Institut national des sciences appliquées de Rouen

INSA de Rouen

Mini-projet COO - GM3

Réalisation d'un jeu de dames en Java

Auteurs :
Thibaut André-Gallis
thibaut.andregallis@insa-rouen.fr
Kévin Gatel
kevin.gatel@insa- rouen.fr

Enseignants :
Mathieu Bourgais
mathieu.bourgais@insa-rouen.fr
Habib Abdulrab
habib.abdulrab@insa-rouen.fr

23 Mai 2021

Table des matières

Introduction

Le mini-projet que nous avons choisi est la réalisation du jeu de dames en Java.

Le jeu de dames est un célèbre jeu de réflexion joué à deux inventé pour la première fois en Egypte antique vers -1500 avant J-C. Pour la version classique, le jeu de dames est composé d'un damier de 10 cases sur 10 cases de deux couleurs alternées et d'une vingtaine de pions par joueur disposés les uns en face des autres. Il existe également des variantes de ce jeu qui utilisent des damiers de 64 cases (8 sur 8) et 144 cases (12 par 12). Le but du jeu est alors de capturer ou d'immobiliser toutes les pièces du joueur adverse.

Le but de ce projet est d'utiliser habilement les concepts de la programmation objet, c'est-à-dire savoir écrire les différents diagrammes, maîtriser l'héritage entre les classes, le polymorphisme, la gestion des interfaces ou des classes arbitraires, etc...

Pour bien répondre au problème, nous allons d'abord regarder le **diagramme de cas** (Use-cases) et le **diagramme de classe** (UML) afin de comprendre l'organisation du code.

Ensuite, nous allons présenter et décrire le rôle de **chacune des classes** qui ont été nécessaires à la réalisation du projet.

Enfin, nous verrons **les différents problèmes** que nous avons pu rencontrer lors de la programmation du jeu.

1. Les diagrammes

1.1 Use-cases

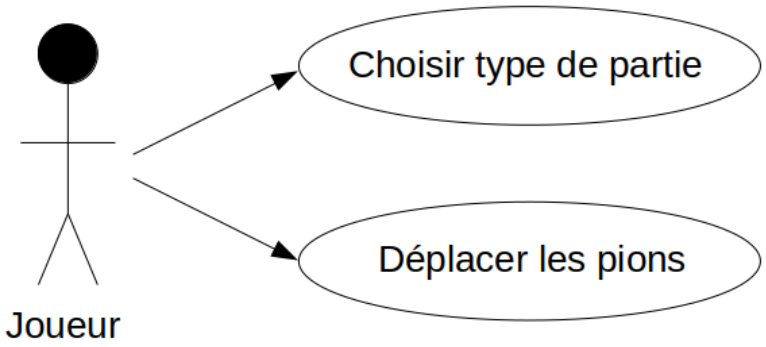

FIGURE 1.1 – Diagramme de cas pour le jeu de dames

Les joueurs commencent par choisir la variante de leur choix pour jouer au jeu de dames. Ils ont le choix entre :

- Une partie rapide : damier 8 sur 8 où chaque joueur possède 16 pièces.
- Une partie classique : damier 10 sur 10 où chaque joueur possède 20 pièces.
- Une partie longue : damier 12 sur 12 où chaque joueur possède 24 pièces.

Ils choisissent ensuite s'ils veulent ou non obliger le saut de pièce (dans la version classique le saut est obligatoire).

La partie se lance et les joueurs déplacent chacun leur tour une pièce de leur couleur jusqu'à ce que la partie se termine.

