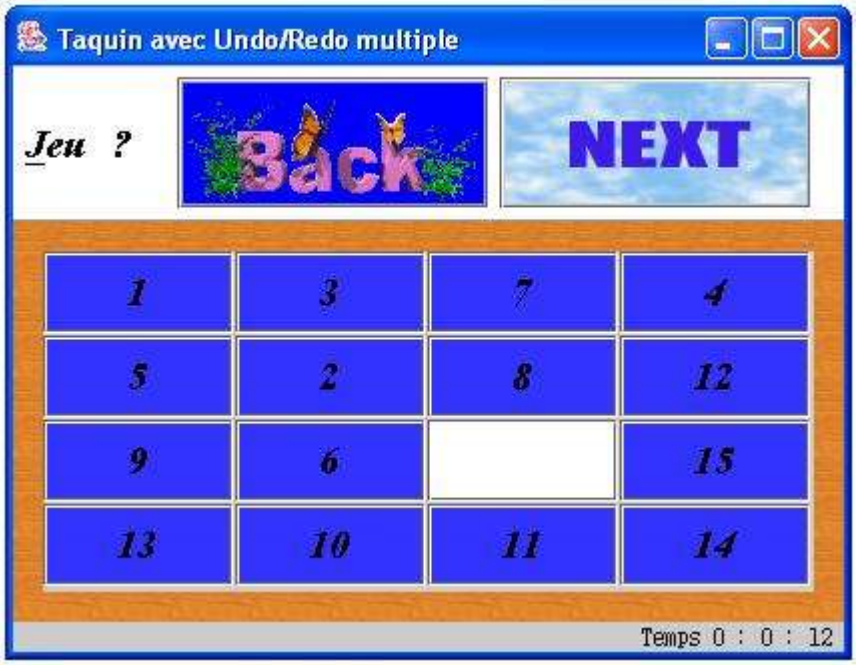
CAPELLIER Sebastien
ZORGATI Maher

Projet

	L	E
T	A	Q
U	I	N

Plan du Rapport



Chapitre I: Présentation du projet

Objectif:

Ce projet consiste en l'implémentation du jeu Taquin en utilisant le langage JAVA.

L'objectif du projet est de mettre en œuvre les techniques de la programmation orientée objet afin de réaliser avec SWING et AWT une interface utilisateur. Un soin particulier a été pris pour la lisibilité, l'ergonomie et la cohérence de l'interface.

Quelques explication sur le taquin:

Le *taquin* se joue seul sur une grille rectangulaire. Un jeton remplit chacun des cases de la grille sauf une. La case restante de la grille est vide. Les jetons sont numérotés à partir de 1. L'objectif est que les jetons soient disposés en ordre et que la case libre soit celle en bas et à droite.

Si la grille est par exemple de taille 4×4, il y a donc 15 jetons numérotés de 1 à 15. Une remplissage de la grille possible est montré ci-dessous. À droite se trouve le remplissage à obtenir.

Configuration initiale

12	**3**	**11**	**8**
2	**15**	**5**	**7**
1		**6**	**9**
10	**4**	**13**	**14**

Configuration à obtenir

1	**2**	**3**	**4**
5	**6**	**7**	**8**
9	**10**	**11**	**12**
13	**14**	**15**	

Le seul mouvement possible est de déplacer dans la case vide un des quatre jetons qui se trouve dans une case adjacente à la case vide. Par *adjacente*, on entend, au dessus, en dessous, à gauche ou à droite. La case où se trouvait le jeton devient vide.

Si dans la configuration ci-dessus, le jeton 6 se déplace, on obtient le coup suivant.

Deplacement

12	**3**	**11**	**8**
2	**15**	**5**	**7**
1	**6**		**9**
10	**4**	**13**	**14**

Réalisation:

Le programme est construit sur le modèle de l'architecture MVC (Model-View-Controller). ce modèle a été choisi car il permet une interaction entre les différentes classes, et donc le découpage des responsabilités au sein d'un ensemble de classes. Plus besoin alors de dupliquer les données pour pouvoir les manipuler puisque le composant ne contient plus physiquement les données mais une référence sur celles-ci. Une instance d'un même modèle de données peut alimenter simultanément plusieurs composants.

