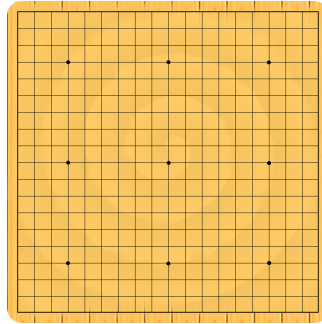


FACULTÉ DES SCIENCES DE MONTPELLIER



26 avril 2015
Rapport de projet L2

Assistant de mémorisation d'ouvertures au jeu de Go

Florent Occhiuzzi , Rémi Debuyer , Jordan Ferrad



Sommaire

I	Le Projet	2
1	Encadrement	3
2	Introduction	4
II	L'implémentation	5
3	le langage de programmation	6
3.1	Java (sun)	6
3.2	avantages	6
3.3	inconvenients	6
4	Conception	7
4.1	Interface	7
4.1.1	Cahier des charges	7
4.1.2	Découpage de la fenetre	7
4.1.3	Programation evenementielle	8
4.2	Moteur de calcul	10
4.3	Parsing-Deparsing	10
5	Diagrammes	11
5.1	Classes	12
III	Le logiciel	13
6	Manuel d'utilisateur	14
6.1	Compilation	14
6.2	Utilisation	15
7	Mises a jours en prévision	16
7.1	Applet	16
7.2	Fenetre Redimensionnable	16

Première partie

Le Projet

Chapitre 1

Encadrement

- **responsable de l'UE** : Christian Retoré
- **professeur référent** : Pierre Pompidor
- **date d'attribution du sujet** : 10 Janvier 2015
- **date de retour du présent rapport écrit** : Dimanche 26 Janvier 2015
- **date de soutenance** : Vendredi 22 mai 2015 9h45
- **jury de la soutenance** : Christian Retoré , Federico Ulliana , Hervé Dicky

Chapitre 2

Introduction

Dans le cadre de nos études en 2ème année en informatique, nous nous sommes vu proposer une liste de projets afin de parfaire nos connaissances dans les différents langages les plus populaires, ainsi que de manière plus générale, nos compétences en algorithmique. Nous avons choisi le sujet **Assistant de mémorisation d'ouvertures au jeu de Go** pour la grande popularité de ce jeu qui a réussi à traverser les millénaires, mais aussi car il représente pour nous un challenge de taille, sur le plan de la difficulté d'implémentation.

Voici les consignes du projet telles qu'elles nous ont été fournies :

Assistant de mémorisation d'ouvertures au jeu de Go Encadrant : Pierre Pompidor.

Le jeu de Go est le seul jeu (classique) à information complète où les meilleurs joueurs battent les ordinateurs (même les très très gros).

Ce jeu se joue entre deux joueurs et sur un plateau (goban).

Dans ce jeu les ouvertures (fuseki) sont les coups joués en début de partie

et qui permettent aux deux joueurs d'occuper rapidement l'espace avec leurs pierres .

Un format de fichiers (SGF : Smart Game Format) permet de décrire une séquence de coups (et ses variantes possibles).

Le but de ce TER est :

de pouvoir grâce à une interface graphique poser des pierres

pour créer un fuseki (comme si deux joueurs se confrontaient) ;

d'enregistrer ces ouvertures dans des fichiers au format SGF ;

de pouvoir également y rajouter des commentaires.

Langages autorisés :

pour créer les fichiers SGF : C, Python (à préférer)

pour l'interface graphique : un framework web tel que le merveilleux

D3JS serait il le bon choix ?

Deuxième partie

L'implémentation

Chapitre 3

le langage de programmation

3.1 Java (sun)

Le langage de programmation a été laissé à notre choix. Après avoir légèrement hésité entre une implémentation en **C++** ou en **Java**, nous avons opté pour le 2ème choix.

3.2 avantages

- génération d' "**Applet**"¹ Web très facile
Cet élément est un plus car même s'il en existe déjà plusieurs, les visualiseurs de go sont très appréciés compte tenu de la popularité et de la communauté qui entoure le jeu.
- généricité du **code-source**
En effet, ce langage permet, notamment grâce à son système de packages à décomposer parfaitement et facilement son code de manière à valider l'orientation "**Poo**"².
- **portabilité** grâce à la machine virtuelle java

3.3 inconvénients

1. **performance** :
 - temps de chargement au démarrage
 - réactivité des événements
2. **code** :
 - taille de chaque fichier sources
 - pas de "headers"

1. Un applet est un logiciel qui s'exécute dans la fenêtre d'un navigateur web
2. Programmation Orientée Objet

Chapitre 4

Conception

4.1 Interface

Jordan Ferrad

4.1.1 Cahier des charges

- utilisation instinctive
- aspect graphique agréable et bonne resolution
- toutes les fonctions de base attendues doivent etre présentes
- lisibilité du code source
- sons

4.1.2 Découpage de la fenetre

- Zone supérieure :
Cette zone sera constituée de la barre de boutons qui sera suffisante pour realiser toutes les actions nécessaires.
- Zone inférieure gauche :
C'est la Zone de jeu qui comportera un espace dévoilant un fond d'écran , et le "Goban".
- Zone inférieure droite :
cette derniere comportera :
 1. l'arbre de visualisation de l'historique des "**Variations**"¹ et des "**Coups**"².
 2. une zone de notifications

1. Correspond a toutes les nouvelles possibilités de jeu apres un coup
2. Placement d'un pion , ce qui change de variation

4.1.3 Programmation événementielle

les 3 zones en java

Le principe de la **programmation événementielle**³ permet de produire une interface facilement en utilisant au choix plusieurs **packages**⁴ comme JButton, JPanel, AWT et beaucoup d'autres. Pour obtenir un résultat complet nous avons besoin des 3 précédemment nommées.

