Latroncules

Sofiane AMARI et Marwan OTHMANN

10 mai 2013

1 Interface homme-machine de l'application

L'application se décompose en six parties : l'accueil, la création de nouveaux joueurs, la modification des joueurs, le classement de ces joueurs, le chargement de parties sauvegardées et enfin la partie réservée au jeu. Ces six parties ont été programmées à l'aide des librairies Swing et Awt. Nous allons exposer le fonctionnement de ces diverses parties aux cours de ce chapitre.

1.1 L'accueil

La fenêtre d'accueil permet d'accéder à toutes les fonctionnalités de l'application en un clic. Techniquement, elle est divisée en trois :

- un JPanel dans lequel sont affichées les règles du jeu ainsi qu'un rappel de la forme des pions
- plusieurs JButton (Nouveau jeu, Charger ...) réunis au sein d'un même JPanel donnent accès à toutes les fonctionnalités de l'application
- une JmenuBar joue le même rôle

1.2 Nouveau joueur

La fenêtre d'ajout d'un nouveau joueur se compose de deux JtextField ainsi que de deux Checkbox qui permettent respectivement de renseigner le nom, le prénom et le sexe du joueur. On y trouve aussi un Jlabel dans lequel est affiché l'avatar du joueur qui peut être changé grâce à un Jbutton donnant accès à une fenêtre JfileChooser. Un FileFilter a été mis en place pour filtrer le fichier choisi par l'utilisateur. Pour se faire la classe FiltreExtension a été créée. Elle prend en paramètre un tableau de String qui renseigne sur les extensions acceptées ainsi qu'un String qui permet de choisir le message d'erreur à afficher en cas de maladresse de l'utilisateur. Cette classe est composée de deux fonctions principales. La fonction accept qui prend en paramétre le fichier choisi et qui teste si l'extension est bonne (après avoir couper le nom du fichier à partir de la dernière occurrence de « . » et mis cette chaîne de caractère en minuscule) à l'aide de la fonction appartient.

1.3 Options

Le bloc ≪options≫ permet de :
– Réinitialiser les scores des joueurs
– Supprimer tous les joueurs
– Modifier un joueur

Cette dernier fonctionnalité permet de remettre à zéro le score d'un joueur, de le supprimer ou de modifier ses attributs. Pour choisir le joueur à modifier, la fenêtre contient une JComboBox à laquelle est attribué un ItemListener. Cela permet de mettre à jour les champs correspondant aux attributs du joueur. La sauvegarde de ces champs se fait de la même manière que dans le bloc qui ajoute un nouveau joueur.

1.4 Classement

La fenêtre qui affiche le classement des joueurs se compose d'un JtabbedPane qui la divise en deux onglets. L'un permet d'afficher le classement par Head2Head et l'autre par ratio. Pour classer les joueurs nous avons créé deux fonctions qui implémentent la classe Comparator. Une sert à comparer les joueurs en fonction de leur Head2heat et l'autre en fonction de leur ratio.Cela nous a permis à l'aide de la fonction statict sort de la classe Collection de trier l'ArrayList contenant les joueurs selon le critère désiré. Afin de donner une touche plus conviviale, nous avons mis en place un podium sur lequel est affiché les avatars des trois premiers joueurs du classement. Les high score sont calculés grâce à deux variables accessibles via une instance de joueur (nbPartie et head2head) et sont donc enregistrés indirectement par leur biais.

1.5 Chargement

Le clic sur le bouton ≪Charger≫ déclenche l'ouverture d'un JfileChooser. L'extension .lat (Fichier Latroncule) est imposée afin d'éviter d'ouvrir un fichier n'ayant aucun rapport. Ce chargement déclenche la sérialisation décrite plus tard dans le rapport.

1.6 Plateau de jeu

Pour lancer une nouvelle partie les utilisateurs doivent choisir les joueurs qu'ils vont incarner. Ils accèdent à une fenêtre composée de deux JcomboBox par l'intermédiaire desquels ils pourront choisir leur joueur. Un bouton leur permet de valider leur choix. Afin de ne pas avoir à caster des objets les joueurs choisis sont récupérés grâce à leur position dans la JcomboBox, cette position étant la même que celle ou ils se trouvent dans l'ArrayList. Une fois les joueurs récupérés, des tests sont effectués pour respecter les consignes comme celles indiquant qu'un joueur ne peut pas jouer plus de deux parties en même temps. D'autres tests sont effectués pour, par exemple, éviter que le joueur 1 et le joueur 2 ne soient identiques.

