École Centrale Lyon

UE APPRO
Rapport

Rapport BE 1 - Jeu d'araignée

Élève :
Mohammed Achraf El Khamlichi

Enseignant :
Stéphane DERRODE

4 octobre 2022

Table des matières

1 Introduction

Le jeu de l'araignée se joue à deux joueurs qui interviennent alternativement. Chaque joueur dispose de trois pions qu'il peut poser puis déplacer sur une grille $3x3$. Le but du jeu pour chaque joueur est d'aligner ses 3 pions avant que l'adversaire ne le fasse. Le jeu est donc divisé en deux phases : une phase de placement et une de déplacement.

Le but du BE est donc de programmer ce jeu en appliquant les connaissances vues dans les deux premiers chapitres. Ainsi, je présenterais dans ce rapport les diagramme de classes de mon projet, ainsi que des explications et commentaires sur le code et les différentes étapes du jeu.

2 Diagramme de classes

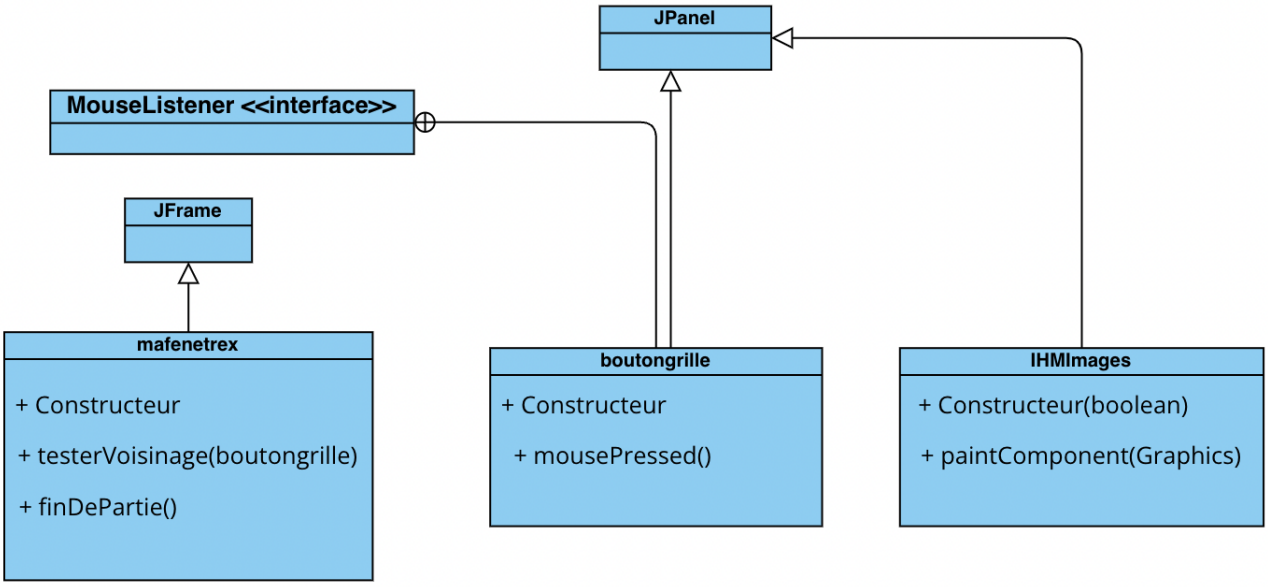

FIGURE 1 – Diagramme de classes

Le diagramme des classes ci-dessus ne présente pas les attributs, ni présente complètement les méthodes : On ne précise pas les arguments des méthodes **Constructeur()** ni le nom des arguments des autres méthodes. Il a pour simple but de présenter les relations entre les différents classes : On retrouve une implémentation d'une interface écouteur et des relations d'héritage.

3 Explication sur le code

La classe **mafenetrex** sert à construire et personnaliser la fenêtre du jeu. Elle comporte deux Panels : Le premier contient un texte informant l'utilisateur si le jeu est en cours ou la partie est terminée et affiche dans ce cas le joueur gagnant.

Cette classe est caractérisée par deux méthodes :

- **testerVoisinage()** permet de valider un déplacement si et seulement si on vise une case voisine à celle de départ.
- **finDePartie()** permet de vérifier si un joueur a pu alligné ses trois pions ou non.

La classe **boutongrille** permet de créer une case où les pion peuvent être placé ou déplacés. Elle implémente un **MouseListener** qui permet lorsqu'on clique sur la case de générer un événement :

- Déposer un pion dans une case vide si on est dans la phase de placement (Nombre de pions total inférieur à 6.
- Capturer un pion pour pouvoir le déplacer.
- Déplacer un pion s'il y en a déjà un capturé.

La classe **IHMImages** sert à dessiner les pions blancs et noirs dans les cases lors du phases de placement et de déplacement.

Un cas particulier

Parfois, il se peut que l'utilisateur sélectionne sans faire attention un pion qui ne peut se déplacer (Toutes les cases voisines sont pleines), dans ce cas, on permet à l'utilisateur de remettre son pion dans sa case de départ. Deux situations se présentent :

- On considère que le mouvement est validé comme les autres mouvement et on passe au tour suivant (si c'est le blanc qui a fait ce mouvement alors c'est au noir de jouer). C'est le cas présenté dans notre programme.
- On ne prend pas compte du mouvement (si c'est le blanc qui a fait ce mouvement alors c'est encore son tour de jouer).

