**COURS: Java-OOP**

Rapport du projet

Sujet : Jeu d’échecs



Faculté d’Informatique

Université des Sciences



Université Claude Bernard Lyon 1

Note finale

– par l’enseignant

# Auto-évaluation

Étudiant 1 : NGUYEN Van Thanh Tu - p1419808

Étudiant 2 : HUYNH Cong Lap -

Contribution des membres dans le projet:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **NoE** | **Prénom et nom** | **Email** | **Contribution** |
| 1 | P1419808 | NGUYEN Van Thanh Tu | nguyenvanthanhtu@gmail.com | 50% |
| 2 | P1419778 | HUYNH Cong Lap | Laphuynh915@gmail.com | 50% |
|  |  |  |  | **100%** |

Liste des taches réalisées par chacun des membres évalué par le professeur.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Réalisateur** | **Tache** | **Accomplishment** | **Commentaire** |
| **1** | **NGUYEN** | **Les algorithmes pour la déplacement de chaque pièce** |  |  |
| **2** | **HUYNH** | **Vue (Coup, Plateau... )** |  |  |
| **3** | **HUYNH** | **Interface** |  |  |
| **4** | **HUYNH** | **Menu** |  |  |
| **5** | **HUYNH** | **Traitement de l'interaction** |  |  |
| **6** | **HUYNH** | **Logic game (boucle, fin...)** |  |  |
| **7** | **NGUYEN** | **Ajouter observer pattern** |  |  |
| **8** | **NGUYEN** | **Organiser les modèles et ses relations** |  |  |

# Évaluation des vues principales

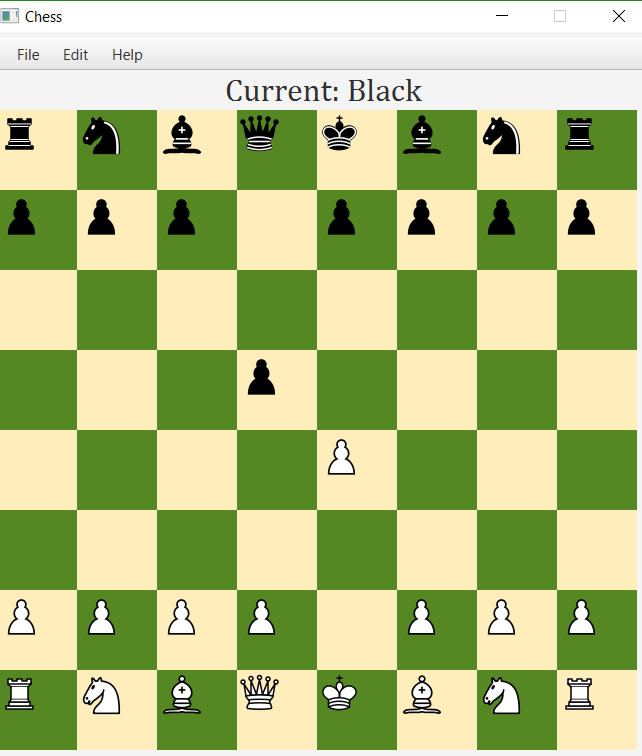
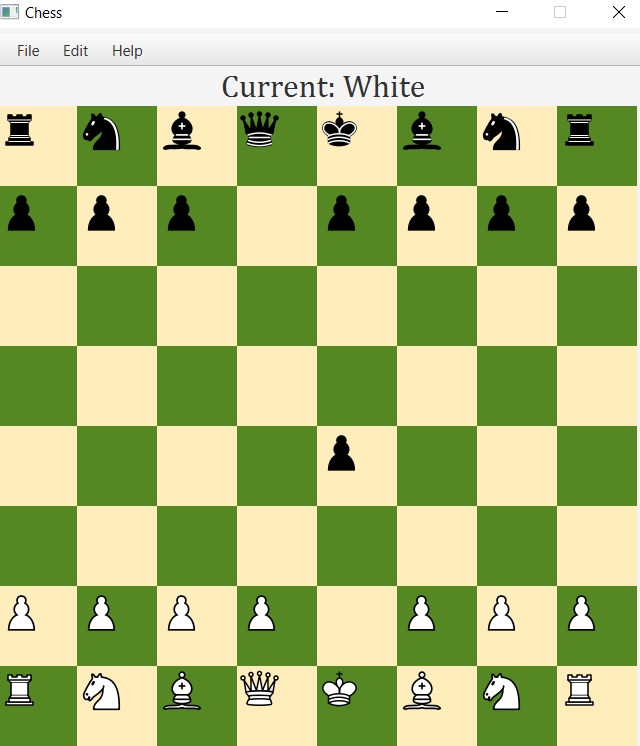
Liste des fonctions réalisées par chacun des membres.

