Projet **J ²G** : Développement en **J**ava d'un **J**eu **G**énérique

Objectif :

Le but de ce projet est de développer une plateforme de jeu en Java fonctionnant sur un plateau d'échecs (8x8 cases, modifiable ensuite par l'utilisateur). Le programme doit permettre de configurer des types de pièces prédéfinies, des pièces définies par l'utilisateur, une taille de plateau, un placement initial et des règles de victoire.

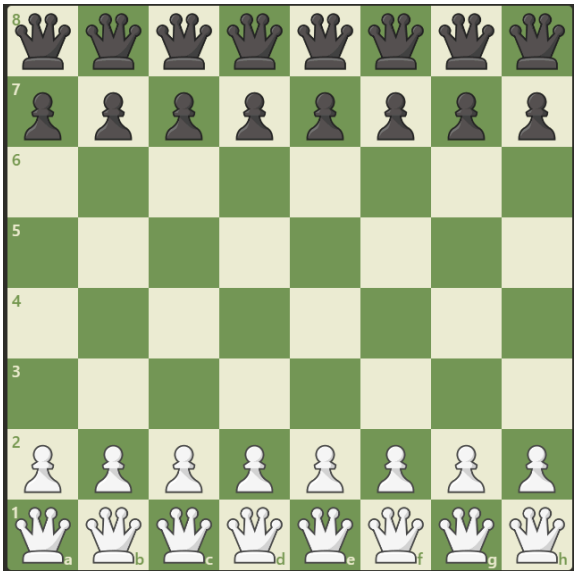
Le projet inclura une interface graphique pour l'utilisateur et une interface dédiée pour les clients qui se connectent à la plateforme. Deux clients maximums pourront se connecter pour participer à une partie. Le jeu pourra aussi se jouer en Hot Seat en local directement via l'interface.

Le programme principal devra inclure une interface graphique pour l'utilisateur permettant de configurer et de visualiser la partie. Des clients pourront se connecter à la plateforme via un protocole de message standardisé, en respectant une interface définie (cf Annexe) pour assurer une communication homogène.

Plateau du jeu (Board)

L'utilisateur choisira le type de pièce et leur placement :

- Roi, Reine, Fou, Cavalier, Tour, Pion.
- Le déplacement devra pouvoir respecter les règles du jeu d'échecs.
- Voici 2 plateaux possibles prédéfinis obligatoirement (il devra être possible d'en rajouter)

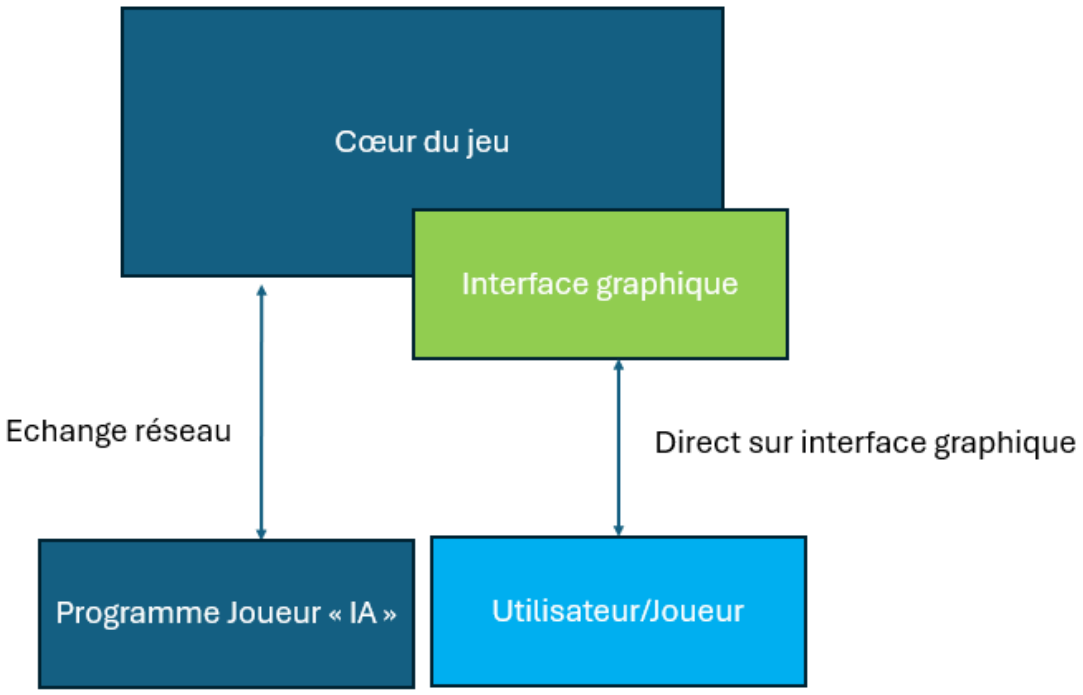


Les règles de victoire pourront être configurées avec **des** conditions comme :

- **Prendre le roi** de l'adversaire (s'il y en a au moins 1)
- **Capturer la première pièce de type X** de l'adversaire. (X pouvant être un type de piéce précis ou "quelque soit" la pièce)

- **Atteindre la dernière ligne** avec une pièce.
- **Aligner x pièces** avec une séquence et orientation. (Par exemple une Reine, Pion , Cavalier alignés en diagonale South-East)

Voici un schéma d'architecture de votre programme :



Il y aura donc 2 programmes à développer : le cœur du jeu, et les programmes clients qui s'y connectent pour jouer une partie. De plus, une interface graphique est demandée pour l'utilisateur.

