ORTANCA Baris
L2-S4-TD1-TP2
n étudiant :22003354

Rapport projet POO 2

Jeu d'échec

Sommaire :

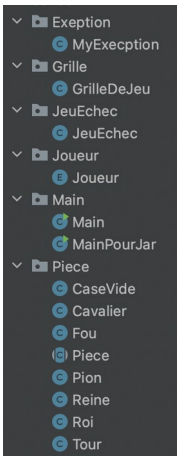
Idée principal :

Pour commencer , j'ai pense a faire un liste allant de 0 jusqu'a 64 mais je me suis dit que ça serait plus compliquer pour les déplacement.

Je suis partie sur l'idée de faire plusieurs classe pour les pièces (tels que les pions, tours , rois, etc...), une class jeu (tels que pour les fonctions qui permet de lancer une partie, initialiser la partie , etc …), une class grille (tels que pour les fonction d'ajout de pièce dans la grille, affichage de la grille, déplacer une pièce, vérifier si le roi est encore en vie, etc …).

Idée actuelle :

Je code actuellement sur intellij Idea, ce qui aide pour lancer le programme ou autre. Actuellement j'ai plusieurs class (tels que sur l'image a droite).
J'ai opter pour une grille tels que `public Piece[][] grille ;` ou Pièce est une class abstraite contenant les information principal pour les autres pièces qui sont extends. J'ai choisie de coder le code en Swing mais j'ai vite changer d'avis pour continuer en javaFX (je vais en revenir dessus dans la partie Problème rencontrer).



Code général explication des fonctions :

- La fonction afficher grille comme dit le nom permet d'afficher la grille dans une fenêtre, en prenant les noms des pièces présentes et en ajustant les positions des images.

```
public void afficherGrille() {

    root.getChildren().clear();
    Image board = new Image(IMAGE_PATH);
    ImageView boardView = new ImageView(board);
    root.getChildren().addAll(boardView);
    for (int x = 0; x < LIGNE; x++) {
        for (int y = 0; y < COLONNE; y++) {
            Piece piece = grille[x][y];
            if (!piece.caseVide) {
                String imagePath = String.format("file:src/main/resources/echec/image/%s_%s.png",
                        piece.nomPiece, piece.joueur == Joueur.noir ? "n" : "b");
                ImageView pieceView = new ImageView(new Image(imagePath,  v: 100,  v1: 100,  b: true,  b1: true));
                pieceView.setLayoutX(x * 100);
                pieceView.setLayoutY(y * 100);
                grillePane.getChildren().add(pieceView);
            }
        }
    }
    root.getChildren().add(grillePane);
    stage.show();
}
```

- La fonction si contre est la fonction qui permet de prendre dans la class des pièces directement la liste de mouvement possible par la pièce depuis la ou le joueur a clique.

```
public int[][] getDeplacementPossiblePourUnePiece(Piece piece, int x, int y) {
    int[][] resultat = new int[LIGNE * COLONNE][2];
    int[][] deplacementPiece = piece.getDeplacement();
    int n = 0;

    for (int[] ints : deplacementPiece) {
        int xDestination = x + ints[0];
        int yDestination = y + ints[1];
        if (yDestination >= 0 && yDestination < COLONNE
                && xDestination >= 0 && xDestination < LIGNE) {
            if (!grille[yDestination][xDestination].caseVide) {
                resultat[n] = ints;
                n++;
            }
            if ( grille[yDestination][xDestination].caseVide && grille[yDestination][xDestination] != null) {
                resultat[n] = ints;
                n++;
            }
        }
    }
    return resultat;
}
```

- La fonction déplacement un peut long, je vous conseille d'allez voir ligne 130 de la class GrilleDeJeu. La fonction consiste a colorier en cercle jaune et a regarder si la pièce choisi et l'endroit destination est une case vide ou non.
Si c'est une case vide alors on échange juste de place, si la case et une pièce adverse alors on peut la manger et pour ceci on on remplace la case de départ par une pièce nommer caseVide.

- La fonction roiToujoursVivant renvoie un string qui permet de dire si il y a un gagnant ou non et qui vas servir pour la popup en cas de victoire.

```
public String roiToujoursVivant() {
    boolean roiBlanc = false, roiNoir = false;
    for (int x = 0; x < LIGNE; x++) {
        for (int y = 0; y < COLONNE; y++) {
            if (Objects.equals(grille[x][y].nomPiece,  b: "Roi")) {
                if (grille[x][y].joueur == Joueur.noir)
                    roiNoir = true;
                else
                    roiBlanc = true;
            }
        }
    }
    if (roiNoir && roiBlanc)
        return "";
    if (roiNoir)
        return "Roi noir gagnant";
    else
        return "Roi blanc gagnant";
}
```