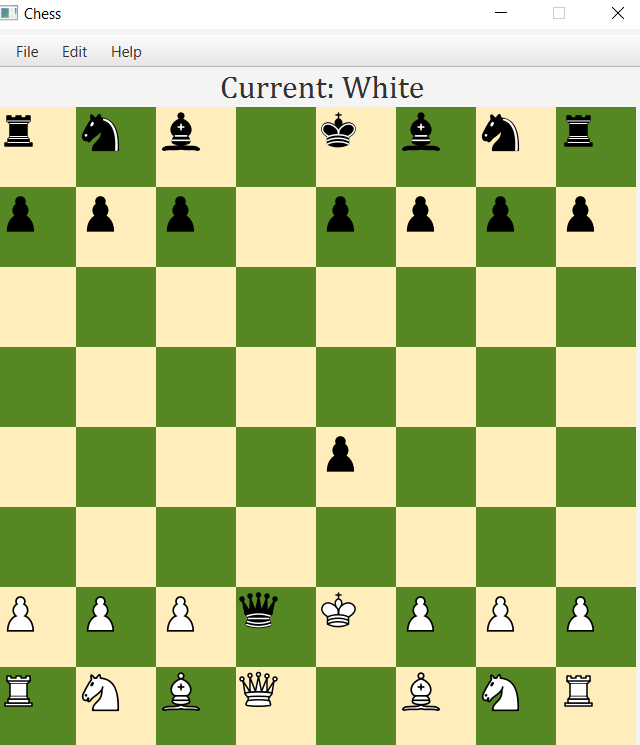
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Réalisateur** | **Vue** | **Note (Auto-Evaluation)** | **Commentaire** |
| **1** | **HUYNH** | Piece (Stackpane) | 10/10 | Cette vue affiche l'image des pieces. Il contient PieceType |
| **2** | **HUYNH** | PieceType (Enum) | 10/10 | Il contient 12 types (6 noirs, 6 blancs) |
| **3** | **HUYNH** | Tile(Careaux) | 10/10 | Il y a 8x8 Tiles, il cree un plateau, un tile peux contenir un Piece |
| **4** | **HUYNH** | MoveResult | 10/10 | Il contient MoveType, retour resultat si on peut déplacer des pieces. |
| **5** | **HUYNH** | MoveType(Enum) | 10/10 | Il y a 3 état.  None  Normal  Kill |
| **6** | **HUYNH** | Stage(Interface) | 10/10 | Menu, Dialog, Text Label, Message |
| **7** | **HUYNH** | Controller l'interaction de la vue au model | 10/10 | Envoyer la coordination au model |
| **8** | **HUYNH** | Controller l'événement du joueur | 10/10 | Traitement des actions du joueur (clicks, drags..) |
| **9** | **NGUYEN** | Piece | 10/10 | Une classe abstract avec la fonctionnalité de calculer la déplacement des Pièces |
| **10** | **NGUYEN** | Pawn, King, Knight | 10/10 | 3 classes spéciales avec 3 différences algorithmes de la fonction can\_Move() |
| **11** | **NGUYEN** | Grid | 10/10 | Variable contient l’information d’un carreau (Piece, Player) |
| **12** | **NGUYEN** | BoardChess | 10/10 | Important classe qui contient la liste de carreaux (Grid[8][8]) et toutes les pièces. |
| **13** | **NGUYEN** | ChessGame | 10/10 | La classe principale de changer le tour de jouer, implémenter « Observable » et faire la déplacement des pièces. |