Le programme client doit pouvoir recevoir les informations de la partie et envoyer les coups en qualité de joueur.

Il est demandé de réaliser une "IA naïve" pour pouvoir tester de bout en bout le système. Les IA doivent gérer le temps alloué pour renvoyer une réponse, qui sera un paramètre du cœur du jeu. Si l'IA n'envoie pas le coup dans le délai, un coup aléatoire sera renvoyé.

Fonctionnement

L'utilisateur défini dans un premier temps les fonctionnements d'une partie :

-Le plateau

-Règles utilisées

-Temps donnés par joueur pour proposer un coup

-Nombre de joueur "locaux" ou devant se connecter ("IA")

Ensuite si il y a au moins un joueur se connectant "IA" :

-Attente de la connection de tous les joueurs

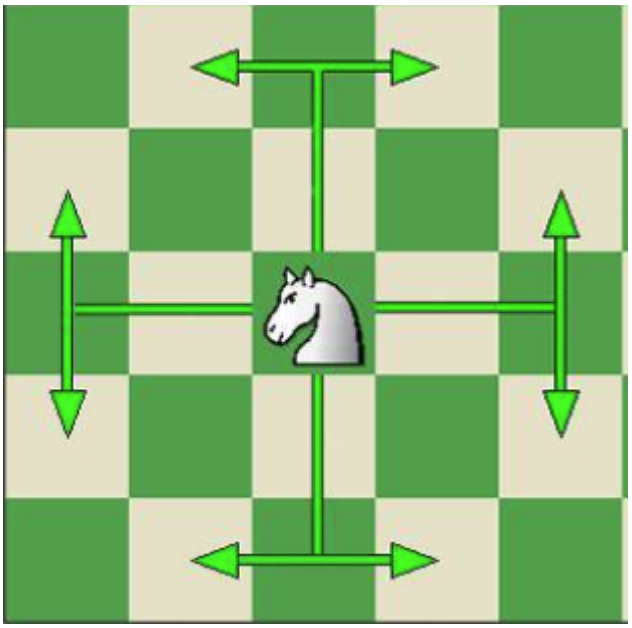
-Après synchronisation lancement de la partie

Nouvelles pièces :

Il devra être possible de configurer des nouvelles pièces avec un nouveau déplacement.

Une pièce sera définie par différentes caractéristiques (cf Annexe). Il devra être possible de les sauvegarder. Les couleurs sont à définir, voici des pièces connues des échecs représentées dans l'interface si on voulait les créer :

Par exemple le cavalier sera défini en tant que déplacement et prise :



Les déplacements "non bloquants" étant en vert.

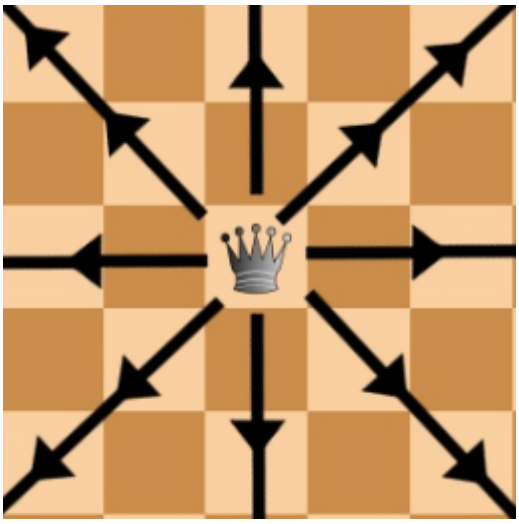
Un pion sera de ce type :

Flèche bleu : (valeur 1) : déplacement de base

Fléche noir : (valeur 2) premier déplacement possible

Flèches oranges : prise

Reine :



Flèches noires : (valeur 1,+infini) déplacement et prises possibles bloquant

Il devra donc être possible de définir toutes les possibilités des pièces. On limitera les déplacements non rectilignes à ceux du cavalier.