Sur le plan de la programmation, la traduction des 3 zones se fait par l'ajout de 3 JPanel au "ContentPane" d'origine de la fenêtre de l'application Java. Ces 3 Panels (haut, droite, gauche) doivent avoir été réglés à l'aide d'un Layout au choix, avant d'être ajoutés au ContentPane principal (fond), qui lui-même doit être réglé. Voici le code qui correspond :

```
...
public class Fenetre extends JFrame
{
    ...
    private Panneau haut = new Panneau(new Color(255,255,255,230));
    private Panneau gauche = new Panneau(new Color(255,255,255,0));
    private Panneau plateau = new Panneau("plateau.png", false);
    private Panneau droite = new Panneau(new Color(255,255,255,230));
    ...
    public Fenetre(String titre)
    {
        ...
        this.setContentPane(fond);
        ...
        haut.setPreferredSize(new Dimension(1000,30));
        gauche.setPreferredSize(new Dimension(800,770));
        droite.setPreferredSize(new Dimension(200,770));
        ...
        fond.setLayout(new BorderLayout());
        fond.add(haut, BorderLayout.NORTH);
        fond.add(gauche, BorderLayout.WEST);
        fond.add(droite, BorderLayout.EAST);
        ...
    }
}
...
```

3. En informatique, la programmation événementielle est un paradigme de programmation fondé sur les événements. Elle s'oppose à la programmation séquentielle.

4. A Java package is a mechanism for organizing Java classes into namespaces similar to the modules of Modula

Les evenements

La grille du "**Goban**"⁵ est implementee par une "GridLayout" de plusieurs "Jbutton" dont chaque fond d'ecran qui peut etre une pierre noire ou une pierre blanche va changer en fonction des clics de l'utilisateur.

pour rendre cela possible il a fallu creer une **classe interne**⁶ pour que chaque pion de la grille aie son Propre "Listener"⁷.

J'ai donc decide de garder cette methode d'ecoute des evenements pour toutes les autres entites de l'interface,uniquement dans un soucis de lisibilité.

Cette portion de code ajoute un **listener** a chaque case de la grille du "Goban" avant de l'ajouter lui meme a la grille :

```
public class Fenetre extends JFrame
{
    ...
    public Fenetre(String titre) //constructeur
    {
        ...
        for(int i=0 ; i< 19 ; i++)
            for(int j=0 ; j< 19 ; j++)
            {
                (curent.getTabPion())[i][j] = new Pion(30,i,j,curent); /*
                (curent.getTabPion())[i][j].addActionListener(new PionListene
                plateau.add((curent.getTabPion())[i][j]); /*
            }
        ...
    }
    ...
    //ecouteurs sur les pions de la grille via classe interne
    class PionListener implements ActionListener
    {
        public void actionPerformed(ActionEvent arg0)
        {
        }
    }
    ...
}
```

5. Plateau de jeu composé d'une grille 18x18

6. Classe contenue dans une autre, la classe contenue a acces aux attributs et methodes de la contenante

7. typique de la programmation événementielle , attend une action pour en declencher une autre