- La fonction popup permet de crée une fenêtre lorsque un des deux roi est mort, la fenêtre contient le nom du vainqueur et aussi avec un bouton qui permet de recommencer la partie.

```
void popupVainqueur() {

    if (!Objects.equals(jeu.roiToujoursVivant(),  b: "")) {
        Stage vainqueur = new Stage();
        Text message = new Text();
        message.setText(jeu.roiToujoursVivant());
        message.setLayoutX(100);
        message.setLayoutY(400);
        Image backGround = new Image( s: "file:src/main/resources/echec/image/gagner.jpeg",
                v: 500,  v1: 500,  b: true,  b1: true);
        ImageView backGroundView = new ImageView(backGround);
        Group group = new Group();
        group.getChildren().add(backGroundView);
        group.getChildren().add(message);
        Button recommencerPartie = new Button( s: "Recommancer !!");
        recommencerPartie.setLayoutX(250);
        recommencerPartie.setLayoutY(250);
        recommencerPartie.setOnMouseClicked(e -> {
            joueur1 = true;
            joueur2 = true;
            jeu = new GrilleDeJeu(stage);
            initialisationJeu();
            lancerJeu();
            vainqueur.close();
        });
        group.getChildren().add(recommencerPartie);

        Scene vainqueurScene = new Scene(group,  v: 500,  v1: 500, Color.LIGHTGRAY);
        vainqueur.setScene(vainqueurScene);
        vainqueur.setTitle("RESULTAT !!!!");
        vainqueur.show();
    }
}
```

- Les deux fonctions qui suivent permettent de mettre en place le système de tour par tour, c’est à dire une fois que le blanc a jouer alors c’est au suivant ce qui empêche de jouer 2 fois.

```
void Joueur2DeJouer() {
    joueur2 = true;
    popupVainqueur();
    jeu.scene.setOnMouseClicked(mouseEvent -> {
        int mouseX = (int) mouseEvent.getSceneX() / 100;
        int mouseY = (int) mouseEvent.getSceneY() / 100;

        if (jeu.grille[mouseX][mouseY].joueur == Joueur.noir) {
            Piece selectioner = jeu.grille[mouseX][mouseY];
            System.out.println("le joueur 2 a clicker en x = " + mouseX + ", y = " + mouseX);
            jeu.deplacement(mouseX, mouseY, selectioner);
            joueur1 = true;
            joueur2 = false;
            jeu.afficherGrille();
            Jouer();
        } else
            Joueur2DeJouer();
    });

}
```

```
void Joueur1DeJouer() {
    joueur1 = true;
    popupVainqueur();
    jeu.scene.setOnMouseClicked(mouseEvent -> {
        int mouseX = (int) mouseEvent.getSceneX() / 100;
        int mouseY = (int) mouseEvent.getSceneY() / 100;
        if (jeu.grille[mouseX][mouseY].joueur == Joueur.blanc) {
            Piece selectioner = jeu.grille[mouseX][mouseY];
            System.out.println("le joueur 1 a clicker en x = " + mouseX + ", y = " + mouseY);
            jeu.deplacement(mouseX, mouseY, selectioner);
            joueur2 = true;
            joueur1 = false;
            jeu.afficherGrille();
            Jouer();
        } else
            Joueur1DeJouer();
    });

}
```

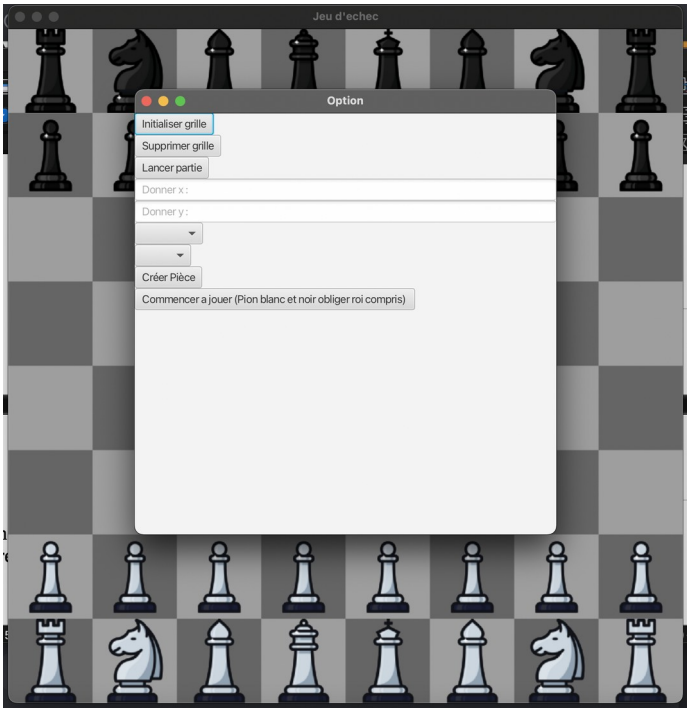
- La fonctions si contre (Jouer) permet de faire juste une vérification préliminaire afin de de passer au joueur prochaine.

```
void Jouer() {
    jeu.afficherGrille();
    if (!(Objects.equals(jeu.roiToujoursVivant(),  b: ""))) {

        joueur2 = false;
        joueur1 = false;
    }

    if (joueur1)
        Joueur1DeJouer();
    if (joueur2)
        Joueur2DeJouer();
}
```

