-ci ne peut pas être appelé si l’un des cas d’erreurs pouvant la faire potentiellement crasher est présent)

### afficherConfigDispo :

Récupère l’ensemble des configurations enregistrées et les retourne sous forme de String :

(Utilisation dans l’application :)



### AttribuerNomJ1, attribuerNomJ2 :

Pas de réel test nécessaire car aucun cas d’erreur possible

Exemple attribution nom J1 : (Fonctionnement similaire pour J2 donc exemple de test pour les deux fonctions) :

### Personnalisé :





### Défaut :

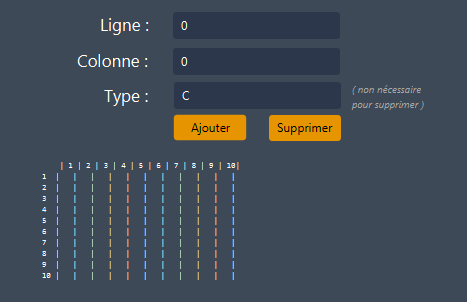




### cordOk :

Vérifie que le pion est bien dans le tableau :

### Essai trop petit :





### Essai trop grand :





### Test configExiste :

Vérifie que le numéro de la config entrée est valide

(Les cas d’erreurs comme les lettres etc. sont testés précédemment la fonction de peut pas rencontrer ces cas d’erreurs)



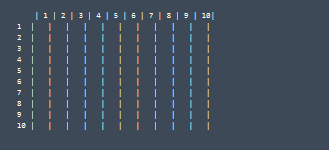


### Test de placementPion :

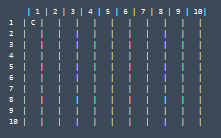
Place le pion entré par l’utilisateur dans le plateau en cours de création

(Les vérifications de validité sont effectuées précédemment)

Avant :



Après :



### Test sauvegarder :

Avant :



Après :



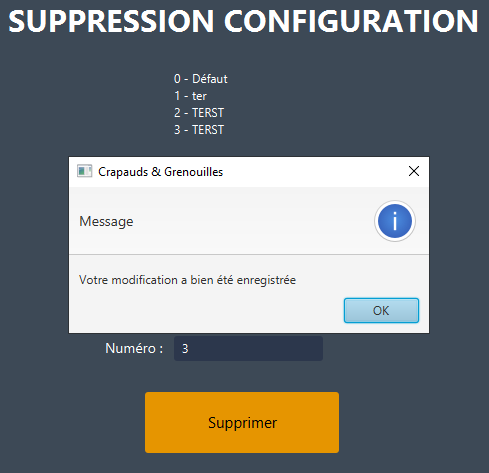
### Test supprimerConfig :

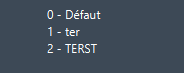
Vérifie que l’entrée de l’utilisateur est correcte puis supprime une config

Avant :



Après :





### Test suppressionPion :

Supprime un pion lors de la création du plateau si l’entrée de l’utilisateur est valide

Avant :



Après :



### VerifVictoire :

Aucun test réel fonction qui retourne un Boolean si la variable victoire indique qu’une équipe a gagné

### configIni :

Appel estNonVide, verificationLettre, coordValide

Pas de test réel appel une série de tests et affiche les messages d’erreurs / exécute la création du plateau si tout est correct (cf. voir les tests des fonctions appelées)

### placementPion :

Appel estNonVide, verificationLettre, cordOk,type Valide

Pas de test réel appel une série de tests et affiche les messages d’erreurs / exécute le code de placement du pion (cf. voir les tests des fonctions appelées)

### configValide :

Appel estNonVide, verificationLettre, configExiste

Pas de test réel appel une série de tests et affiche les messages d’erreurs / Retourne true si tout est valide (cf. voir les tests des fonctions appelées)

### actualisationPlateau :

Vérifie que les éléments entrés par l’utilisateur sont corrects et actualise le plateau de jeu

Appel estNonVide, verificationLettre, ligneEstValide, colonneEstValide

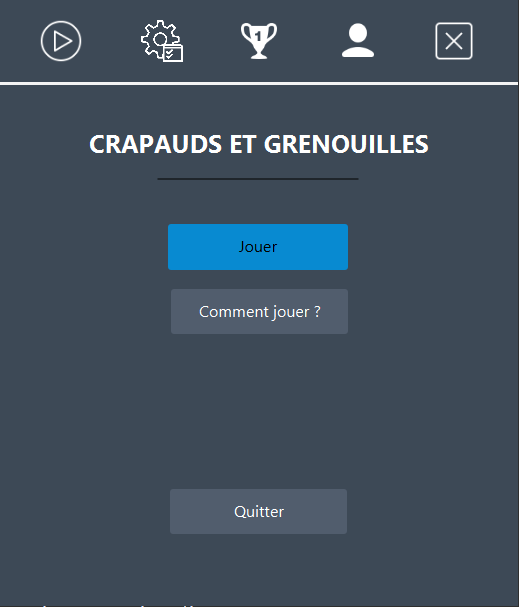
Pas de test réel appel une série de tests et affiche les messages d’erreurs / actualise le plateau de jeu après mouvement du pion (cf. voir les tests des fonctions appelées)

## Test InterfaceController :

### Test showMVP (Pas de réel test) :

Affiche la page MVP

Avant :



Après :