4.2 Moteur de calcul

Rémi Debuyser

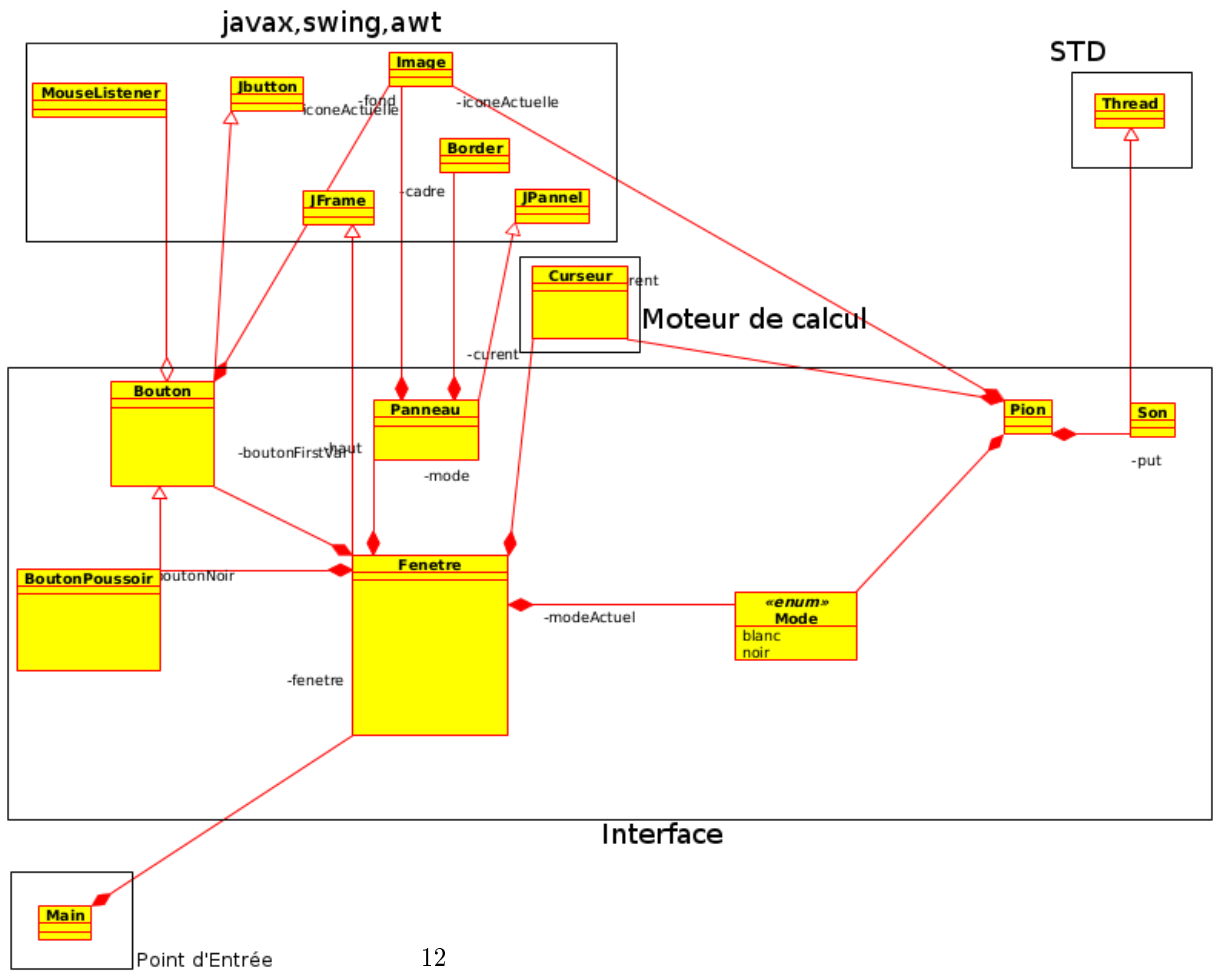
4.3 Parsing-Deparsing

Florent Ochiuzzi

Chapitre 5

Diagrammes

5.1 Classes



Troisième partie

Le logiciel

Chapitre 6

Manuel d'utilisateur

6.1 Compilation

Voici les instruction pour les différentes compilations et exécution en ligne de commande.

















```
-----  
compile:  
    javac Beta/Interface/*.java  
-----  
run:  
    java Beta.interface.Main  
-----  
creer jar:  
    jar cvfm executable.jar META-INF/MANIFEST.MF *  
-----  
executer jar: (deux possibilit\'es)  
    1)  
    chmod 777 executable.jar  
    ./executable.jar  
    2)  
    java -jar executable.jar  
-----
```

A noter que pour la création du .jar, le fichier **MANIFEST.MF** n'est pas obligatoirement nommé ainsi , et n'est pas non plus forcément dans un dossier nommé **META-INF**.

Il s'agit la d'une convention qui est tres utilisée par les develloppeurs.

6.2 Utilisation

Voici le descriptif des actions liées au différents boutons :

-  Retourner a la toute **premiere variation**
-  Aller a la toute **derniere variation**
-  Aller a la **variation precedente**
-  Retourner a la **variation suivante**
-  Retourner au tout **premier coup** de la Variation actuelle
-  Retourner au tout **dernier coup** de la Variation actuelle
-  Aller au **coup precedent**
-  Retourner au **coup suivant**
-  Passer en **mode Blanc** pour ne poser que des pions blancs
-  Passer en **mode Noir** pour ne poser que des pions noirs
-  Passer en **mode alternatif** pour alterner entre noir et blanc
-  **Enlever tous les pions** du goban
-  **Exporter** l'ouverture au format .sgf
-  **Importer** (ouvrir) une ouvertue au format .sgf
-  **Sauvegarder** les modifications dans le fichier importé
-  Activer ou désactiver **les sons**

Chapitre 7

Mises a jours en prévision

7.1 Applet

Le systeme d'applet web permettra de rendre le programme accessible en ligne.

En effet, La spontanéité de l'utilisation directement sur internet sera un aspect non negligeable, pour poursuivre le développement du programme a l'avenir.

7.2 Fenetre Redimensionnable

Cette amélioration va necessiter une refonte importante du code source car la maniere dont les elements sont placés va devoir etre gérée differement au niveau des "JPanel" et de leur(s) contenu(s)