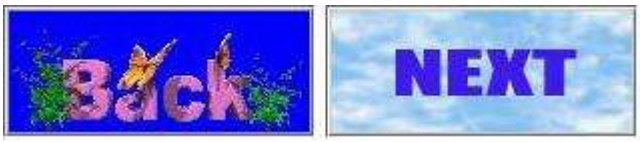
L'interface nous a fournit une première approche que nous avons par la suite adaptée pour arriver au résultat voulu. Ainsi par exemple le contrôleur a été réparti entre les différentes actions de la vue…

Lorsque l'AWT et SWING offrent des fonctionnalités similaires, les objets SWING ont été utilisés en priorité. L'interface doit propose au joueur une configuration initiale et gére une notion de score. Ce score pourra prend en compte la taille de la grille et le temps écoulé pour atteindre la solution.

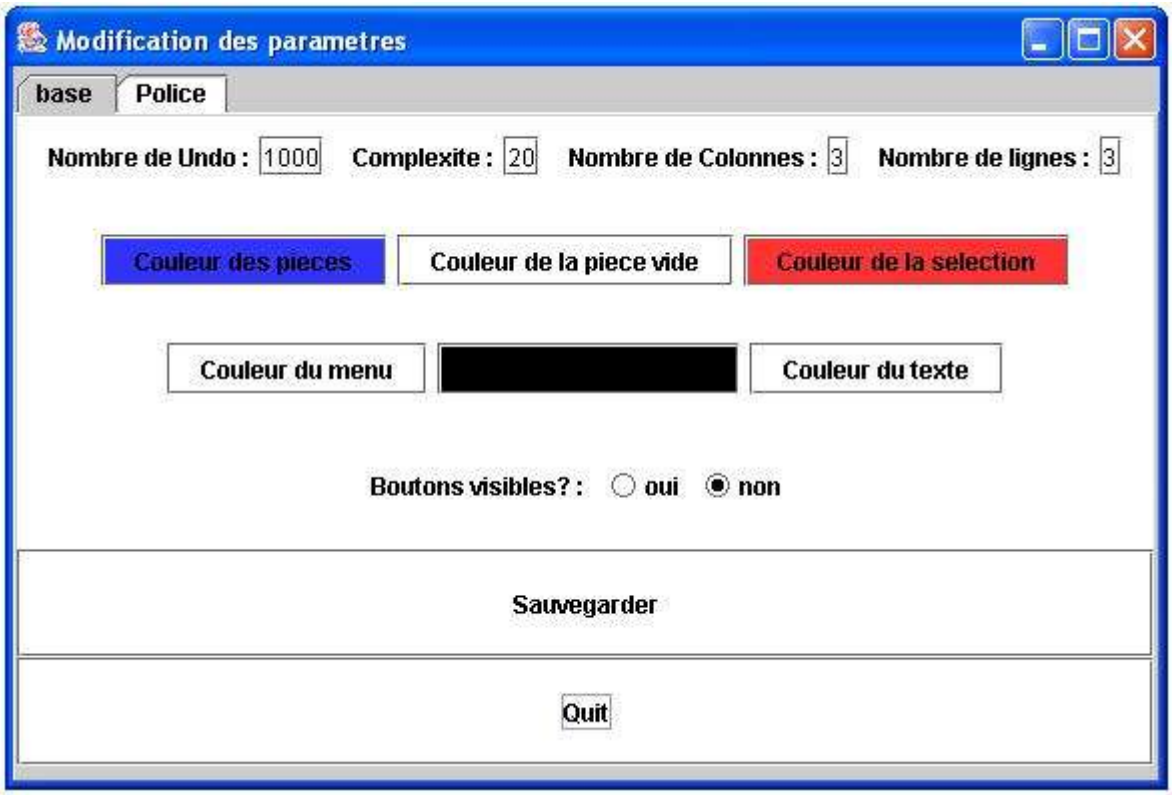
L'interface offre à l'utilisateur la possibilité de sauver une partie en cours et de reprendre une partie sauvée au préalable. Ceci se faire grâce à XML.