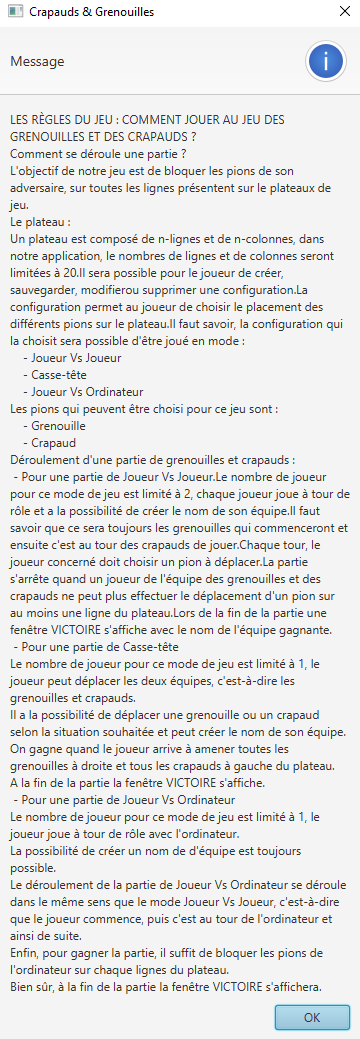
### Test showRegle (Pas de réel test) :

Affiche le texte d’aide

Avant :

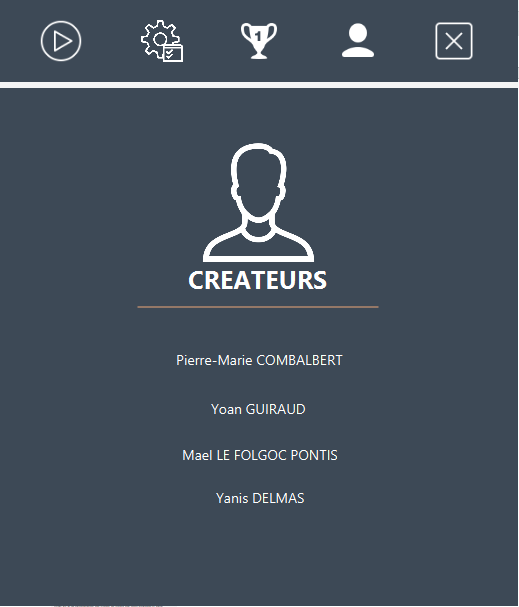


Après :



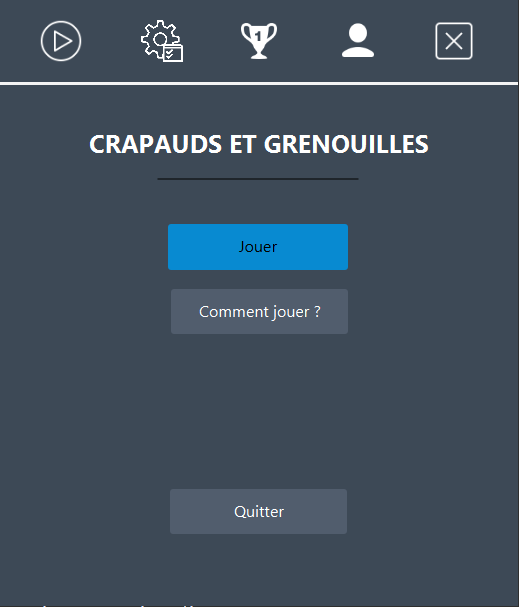
### Test showCreateur(Pas de réel test) :

Affiche page CREATEUR



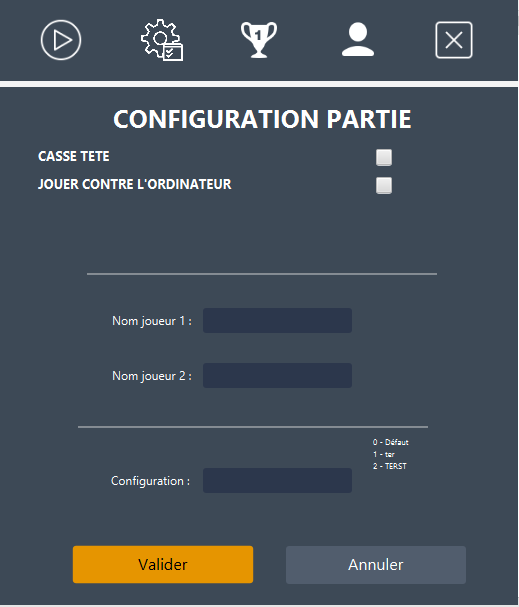
### Test showMenu(Pas de réel test)  :

Affiche menu principale :



### Test showConfigurationPartie (Pas de réel test) :

Affiche configuration de la partie :



### Test showConfiguration (Pas de réel test) :

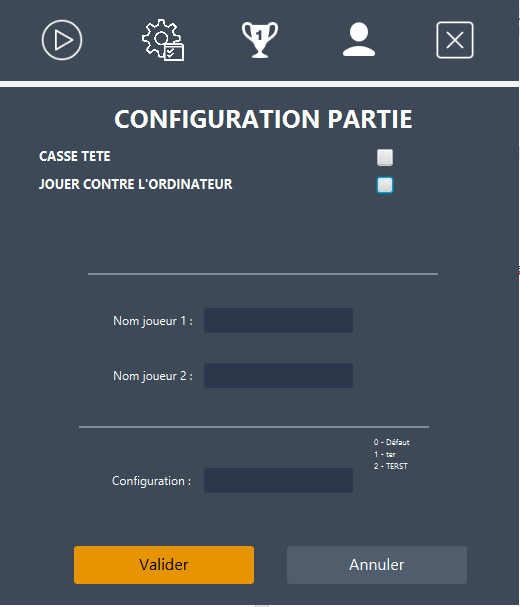
Affiche menue des configurations :