Tournois :

Il devra être possible de mettre x IA dans un répertoire de faire jouer les IA toutes ensembles automatiquement sous le format suivant :

-Ronde suisse

-Puis élimination direct des 8 premiers de la ronde suisse en ¼ finale, ½ finale , finale

Un récapitulatif clair et efficace des scores devra être consultable

Tâches à Réaliser :

1. **Conception de la plateforme de jeu :**
    - Créer un système permettant de configurer un plateau d'échecs avec des pièces prédéfinies (ou personnalisées) et des positions de départ personnalisées.
    - Utiliser des classes abstraites et des interfaces pour la gestion des pièces et des règles de déplacement.
    - Permettre de créer des pions spécifiques.
    - Permettre de définir un plateau de jeu de taille variable.
    - Gestion des tournois.
2. **Configuration des règles de victoire :**
    - Permettre la configuration des règles de victoire
    - Possibilité d' ajouter des règles de victoires facilement
3. **Interface graphique :**
    - Développer une interface graphique pour l'utilisateur principal, permettant de visualiser le plateau et de configurer les paramètres du jeu.La technologie graphique (JavaFX, Swing, etc.) est à votre choix.
    - Cette interface doit être intuitive et réactive.
    - L'interface doit complètement être séparée au niveau code du cœur du système.
4. **Définition d'une interface pour les clients :**
    - Définir une interface standardisée pour les clients qui se connecteront à la plateforme. Cette interface doit permettre une communication cohérente entre les clients et la plateforme, incluant des méthodes pour envoyer des mouvements et recevoir les états de jeu (cf Annexe fin du document).
    - Le protocole de communication pourrait être basé sur des messages en JSON, XML ou tout autre format structuré.
5. **Gestion des exceptions personnalisée :**
    - Mettre en place une gestion des exceptions personnalisée pour capturer les erreurs spécifiques (par exemple, mouvements invalides, erreurs de connexion, erreurs liées aux règles de victoire).

    - Créer des classes d'exceptions dédiées pour une meilleure gestion des erreurs.

6. **Système de log performant :**
    - Intégrer un système de log performant (comme Log4j, SLF4J ou votre propre système) pour suivre les événements importants (mouvements, erreurs, connexions des clients).
    - Les logs doivent être configurables avec différents niveaux (INFO, DEBUG, ERROR) et permettre un suivi détaillé.
7. **Protocole de communication et gestion des clients :**
    - Implémenter un protocole de communication entre la plateforme et les clients, en respectant l'interface définie en annexe.
    - Limiter le nombre de clients connectés à deux maximums et gérer les connexions en parallèle.
8. **Utilisation de l'héritage et des classes abstraites :**
    - Exploiter l'héritage, les classes abstraites et les interfaces pour garantir la modularité et l'extensibilité du programme.
    - Il doit être aisé d'ajouter une nouvelle pièce.
9. **Documentation et Tests :**
    - Documenter l'architecture, les classes, et les règles de communication des clients.
    - Inclure des tests unitaires et fonctionnels pour valider la bonne gestion des règles, des exceptions, et des interactions entre la plateforme et les clients.

Livrables :

- Code source complet incluant la plateforme, les clients et l'interface graphique.
- Documentation détaillant l'architecture, l'interface des clients et le protocole de communication.
- Documentation utilisateur pour un non informaticien souhaitant utiliser la plateforme.
- Fichiers de configuration des règles de victoire et des logs.
- Rapport final sur les choix techniques et les difficultés rencontrées.

Critères d'Évaluation :

- Qualité de la modularité et extensibilité de l'architecture logicielle. (Facilité de rajouter une pièce ou une condition de victoire par exemple)
- Conformité de l'interface des clients et de leur interaction avec la plateforme.
- Robustesse de la gestion des exceptions et des performances du système de log.

- Qualité de l'interface graphique et de son ergonomie.

Étudiants :

- Ce projet est prévu pour des groupes d'étudiants de 3 étudiants.
- 18/01 : Livraison finale, pas de modification possible
- 23/01 : Présentation devant l'enseignant, il sera obligatoire de pouvoir expliquer l'ensemble du code
- Il est possible que cette plate-forme devienne le support de travail pour l'IA des futurs étudiants !

Annexes.

Système échange de message minimal à implémenter :

Serveur -> Joueur

BEGIN[x] : lancement d'une partie avec dénomination du numéro du joueur.

BOARD [Board] : position du board

ERROR [Msg] : message d'erreur (coup impossible)

RULES[Rules] : envoi les règles de la partie

TIME[x] : temps x millisecondes pour jouer un coup

MOVEIMMEDIATE[x] : sous x millisecondes sans réponse du joueur, un coup aléatoire sera donnée

MOVE[xdépart,ydépart;xarrivé,yarrivé;A] : envoi du coup qui a été joué par le joueur A

CONNECTION[connection] : envoi les informations de connection

CHECKCONNECTION[msg] : vérifie la connection

WIN[x] : indique le joueur gagnant

DRAW[] : indique une partie nulle (plus de coup disponible, ou plus de possibilité de gagner)

Joueur -> Serveur

READY[] : signifie que le joueur est prêt à jouer

MOVE[xdépart,ydépart;xarrivé,yarrivé] : envoi du coup à jouer

BOARD?[] : demande du board

RULES?[] : demande d'envoi des règles

CONNECTION[connection] : envoi les informations de connection

CHECKCONNECTION[msg] : vérifie la connection

<u>Caractéristique d'une nouvelle pièce :</u>

Nom

Image Joueur1/Joueur2

Déplacement (direction + plage de cases + non bloquant par autres pièces)

Déplacement premier coup (direction + plage de cases + non bloquant par autres pièces)

Prise (direction + plage de cases)