Le plateau de jeu se décompose en trois parties :
– à droite la partie concernant le joueur 1
– au centre les damiers ainsi que trois boutons (enregistrer, armistice et déposer les armes)
– à gauche la partie réservée au joueur 2

Le partie concernant les joueurs permet d'afficher leur avatar, leur nom et prénom, leur nombre de victoires, le nombre de pions à leur disposition ainsi qu'une indication sur le joueur ayant la main. L'appui sur sauvegarder ouvre un JfileChooser permettant de choisir le nom du fichier. L'extension .lat (Fichier Latroncule) est imposé. Un JoptionPane permet au joueur d'accepter ou non l'armistice que son adversaire lui propose. L'appuie sur la croix rouge de fermeture de la fenêtre déclenche l'ouverture d'un JoptionPane qui demande aux utilisateurs s'ils veulent réellement quitter et si tel est le cas, s'ils désirent sauvegarder leur partie.

1.7 Représentation graphique du damier

1.8 Damier

Le damier est composé de 8 fois 8 cases. L'objet Case hérite de jPanel. Le damier est donc composé de 64 jPanel.

1.9 Pions

Il existe deux sortes de pions : les Larrons et les Latroncules. Ils ont tous les deux en commun de nombreuses caractéristiques. Ils héritent donc tous les deux de la classe abstraite Pion. Un pion est représenté sur le damier par une imageIcon posée sur un JLabel. Selon qu'il soit Larron ou Latroncule l'image du pion changera, l'image étant une caractéristique propre aux classes filles. Les Larrons sont les tours du jeu d'échec tandis que les Latroncules sont représentés par les pions.

2 Les événements, déplacement des pions

2.1 Mécanisme de déplacement des pions

Si l'on clique sur un pion, cela voudra dire que l'on souhaite le déplacer et si l'on clique sur une case vide, cela signifie que l'on souhaite déplacer un pion sur cette case. Le mécanisme de déplacements des pions repose donc sur deux MouseListener :
– On pose un MouseListener sur ImageIcon de la classe Pion. Il s'agit de l'image du pion. Le listener réagira au clique (utilisation de void mouseClicked).
– On pose un MouseListener sur chaque case (JLabel). Le listener réagira au clique (utilisation de void mouseClicked).

Au sein des fonctions mouseClicked, on insert les mécanismes :

– internes du jeu, touchant aux structures de données On modifie par exemple les coordonnées du pion sélectionné avec un setter ou encore on change le joueur qui a la main
– graphiques du jeu. On supprime par exemple l'image d'un pion sur la case de départ et on la fais réapparaitre sur la case de destination.

2.2 Déplacements permis

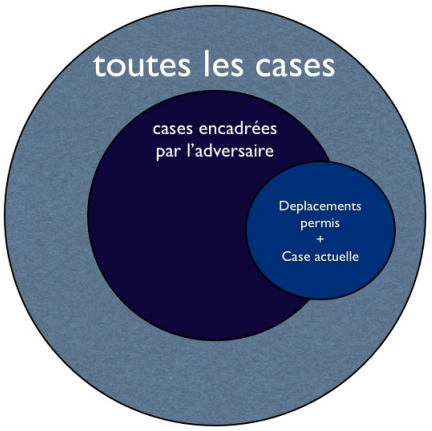
Afin de faciliter la compréhension des règles et rendre le jeu plus agréable, les casses ou le pion peut se déplacer sont colorées en vert tandis que la case ou le pion se trouve en jaune. La fonction abstraite dePlacementsPermis() de pion se charge de calculer ces cases. Elle est implémentée de façon différente selon que le pion soit un Larron ou un Latroncule pour répondre aux règles du jeu. Lorsque le joueur clique à l'extérieur des cases permises, il ne se passe rien.

2.3 Manger les pions

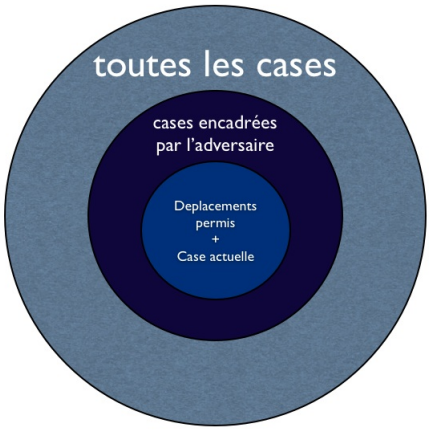
Pour que les pions puissent se manger selon les règles du jeu, il est nécessaire de connaitre :

– les cases ou chaque pion peut se déplacement (les cases vertes). La fonction dePlacementsPermis() de Pion remplit ce rôle en renvoyant pour un pion donné, les coordonnées des cases ou le déplacement est permis.
– Les cases encadrées par les pions de chaque joueur. La fonction casesEncadrees(GroupeDePions pGroupe) de Damier retourne les coordonnées des cases encadrées.