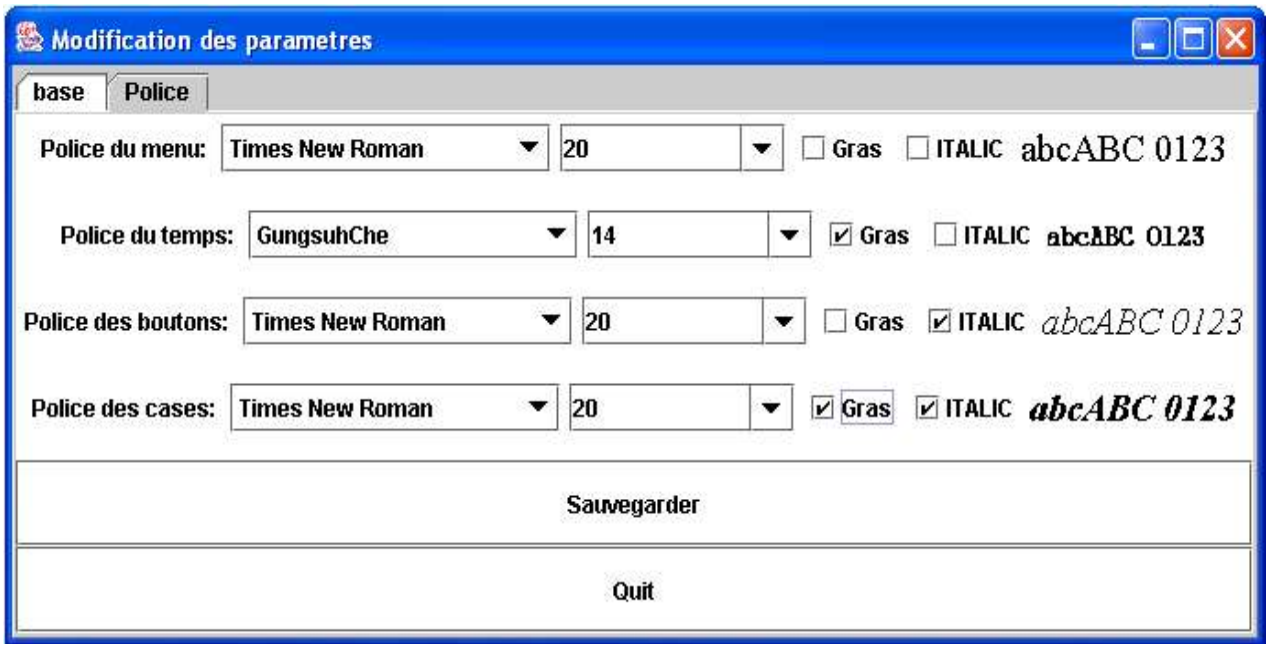
```
<?xml version="1.0" encoding="UTF-8" ?>

 <string>Times New Roman</string>
 <int>3</int>
 <int>20</int>
 </object>
 </void>

 <string>GungsuhChe</string>
 <int>0</int>
 <int>12</int>
 </object>
 </void>

 <string>Times New Roman</string>
 <int>3</int>
 <int>20</int>
 </object>
 </void>

 <string>Times New Roman</string>
 <int>3</int>
 <int>20</int>
 </object>
 </void>
 </array>
 </java>
```

L'interface permet aussi à l'utilisateur de revenir en arrière dans la partie en annulant un certain nombre des derniers mouvements. Il sera alors possible de repartir en avant en effectuant à nouveau les mouvements annulés.



Le jeux a été conçu de façon à ce que l'utilisateur puisse le personnalisé à souhait (couleur, complexité du jeux, nombre de case, périmètre de l'aire de jeu…).





Enfin, le choix des pièces s'est porté sur les numéros car c'est une façon simple intuitive d'ordonner des objet (le rangement d'image peut prêter à confusion dans l'ordre), ainsi les pièces deviennent bien plus personnalisables.