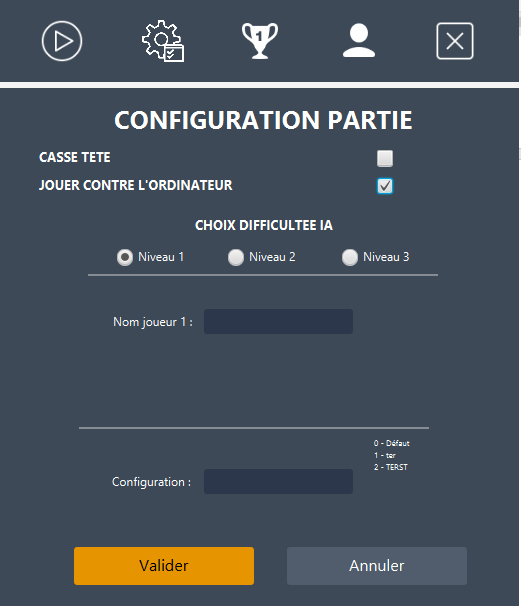
## 

### Test verificationCheck, caseIaCheck, caseIaPasCheck, cacherJ2, cacherIA:

Case IA pas check :

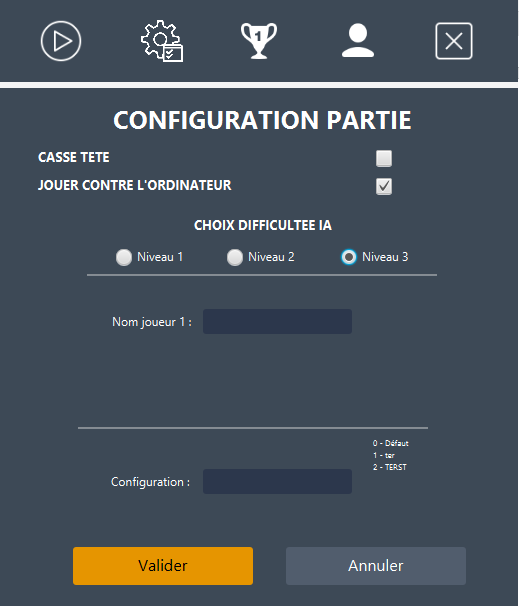


Case IA check :

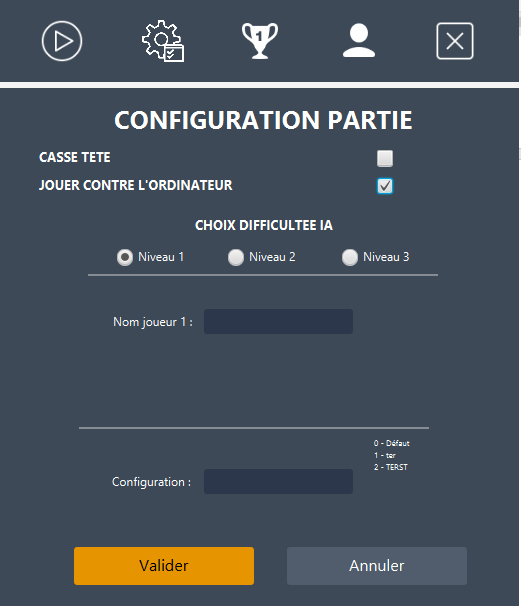


### Test razIA:

Avant :



On cache l’IA puis on la remet et celle-ci c’est bien re mise par défaut



### Test razPartie :

Avant :



On revient au menu puis retourne sur configuration Partie

Après :

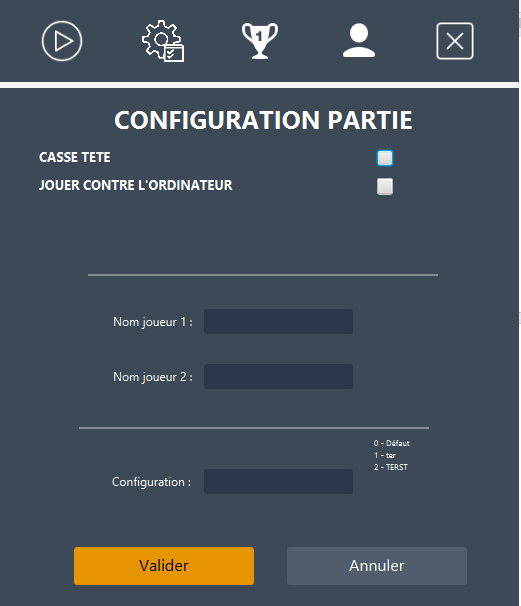


### Test verificationModeJeu, casseTeteConf, versusConf :

Mode casse-tête :

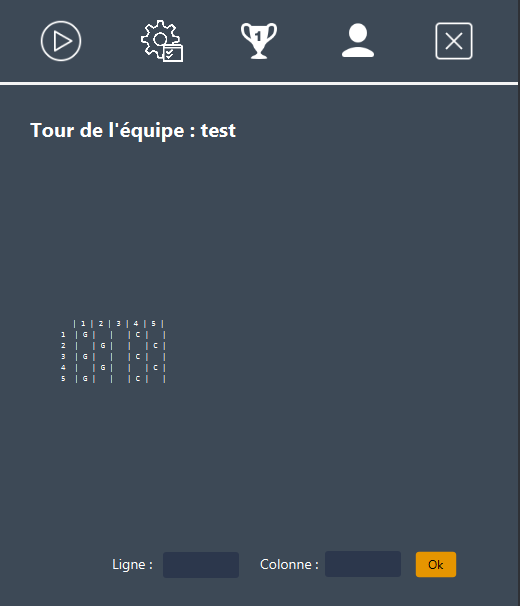


Mode versus :