Pour ce deuxième cas, il suffit de changer la valeur de **return** de la fonction **testerVoisinage()** : Au lieu d'un boolean, on retourne un **int** qui vaut :

- 1 si l'utilisateur a sélectionné une case voisine autre que sa case départ pour son déplacement.
- 2 si l'utilisateur a sélectionné sa case de départ pour son déplacement.
- 0 Sinon. (Ce cas est réalisé pour retourner une valeur int qui engendra une son satisfaction de la condition **(mywindow.testerVoisinage(this) == 1) || (mywindow.testerVoisinage(this) == 2)**

Le code correspondant à ce deuxième cas est présenté ci-dessous : il est également disponible en commentaire dans le script. Pour tester ce cas, Il faut remettre la condition **(mywindow.testerVoisinage(this)** et ses instruction en commentaire.

```
if ((mywindow.testerVoisinageV2(this) == 1) || (mywindow.testerVoisinageV2(this) == 2)){
    PieceEnCase = mywindow.pieceCapturee ;
    monPion = mywindow.PionCapture ;
    add( monPion );
    NbrPieceCase = 1 ;
    mywindow.pieceCapturee = "XX";
    mywindow.PionCapture = null ;
    mywindow.finDePartie();
    repaint();
    revalidate();
    if (mywindow.testerVoisinageV2(this) == 1){
        boolean alpha = mywindow.tourDesBlancs ;
        mywindow.tourDesBlancs = !alpha ;
    }
```

FIGURE 2 – Code du deuxième cas

4 Captures d'écran des différentes étapes du jeu

FIGURE 3 – Avant de commencer

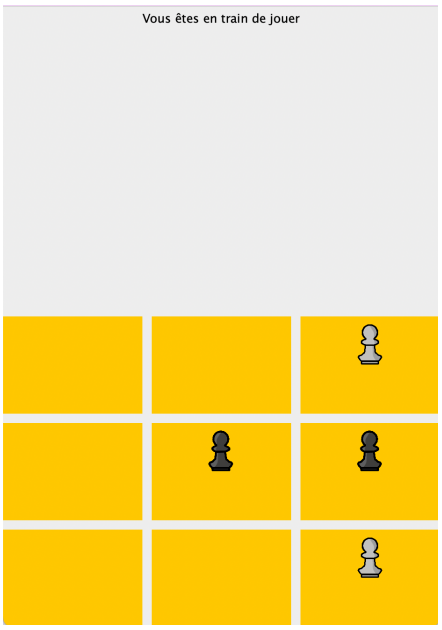

FIGURE 4 – Phase de placement

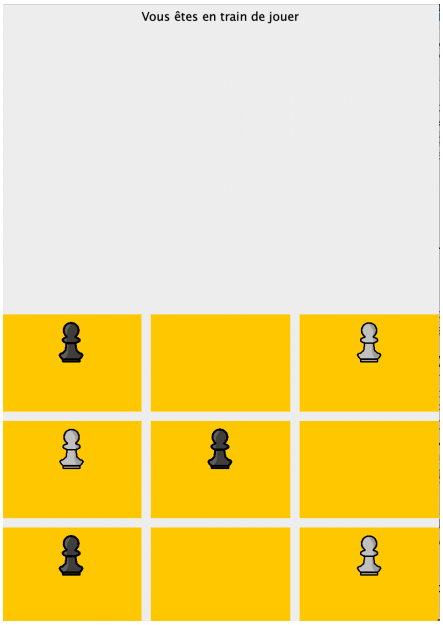

FIGURE 5 – Phase de déplacement

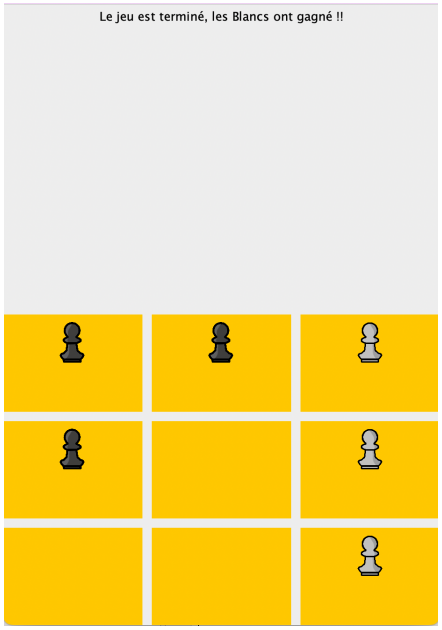

FIGURE 6 – Fin de partie

5 Conclusion

Ce jeu étant simple, il a permis d'appliquer les connaissances vus dans le premier et le deuxième cours de JAVA. On pouvait l'améliorer en proposant un menu qui permet de jouer et rejouer le jeu, intégrer du son et même un tableau de score des différents utilisateurs en utilisant SQL.