- Ligne 171 de la class JeuEchec la fonction fenetreOption() permet de lancer une deuxième fenêtre contenant different option tels que comme sur l'image si dessous :



ou on a different buttons, tels que pour :

. pour initialiser la grille
. pour supprimer le contenue de la grille
. pour lancer la partie

. on 2 emplacement ce qui vas nous aider a insérer une pièce, pour ca nous faut remplir les deux champs avec des entiers entre 0 et 7 pour x et y , puis 2 choix a faire une pour la pièce et l'autre la couleur puis faut appuyer sur crée pièce.

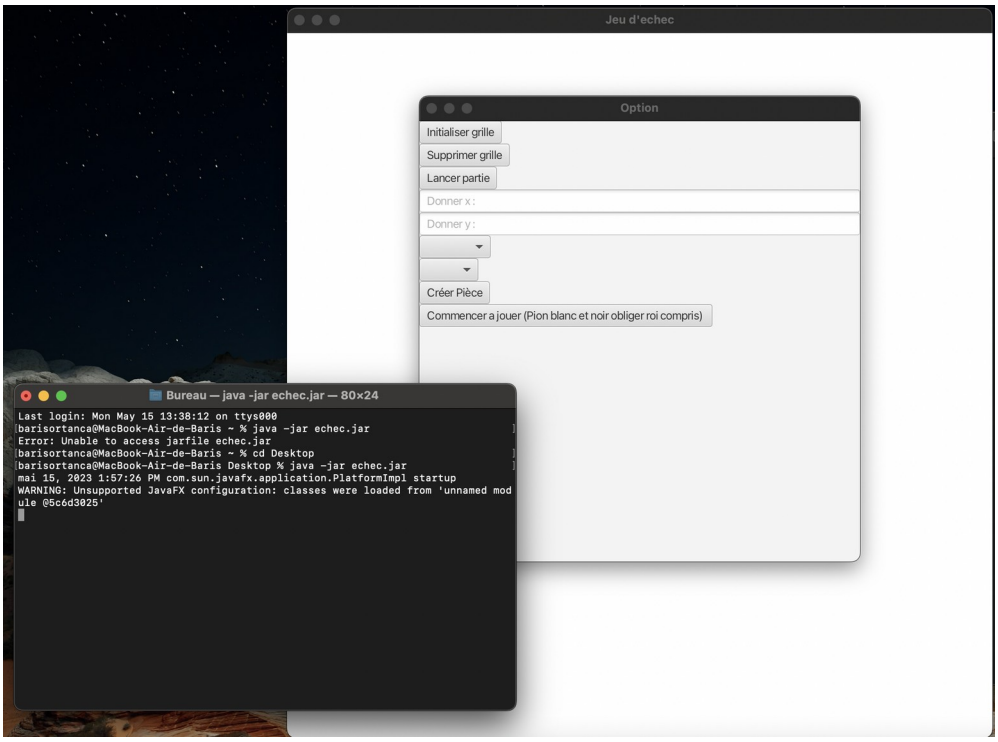
. le dernier bouton sert a commencer a jouer avec les pièces ajouter, pour qu'il fonctionne faut qu'il y ai un roi blanc et un roi noir minimum.

Problème rencontrer :

Voilà pour les problèmes rencontrer, déjà a propos de swing lorsque je fessais l'affichage je ne pouvais pas mettre l'un sur l'autre les images du coup je suis partie en javaFX ou la gestion (pour moi) est plus simple. En javaFX je n'ai pas eu énormément de problème juste les class de la librairies javaFX que je ne connaissais pas très bien et pour résoudre mon manque de connaissance j'ai regarder des videos sur Youtube tels que pour du textArea ou même encore ChoiceBox , etc … Mais j'arriver toujours à avoir un rendu qui me plaisais bien. Juste un problème que j'ai actuellement c'est lors des déplacement, a vrai dire jai 3 soucis avec les déplacements :

- Le premier est que lorsque j'appuie sur une case contenant une pièce de mon tour (si joueur 1 alors blanc sinon noir pour éviter de jouer un pions qui nous appartient pas) je n'ai pas réussir a faire le système de collisions tels que pour éviter de bouger une pièce si il y a une pièce sur son chemin.
- Le deuxième est que lorsque l'on clique sur une case pour la déplace faut cliquer sur le cercle et non sur l'image de la pièce sur la quelle on veut manger
- La troisième est que une pièce toucher est obligatoirement pièce jouer.

J’ai une autre erreur lors de l’exécution du jar ça m’affiche ça dans le terminal :



UML :