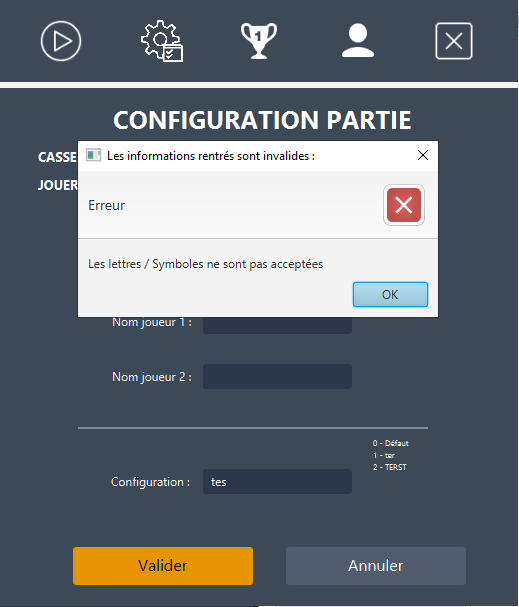
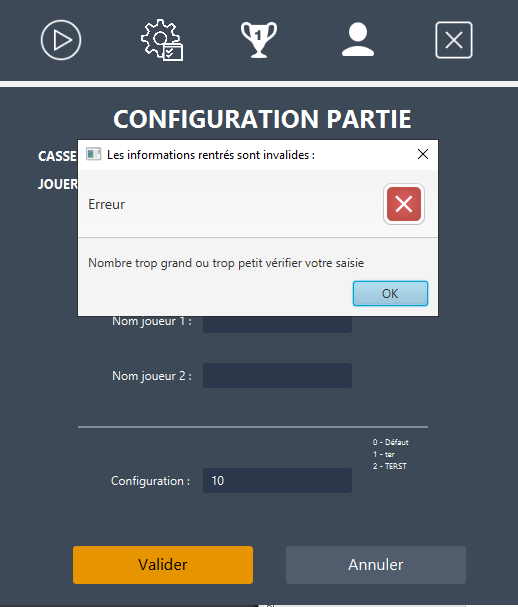


### Test showGameBoard, recupNomEquipe :

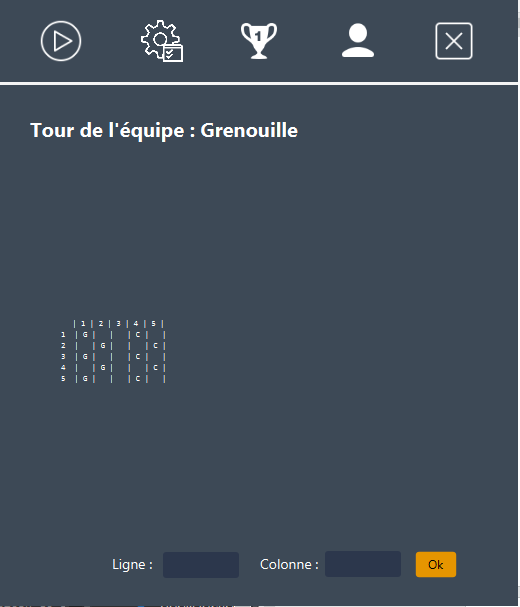
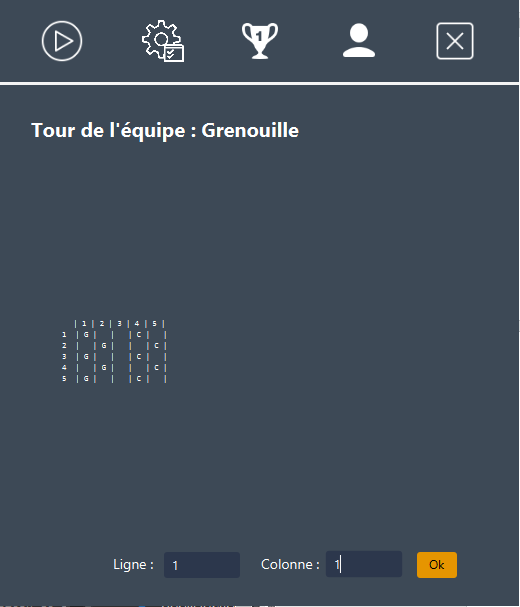
Cas entrée valide :

Cas non valide :