# - Description Technique

## - Interface graphique

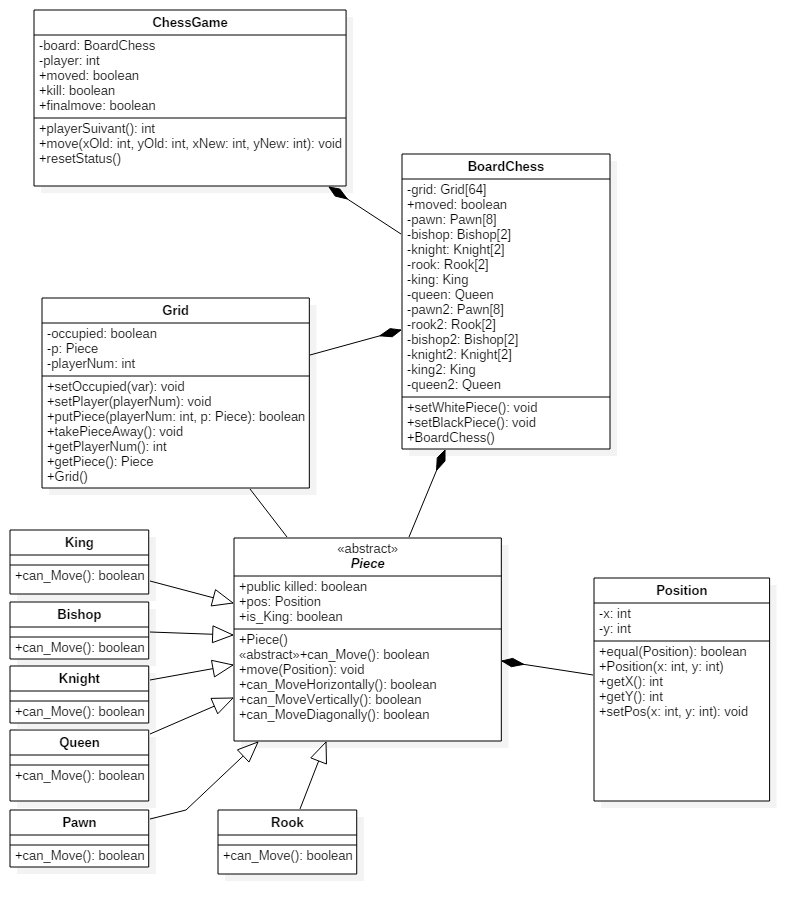
Capturez les écrans de l’application

## - Organisation des modules

**Diagramme de classe pour le modèle**



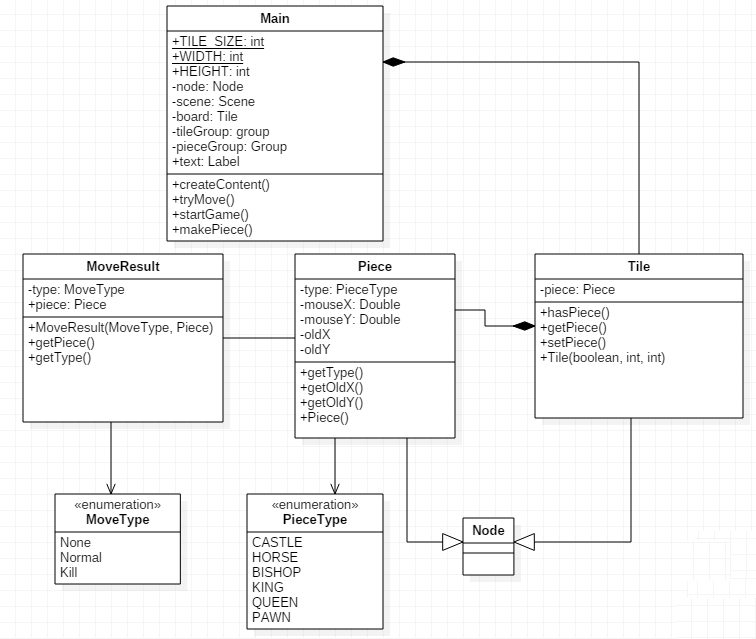
La classe **ChessGame** est le responsable de changer le tour de jouer, déplacer les pièces si possible.

La classe **BoardChess** est pour préparer le plateau et elle contienne toutes les pièces et la liste de careaux (grid).

La classe **Pièce** est une classe abstract qui contient 3 principales fonctionnalités utilisé par la plupart des pièces (can\_MoveHorizontally, can\_MoveVertically, can\_MoveDiagonally).

Il y a 8 types de Pièce : **Pawn (Pion), Knight (Cavalier), Rook (Tour), Bishop (Fou), Queen (Dame), King (Roi).**

**Diagramme de classe pour la vue**



Class Main contient un stage par la scène (menu, text label, root). Root est la Node crée par fonction createContent(). Il contient 8x8 Tiles dans la variable tileGroup() et 16x16 pièces dans la variable pieceGroup. Tout est affiché dans l'écran par stage.show();

**Diagramme de séquence**

Quand l’utilisateur prend une pièce et le met sur une autre location du plateau.

Chessgame

Piece(Vue)

Controller

movePiece(xNew,yNew)

prendre le piece

activer OnMousePressed ()

relocate()

mettre le piece au (x,y)

occupier la location nouvelle

move(xOld,yOld,xNew,yNew)

alt

NONE

abortMove()

NORMAL

KILL

remove(getPiece(xOld,yOld))

movePiece(xNew,yNew)

## - Resume des techniques utilisées

Nous avons utilisé la modèle MVC avec :

* Modèle : **ChessGame, BoardChess et Piece** (8 types)
* Vue : Vide, crée en utilisant JavaFX
* Contrôller : EventListener

Nous avons implémenté le pattern **Observer :**

* On implémente la classe **ChessGame** l’interface **Observable** et ajouter Observer avec la fonction **addObserver()**. On va la mettre dans la fonction **startGame()**.
* Avec ça, chaque fois ChessGame a fini avec la calcule et notifier l’observer (la vue) avec la fonction **setChanged()** et **notifyObservers()**, on va mettre à jour la location de la pièce utilisateur choisi.

## – Références

* [*https://www.tutorialspoint.com/mvc\_framework/mvc\_framework\_introduction.htm*](https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm)
* [*https://docs.oracle.com/javase/7/docs/api/java/util/Observable.html*](https://docs.oracle.com/javase/7/docs/api/java/util/Observable.html)
* [*http://www.javaworld.com/article/2077258/learn-java/observer-and-observable.html*](http://www.javaworld.com/article/2077258/learn-java/observer-and-observable.html)