Chapitre II: Architecture des Classes

Comme expliqué précédemment, Le programme est construit sur le modèle de l'architecture MVC (Model-View-Controller).

Model:

Le model est gérer par la classe model.class du programme. La table de jeu du taquin est considérée comme une matrice n x m (n le nombre de colonne et m le nombre de ligne) ou chaque case comporte un numéro de 1 a n x m-1, et ou une case spéciale contient la valeur -1. Cette dernière représentant la case vide. Le constructeur de la class model permet ainsi la construction d'une tel matrice:

1	**2**	**3**	**4**
5	**6**	**7**	**8**
9	**10**	**11**	**12**
13	**14**	**15**	**-1**

La class model permet

La construction par défaut de la grille initiale (constructeur),
L'emplacement des différentes pièces(constructeur),
Les déplacements: autorisations de déplacement comprises (isMoveDoable, move),
Le mélange des pièces(RandomArea).
Gérer la condition de victoire(Victoire).

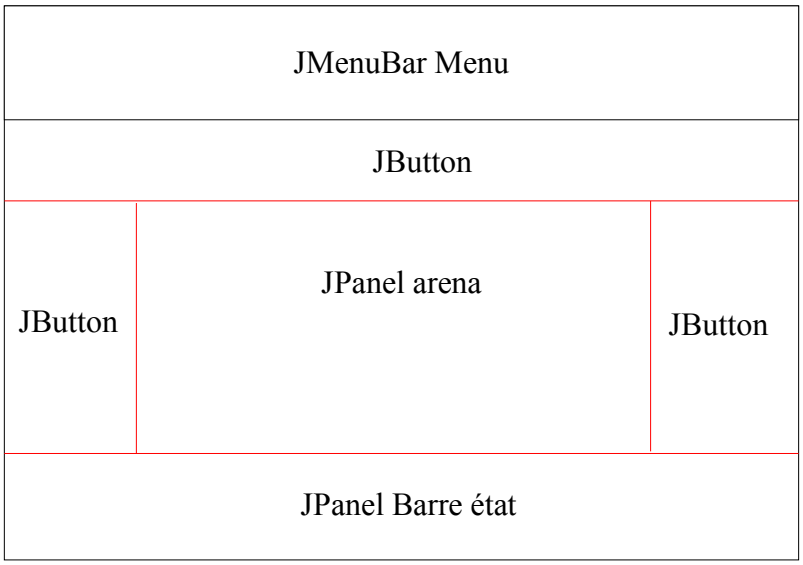
La class model porte surtout intérêt à l'emplacement de la pièce vide, car si l'utilisateur effectue des déplacements sur des cases proches de la pièces vide, c'est en fait la pièce vide que l'on déplace.

Remarque: Le mélange s'effectue notamment grâce à des déplacements aléatoires de la pièce vide, ce qui permet de conserver la possibilité de victoire.

View

Cette classe est comprise dans la classe Taquin. Elle étend la classe JFrame, elle permet ainsi de gérer la vue principale de la table de jeux, elle contient aussi les JFrame des paramètres et du score non-visible au début du jeu.

La fenêtre principale gerer par un BorderLayout contient un menu et un panel contenant le JPanel de l arène, les différents JButon et le JLabel du temps.



Le Menu:

Le Menu mis en place par la fonction loadMenu contient 3 sous menu et 1 Jpanel:

-menu de jeu
-menu info
-menu move, invisible mais permettant d avoir des raccourcis claviers très pratiques
-JPanel contenant les boutons undo/redo

JMenu Jeu	JMenu ?	JMenu move (invisible)	JPanel Placement

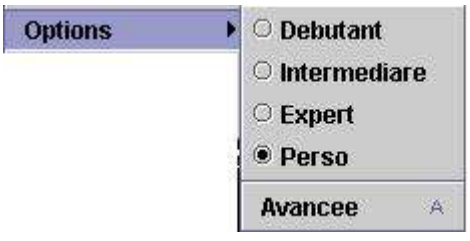
le menu jeu contient:

-Nouvelle partie: permettant de faire une nouvelle partie
-Charger partie: permettant de charger une partie
-Options: sous menu permettant de selectionner certaine options
-Quit: pour quitter la partie
-Quit and save: pour quitter et sauvegarder la partie en meme temps.