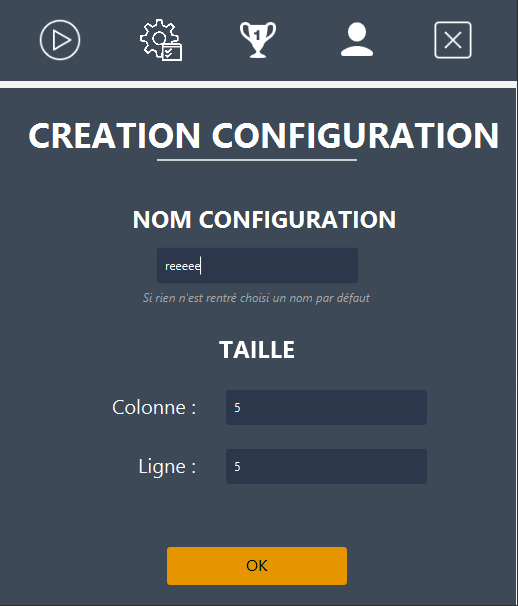
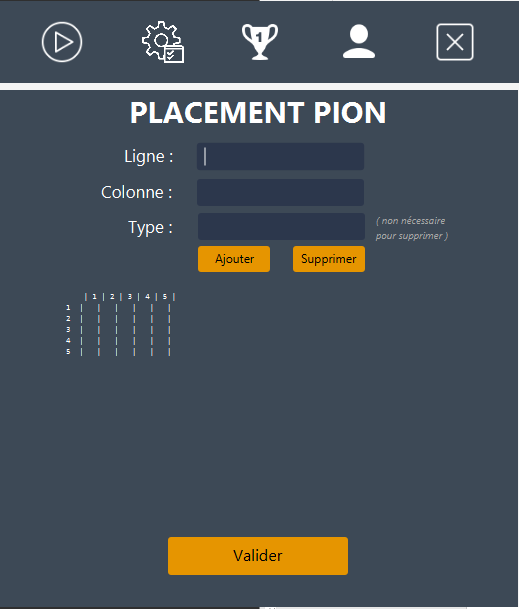
### Test actualiserJeu, rafraichirJeu :

Avant :  
 

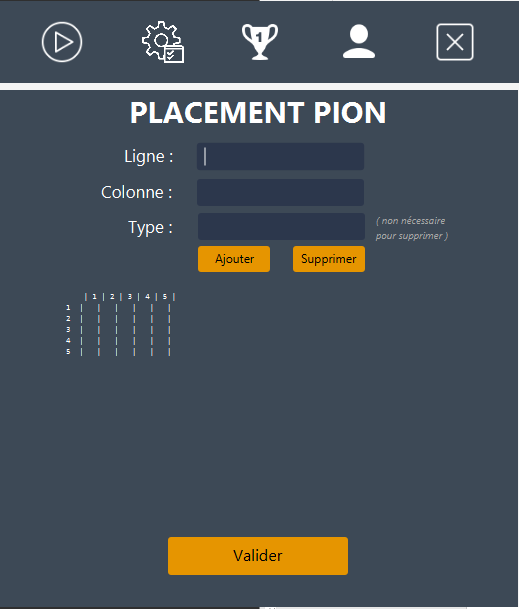
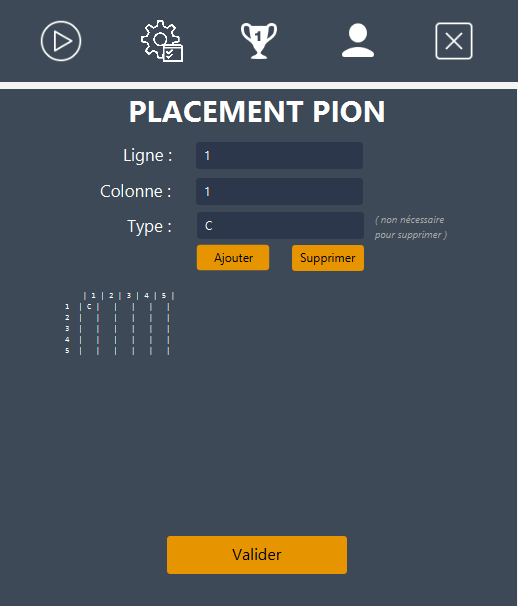
Après :



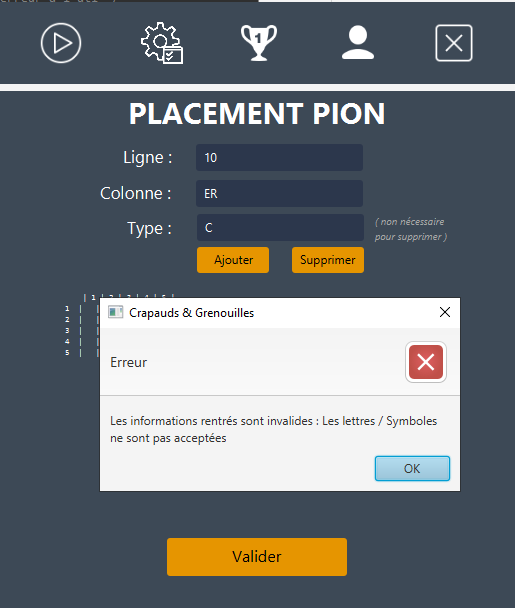
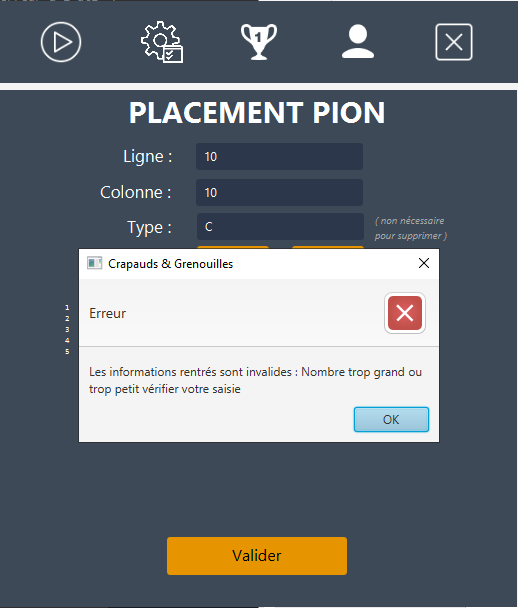
### Test configInitialisation :