1.2 UML

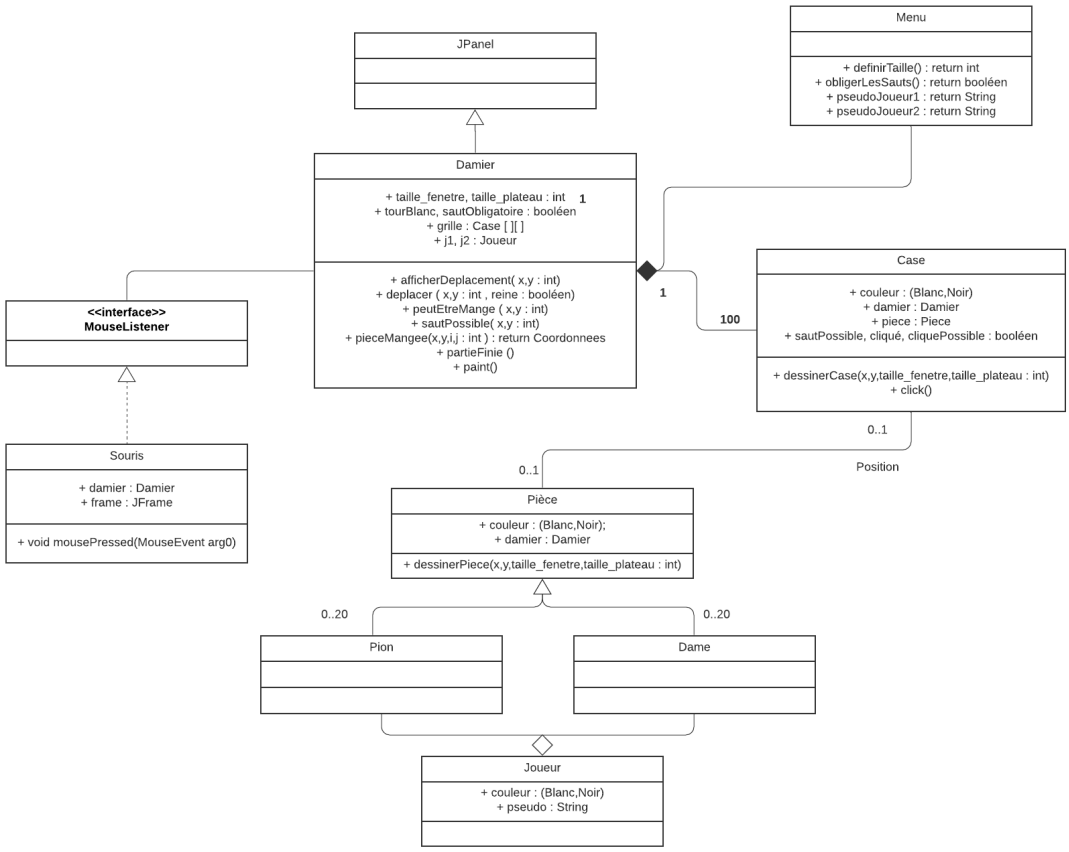

FIGURE 1.2 – Diagramme de classe pour le jeu de dames

La classe *Menu.java* est la première classe utilisée lors du lancement du programme. On définit ici la taille du plateau, on demande aux joueurs s'ils veulent ou non obliger les sauts et on récupère leur pseudo respectif.

La classe *Damier.java* est la classe centrale du projet. Elle permet le déplacement des pièces, l'affichage du damier et la gestion des cases. Grâce à la classe *JPanel*, elle hérite de la méthode prédéfinie *void paint(Graphics g)* qu'on a modifié en utilisant les méthodes *void dessinerPiece()* et *void dessinerCase()* et permet ainsi l'affichage du plateau.

La méthode *void afficherDeplacement(x,y : int)* va colorier les cases disponibles d'une certaine couleur pour le déplacement d'une pièce de coordonnées (x,y). On coloriera en bleu un déplacement sans saut de pièce et en rose pour un déplacement avec saut de pièce.

La méthode *void déplacer(x,y : int)* va permettre ou non le déplacement d'un pion déjà sélectionné sur une case de coordonnées (x,y). Le déplacement sera autorisé si la case a été colorée en bleu ou en rose. Si la case (x,y) est rose, on utilise la méthode *boolean sautPossible* pour savoir s'il peut continuer à jouer, à savoir s'il peut sauter des pièces, le tour du joueur sera alors inchangé.

A chaque saut, on utilise la méthode *Coordonnees pieceMangee(x,y,i,j : int)* qui retourne les coordonnées de la pièce mangée, pour ensuite l'enlever du plateau.

A la fin du tour, on utilise la méthode *boolean peutEtreMange(x,y : int)* sur chaque pièce du joueur. S'il s'avère qu'une pièce peut être mangé au tour d'après, si l'option *sautObligatoire* a été choisie par les joueurs en début de partie, tout déplacement du joueur suivant n'étant pas un saut est bloqué, un message d'erreur est affiché si nécessaire.

La méthode *boolean partieFinie()* retourne vrai si un joueur ne possède plus de pion ou ne peut plus en déplacer. La partie se termine et un message affiche le vainqueur du jeu.

La classe *Souris.java* va permettre aux joueurs de déplacer leur pièce grâce au clique de la souris. En effet, elle est issue de l'interface *MouseListener* qui comprend de nombreuses méthodes prédéfinies. En modifiant la méthode *void mousePressed(MouseEvent arg0)* on est capable de récupérer l'endroit où le joueur clique pour ensuite effectuer des actions sur le damier placé en attribut.

2. Les classes

2.1 Case

La classe *Case* possède une couleur et une pièce. Pour les cases ne contenant aucune pièce, on fixera l'attribut *piece* en *null*. Il est alors impératif de vérifier à chaque fois si la pièce d'une case existe ($\neq$ *null*) avant de demander la couleur d'un pion ou d'une reine.

2.2 Coordonnes

La classe *Coordonnees* ne possède aucune méthode. Elle a été créée dans le seul but de concevoir une fonction (présente dans la classe *Damier*) qui retournerait les coordonnées d'une pièce mangée.

2.3 Couleur

La classe *Couleur* est une classe énumération. Elle contient uniquement les deux couleurs nécessaires : Blanc et Noir.