JMenu Jeu
JMenuItem Nouvelle partie
JMenuItem Charger partie
JMenu Options
JMenuItem Quit
JMenuItem Quit and Save

le menu options contient:

-Débutant: sélectionne un jeu de difficulté simple
-intermédiaire: sélectionne un jeu de difficulté intermédiaire
-expert: sélectionne un jeu de difficulté difficile
-perso: sélectionne un jeu de difficulté personnalisée
-avancee: lance la fenêtre des options avancees



JMenu Options
JRadioButtonMenuItem Debutant
JRadioButtonMenuItem Intermediaire
JRadioButtonMenuItem expert
JRadioButtonMenuItem Perso
JMenuItem Avancee

ButtonGroup

le menu d informations contient:
-un JMenuITem du tableau des scores

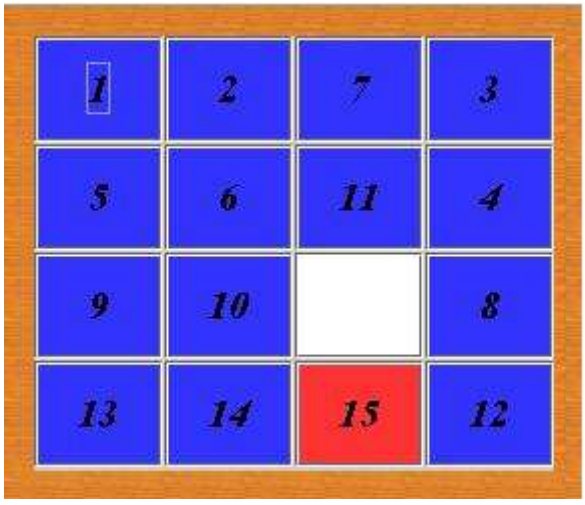
le JPanel placement contient
-leJButton redo
-le JButton undo

le menu move contient:
-le JMenuItem du bouton droite
-le JMenuItem du bouton gauche
-le JMenuItem du bouton haut
-le JMenuItem du bouton bas

L'arène de jeu

L'arène de jeu, mise en place par loadArena, est un GridLayout rempli de Case qui ne sont en fait que des extensions de boutons avec des informations supplémentaires.

JButton	**JButton**	**JButton**	**JButton**
JButton	**JButton**	**JButton**	**JButton**
JButton	**JButton**	**JButton**	**JButton**
JButton	**JButton**	**JButton**	**JButton**



La Barre d'état(ou de temps):

Elle permet de placer grâce a un Borderlayout le temps a un endroit appropriée, fonction loadBouton

JButton Bas	
	JLabel Temps

La fenetre Score:

c'est une simple JFrame ou les boutons et label sont placé grâce a un Gridlayout

JButton Score	**JButton Nom**	**JButton Temps**
JLabel Score	**JLabel Nom**	**JLabel Temps**
JLabel Score	**JLabel Nom**	**JLabel Temps**
JLabel Score	**JLabel Nom**	**JLabel Temps**
JLabel Score	**JLabel Nom**	**JLabel Temps**
JLabel Score	**JLabel Nom**	**JLabel Temps**
JLabel	**JButton Quit**	