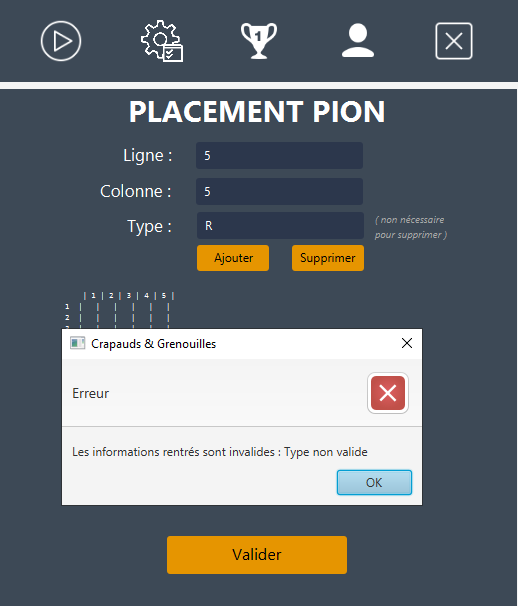
 

### Test ajoutPionConfig :

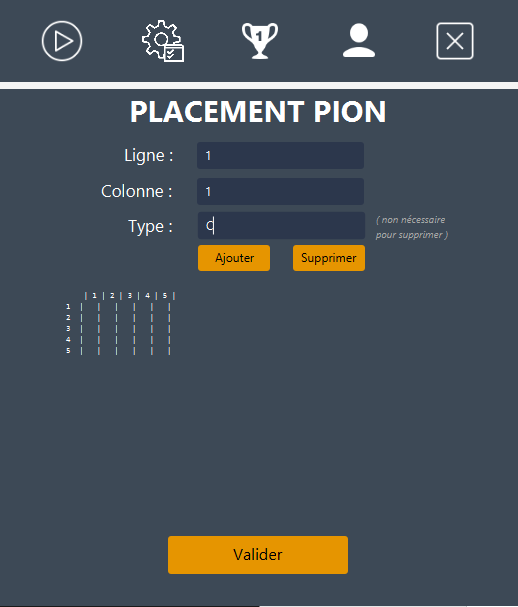
Cas d’erreur vérifier dans la création de configuration :

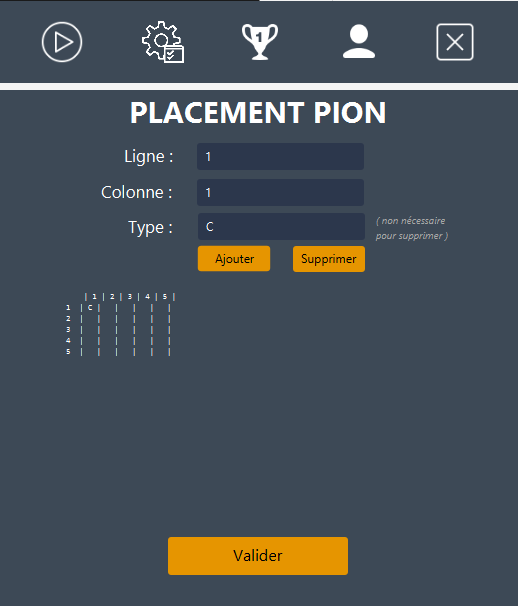
 



### Test rafraichirConf:

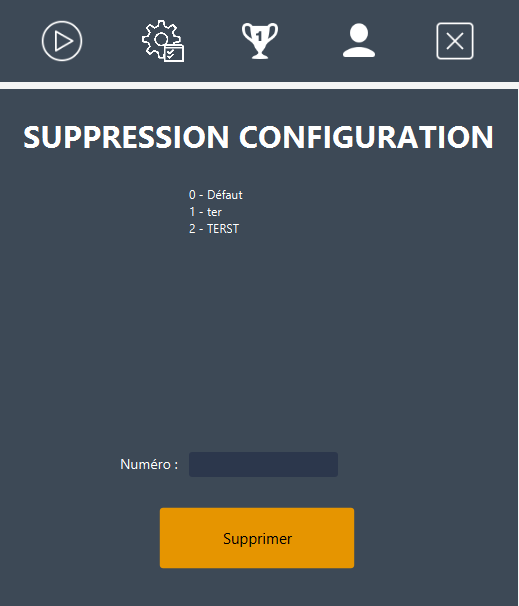
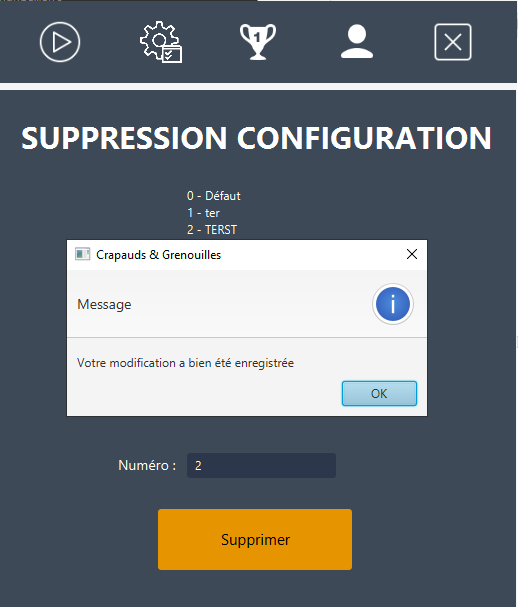
Avant:



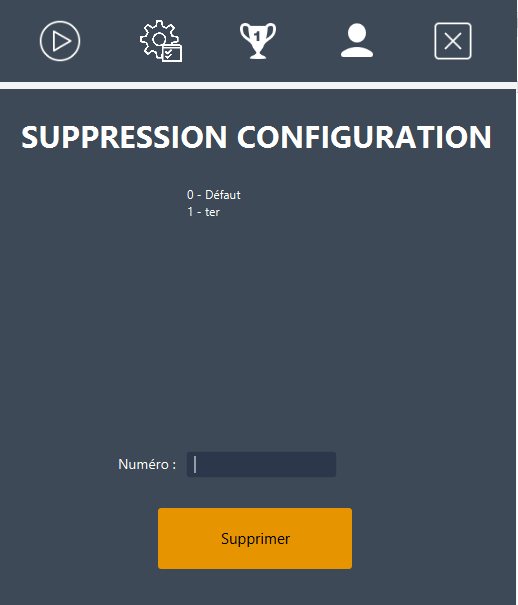


### Test deleteConfig:

Avant :

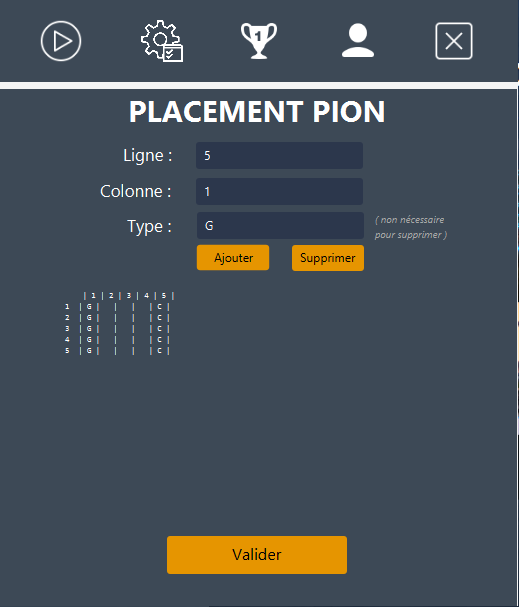
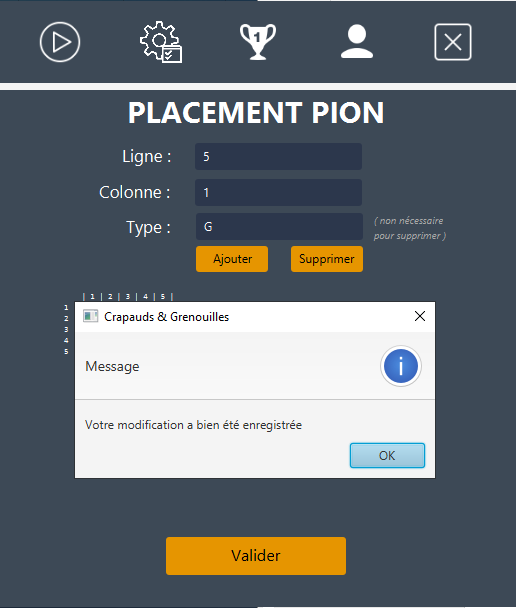
 

Après :

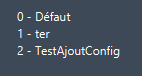


### Test enregistrerConfig :

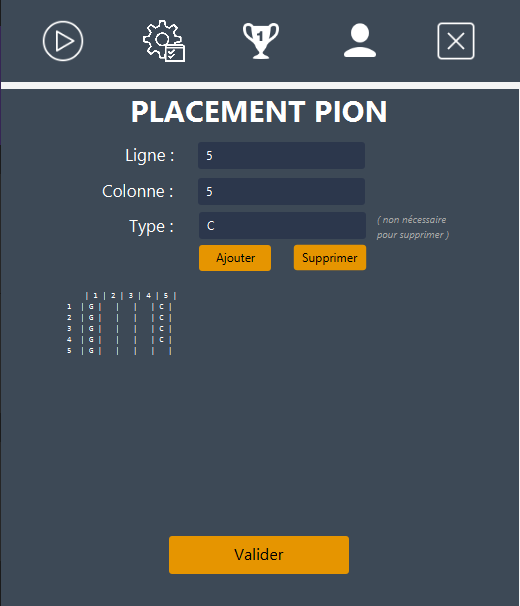
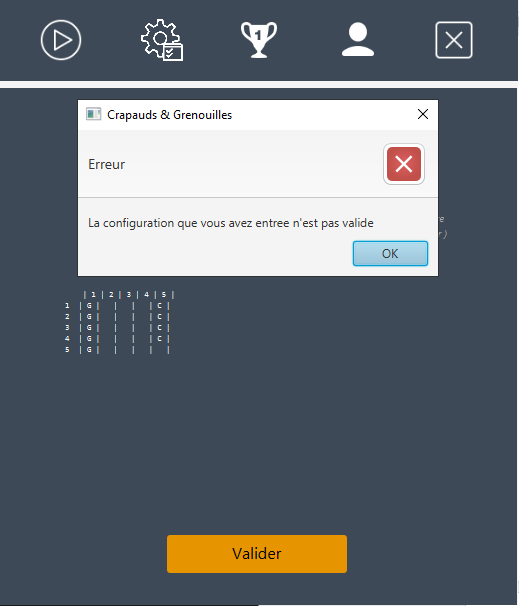
Config valide :

Après ajout :



Config non valide :