2.4 Damier

La classe *Damier* est la classe principale du programme. Elle regroupe l'ensemble des procédures de déplacement et d'affichage du jeu. Grâce à la classe *JPanel*, elle hérite de la procédure *paint()* qui va permettre ainsi en la redéfinissant d'afficher le damier et ses pièces. Ses attributs comme sa taille ou encore son booléen *sautObligatoire* est fixé par l'utilisateur au début du jeu. Pour manipuler les pièces, il possède un tableau deux dimensions de *Case* permettant ainsi de connaître l'ensemble du plateau à chaque instant.

2.5 Joueur

La classe *Joueur* contient en attribut le pseudo (fixé dans la classe *Menu*) et la couleur du joueur. Le premier joueur a la couleur blanche, le second la couleur noire.

2.6 Lanceur

La classe *Lanceur* est la classe *Main* de notre projet. Elle permet de lancer le jeu. Elle lance d'abord le menu puis affiche le damier et les pions afin de débuter une partie.

2.7 Menu

La classe *Menu* demande aux joueurs le type de partie qu'ils veulent jouer afin de déterminer la taille du plateau. Elle demande également si les joueurs veulent jouer avec l'option "Saut Obligatoire".

Les joueurs choisissent également leur pseudo grâce à cette classe. Cette classe sert uniquement au démarrage de la partie, avant même que le plateau soit affiché.

2.8 Piece

La classe *Piece* est la sous-classe de tous les pions et reines du plateau. Elle contient en attribut la couleur de la pièce et contient une seule méthode qui permet de dessiner la pièce selon sa couleur et selon s'il s'agit d'un pion ou d'une reine. Elle appartient forcément à une case c'est pour cette raison qu'il n'y a pas d'attribut *Coordonnees*.

2.9 Pion

La classe *Pion* hérite de la classe *Piece*. Elle ne contient pas plus de méthode ni d'attribut.

2.10 Reine

La classe *Reine* hérite de la classe *Piece*. Identiquement à la classe *Pion*, elle ne contient pas plus de méthode ni d'attribut.

2.11 Souris

La classe *Souris* est une des plus importantes du projet. En effet, elle permet aux joueurs de pouvoir jouer avec la souris. C'est elle qui va gérer l'intéraction homme-machine. Grâce à l'interface *MouseListener* déjà prédéfinie, elle a accès a plusieurs méthodes dont *void mousePressed(MouseEvent arg0)* qui écoute chaque pression d'un bouton sur la souris. Avec les fonctions *getX* et *getY* on récupère les coordonnées où le joueur clique dans la fenêtre. Ces coordonnées sont ensuite filtrés puis selon la case, la pièce indiquée des décisions sont prises.

3. Problèmes rencontrés

3.1 Affichage

L'affichage du damier et des pièces fut notre première mission lors du codage du jeu. Il nous semblait primordial d'observer directement ce que l'on allait coder afin de remarquer les éventuelles erreurs ou oublis.

Nous avons donc utiliser l'environnement *swing* du paquet *javax* afin de pouvoir afficher notre plateau. L'environnement étant tout nouveau pour nous, il nous a fallu beaucoup de temps afin de comprendre comment il s'utilise. Avec *JFrame* nous pouvions ouvrir une nouvelle fenêtre et la renommer "Jeu de dames", puis avec l'héritage de la classe *JPanel* la méthode *paint* pouvait dessiner ce que l'on souhaitait sur la fenêtre en question.

Au départ nous avons fait cet affichage sur un plateau d'une taille fixée 10 par 10. Puis nous voulions étendre cet affichage pour n'importe quelle taille donnée, il nous a fallu alors introduire une échelle TAILLE/taille et comprendre comment bien l'utiliser (TAILLE correspond à la taille de la fenêtre et taille correspond à la taille du damier). Cette échelle nous a permis notamment de centrer les pièces dans chaque case du damier. Ainsi, nous pouvons fabriquer un damier de la taille que l'on souhaite (réglable dans la classe *Lanceur*).

Malheureusement, un mini problème persiste encore au niveau de l'affichage. Il s'agit d'une barre noire verticale à droite de la fenêtre qui apparaît uniquement au lancement d'une partie. Il était pour nous impossible de régler ce léger problème ne comprenant pas l'origine de l'existence. Après réflexion, nous avons compris qu'il s'agissait d'un problème de paquet et non de codage, nous avons donc abandonné la recherche de la solution. Ce problème apparaissant en moyenne 2 fois sur 3 lancement de partie disparaît au premier clique de la souris, ce qui est au final peu dérangeant.

3.2 Ecoute de la souris

L'écoute de la souris nous a également posé de nombreux problèmes. La classe *MouseListener* étant également nouveau pour nous, la comprendre était pas chose facile au début du projet. En effet, nous voulions au départ écouter chaque case afin de récupérer la case cliquée directement. Nous avons alors remarqué des anomalies sur le comportement des méthodes de la classe. Nous avons donc décidé d'écouter le plateau entier puis de filtrer les coordonnées de l'endroit cliqué afin de déterminer la case correspondante.

Conclusion