Quit permet de quitter la fenetre

Remarque: les boutons Score, Nom et Temps ne sont la que pour l esthetique

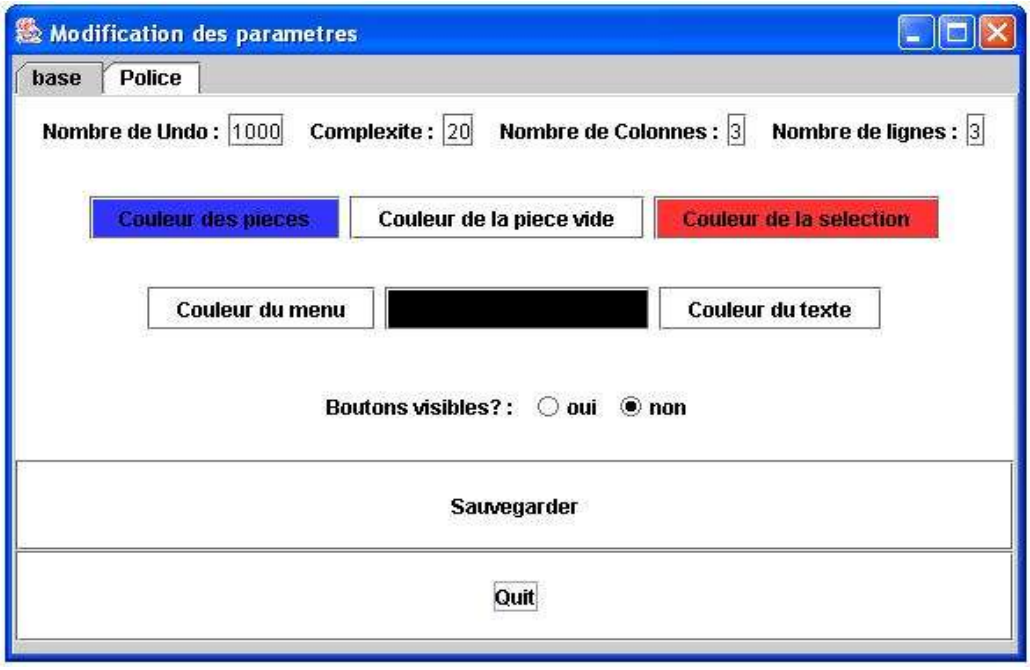
La fenetre Avancee:

C'est une JFrame contenant un JtabbedPane, celui ci permettant une navigation dans les options par onglet.

Cette classe permet à l'utilisateur de personnaliser le jeux; elle gère les couleurs, le périmètre de la grille, le nombre de pièces utilisées, la police…

L'onglet de base et gérer par un GridLayout

Onglet Base
JPanel Ptext
JPanel Pmodel
JPanel Pmenu
JPanel Pboutons
JButton sauvegarder
JButton exit



Ptext:
est une suite de JPanel contenant un Jlabel et JTextField

JLabel Titre	JTextField valeur a rentrer

Pmodel et Pmenu
sont des suites de boutons permettant de selectionner des couleurs

JButton Pieces	JButton de la Piece vide	JButton selections

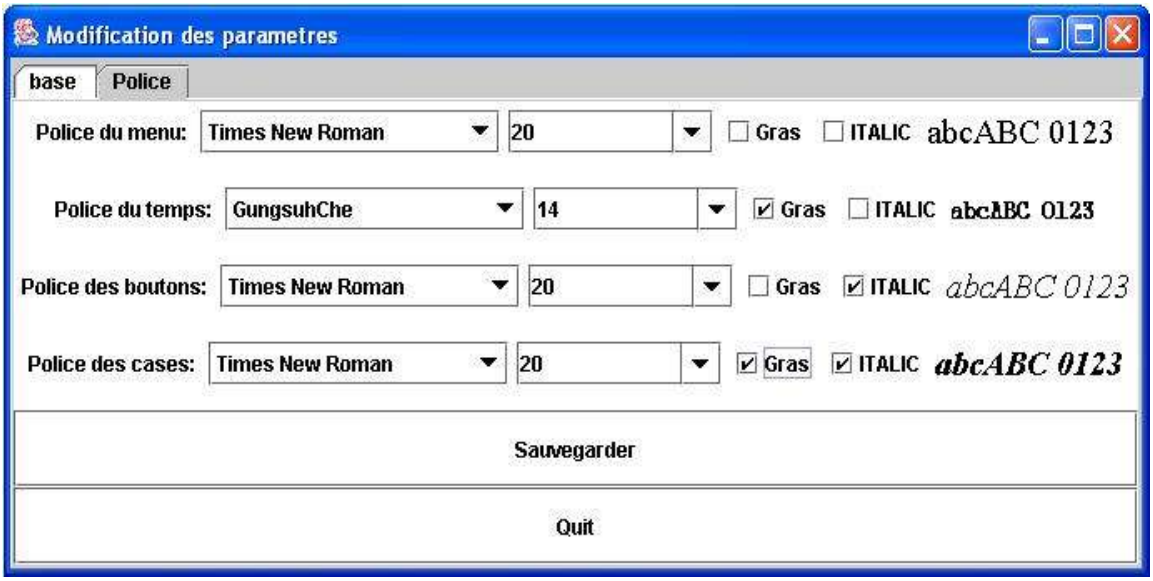
JButton Sauvegarder et exit
permettent de sauvegarder ou de quittez le menu d options