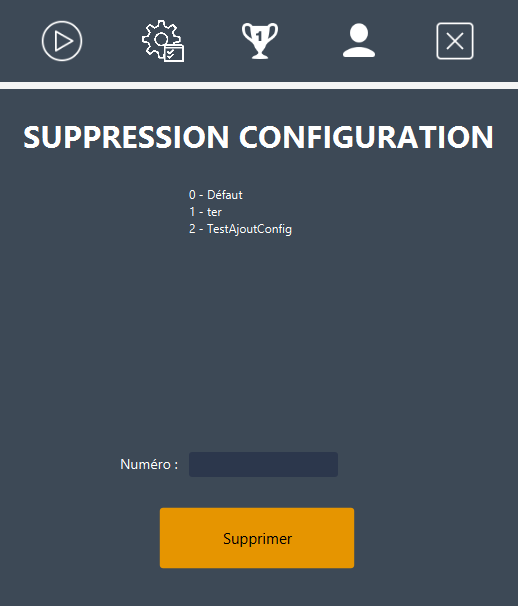
### Test showCreationConfig :

Affiche la page création configuration :



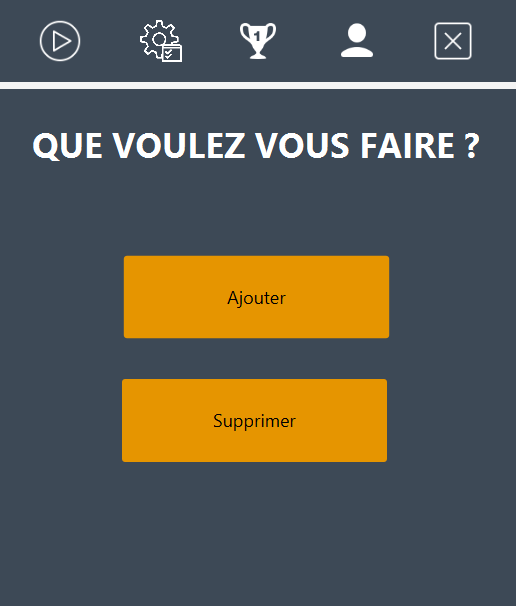
### Test showSupprimerConfig :

Affiche la page supprimer Config



### Test showInitialisationConfig :

Affiche la page de choix :



showInitialisationConfig

Fonctions sans réel test possible :

* Leave
* Reinitialiser
* recupModeJeu
* recupAdversaire
* recupNomConf

# Manuel d’utilisateur

LES RÈGLES DU JEU : COMMENT JOUER

AU JEU DES GRENOUILLES ET DES CRAPAUDS ?

Comment se déroule une partie ?

L'objectif de notre jeu est de bloquer les pions de son adversaire, sur toutes les lignes présentent sur le plateau de jeu.

Le plateau :

Un plateau est composé de n-lignes et de n-colonnes, dans notre application, le nombres de lignes et de colonnes seront limitées à 20.

Il sera possible pour le joueur de créer, sauvegarder, modifier ou supprimer une configuration.

La configuration permet au joueur de choisir le placement des différents pions sur le plateau.

Il faut savoir, la configuration qui la choisit sera possible d'être joué en mode :

- Joueur Vs Joueur

- Casse-tête

- Joueur Vs Ordinateur

Les pions qui peuvent être choisi pour ce jeu sont :

- Grenouille

- Crapaud

Déroulement d'une partie de grenouilles et crapauds :

- Pour une partie de Joueur Vs Joueur.

Le nombre de joueur pour ce mode de jeu est limité à 2, chaque joueur joue à tour de rôle et a la possibilité de créer le nom de son équipe.

Il faut savoir que ce sera toujours les grenouilles qui commenceront et ensuite c'est au tour des crapauds de jouer.

Chaque tour, le joueur concerné doit choisir un pion à déplacer. La partie s'arrête quand un joueur de l'équipe des grenouilles et des crapauds ne peut plus effectuer le déplacement d'un pion sur au moins une ligne du plateau.

Lors de la fin de la partie une fenêtre VICTOIRE s'affiche avec le nom de l'équipe gagnante.

- Pour une partie de Casse-tête

Le nombre de joueur pour ce mode de jeu est limité à 1, le joueur peut déplacer les deux équipes, c'est-à-dire les grenouilles et crapauds.

Il a la possibilité de déplacer une grenouille ou un crapaud selon la situation souhaitée et peut créer le nom de son équipe.

On gagne quand le joueur arrive à amener toutes les grenouilles à droite et tous les crapauds à gauche du plateau.

A la fin de la partie la fenêtre VICTOIRE s'affiche.

- Pour une partie de Joueur Vs Ordinateur\n\n"

Le nombre de joueur pour ce mode de jeu est limité à 1, le joueur joue à tour de rôle avec l'ordinateur.

La possibilité de créer un nom de d'équipe est toujours possible.

Le déroulement de la partie de Joueur Vs Ordinateur se déroule dans le même sens que le mode Joueur Vs Joueur, c'est-à-dire que le joueur commence, puis c'est au tour de l'ordinateur et ainsi de suite.

Enfin, pour gagner la partie, il suffit de bloquer les pions de l'ordinateur sur chaque ligne du plateau.

Bien sûr, à la fin de la partie la fenêtre VICTOIRE s'affichera.

# Bilan

Ce projet nous à permit d’avoir un aperçu la difficulté d’un gros projet avec un groupe de 4 personne. Nous avons pu découvrir la difficulté de la communication dans le groupe qui pouvaient amener à des problèmes sur les codes, mais aussi la difficulté de coder ensemble sur une même application.

Nous sommes plutôt fières de notre application même s’il y a des choses qui pourraient être améliorés ou à ajouter. Nous avons réussi à garder une bonne entente dans tout le groupe.