A chaque tour on parcourt tous les pions du plateau et on se demande pour se pose la quesion suivante. Est-ce que, une des cases où il peut se déplacer ou sa propre case, n'est pas encadrée par l'adversaire ?



Le pion peut fuir
on ne fait rien



Le pion est en échec
Il faut le supprimer

Le fait qu'un joueur déplace un de ses pions dans une position impliquant sa suppression (le pion se suicide) nécessite de lancer la fonction mange sur ses propres pions et non pas seulement sur les pions de l'adversaire.

2.4 Sérialiser pour enregistrer, désérialiser pour charger

Pour enregistrer, les objets Parties sont rendus persistant grâce à la sérialisation. La classe Partie permet de stocker toutes les données inhérentes à une partie : les joueurs qui y participent, les pions en jeu et le joueur qui a la main. La fonction Enregistrer() de Damier sérialise et la fonction charger() de Damier retourne une partie à partir d'un fichier en le déserialisant.

Lorsque l'on charge une partie, les joueurs de celles-ci sont chargés sous contrôle. Si les joueurs ne sont pas dans la liste de l'application actuelle (le fichier de joueur ayant par exemple été remis à zéro), on les y ajoute. On prend soin de mettre alors à 1 la variable indiquant le nombre de parties en cours. Nous avons fait le choix d'effacer l'historique (ratio, nombre de victoires) des joueurs importés pour qu'un nouvel entrant ne puisse pas faire valoir des statistiques invérifiables.

Lors de l'importation d'une partie, les avatars (des images) ne sons pas sauvegardés. Nous avons fait le choix de mettre l'avatar par défaut à des joueurs importés ne se trouvant pas dans la liste des joueurs. Si le joueur existe déjà dans la liste, on prend soin de regarder s'il n'a pas déjà deux parties en cours. Si tel est le cas, il faut l'empêcher de charger la partie.

L'enregistrement des joueurs (et donc indirectement des scores qui sont stockés en leur sein) se fait grâce à la fonction enregistrerJoueurs de la classe Fenetre. Le fichier de sauvegarde des joueurs se nomme ≪joueurs.dat≫ et se trouve dans le même répertoire que les .class.

3 Méthodes de travail

3.1 Outils utilisés

Le gestionnaire de version Git a été utilisé pendant certaines phases du projet. Gitlab a été installé sur une machine virtuelle louée chez un hébergeur. Il s'agit d'une interface web facilitant l'utilisation de Git et contenant un système de tickets. Dans d'autres phases, dropbox a été d'une grande utilité.

NetBeans a été privilégié et le format javadoc des commentaires respecté pour faciliter la compréhension du code.

3.2 Répartition du travail

Nous avons décidé de nous répartir les taches en terme de domaine de compétences afin que chacun puisse s'y spécialiser. Ainsi Marwan Othmann avait en charge :

- la partie interface graphique du projet
- La gestion des joueurs
- Le contrôle du nombre de parties en cours
- Le calcul et l'affichage des High score

Sofiane Amari se chargeait de :

- Modélisation du jeu
- l'algorithmique et les mécanismes du jeu ainsi que leur implémentation
- La création et l'affichage du damier
- L'enregistrement et le chargement des parties et des joueurs

Cette répartition du travail nous a permis d'éviter au maximum de l'ingérence dans le code de l'autre et de mettre facilement notre travail en commun.

Nous réalisions des points régulièrement sur Skype ou par mail pour s'assurer du bon avancement du projet.

4 Remarques

Certaines règles du jeu de Latroncule disent qu'un pion encadré perd son pouvoir offensif (il ne peut participer à un encadrement). Les règles de Becq de Fouquières sont ambiguës sur ce point. Nous avons fait le choix de ne pas prendre en compte cette règle.

Une règle coutumière des échecs impose que lorsqu'un joueur a touché un pion, il est obligé de jouer celui-ci et ne peut pas modifier son choix. Cette règle ayant une réalité dans l'enfance d'un des membres de groupe, elle a été implémentée.

Nous considérons que l'armistice est une forme de défaite et le comptons donc comme tel dans les statistiques.

5 Bonus développés

Les bonus suivants ont été développés :

- Système de gestion de plusieurs joueurs
- Supprimer un joueur
- Modifier un joueur
- Enregistrement automatique des joueurs
- Enregistrer et charger des parties, avec synchronisation des joueurs chargés
- Faire une armistice
- Abandonner une partie
- Mettre les scores à zéro pour tous les joueurs

– Mettre les scores à zéro pour un joueur
– Classement des joueurs par ratio ou par head2head
– gestion d'avatar
– Assistance au joueur : déplacements permis pour les pions affichés