L'onglet de police est gérer par un GridLayout

Onglet Police
MaFontList Fmenu
MaFontList Ftemps
MaFontList FButton
MaFontList FCase
JButton sauvegarder
JButton exit

MaFontList étend JPanel, permet l'affichage d un sélecteur de font

JLabel Titre	JComboBox nom	JComboBox taille	JCheckBox type	JLabel Appercu



Controller

Les Contrôleur sont une partie importante du Taquin. Ils permettent l'interaction entre la vue et le model.

Elles héritent de AbstracAction classe permettant d assigner des événement a des composant graphique.

La Class MoveAction

Permet de faire un mouvement move à la case vide si c'est possible. Met a jour l'action, pour savoir si elle est possible. Enregistre l'événement au près du UndoManager.

La Class NiveauAction

Permet la sélection d un niveau

La Class PersoAction

Permet la sélection du niveau personnel, elle ne fait pas partie de Niveau action car ses paramètres sont différents a chaque fois.

La Class ExitAction

Permet de quitter l application en laissant au choix a l'utilisateur ou non de sauvegarder la partie en cours.

La Class NewAction

Permet de commencer une nouvelle partie.

La Class AvanceeAction

Permet de lancer la fenêtre d'option avancée. Rend le Taquin inaccessible a l'utilisateur.

La Class ScoreAction

Permet de lancer la fenêtre des scores. Rend le Taquin inaccessible a l'utilisateur.

La Class UndoAction

Permet de reculer d'un coup dans le déroulement de l'action. Enlève l'action du UndoManager.

La Class RedoAction:

Permet d avancer d'un coup dans le déroulement de l'action. Remet l action dans le UndoManager.

Dans Avancee

La Class Puit

Permet de sélectionner une couleur.

La Class SaveAction

Permet de sauvegarder les options.

La Class ExitAction

Permet de quitter les options et rend accessible le taquin.

La Class CheckAction

Permet de sélectionner le type de la police

Dans Score

La Class ExitAction

Permet de quitter table des scores et rend accessible le taquin.

Chapitre III:

Sauvegarde

La sauvegarde est utilisé ici pour sauvegarder les scores des différents joueurs, une partie s'il on doit quitter le taquin, les paramètres personnalisés du taquin(couleur des menu, des pièces, police du menu, des cases etc..)

Cette sauvegarde est effectué en XML, grâce au class XmlEncodeur et XmlDecodeur données par java.

Seulement cette class n encode que des objets de type java.beans. Les tableaux de String en faisant parties, les différents paramètres ont été enregistré sous forme de string dans des tableaux puis enregistré en xml dans leur fichiers respectifs.

La Class Fonction permet ainsi d'enregistrer et de charger les fichiers xml.

Conclusion:

Ce projet d'interface graphique fut l'occasion d' utiliser beaucoup de composants graphiques (JFrame, Jpanel,JButton,JLabel,JMenu,BorderLayout,GridLayout,JComboBox,JTabbedPane etc...) et de voir ainsi la mise en place de controllers et un model lié a ces composants.

Le principal intérêt de se projets fut donc d'utiliser au maximun ses composants pour permettre a l'utilisateur de paramètrer à sa guise **son** Taquin et que celui-ci soit visible et fonctionnel sous n'importe qu'elle machine utilisant la virtual machine de java.