TAQUIN GAME

Réaliser par :

Adnen Mrad

Le 28/05/2020

Projet Module :

Algorithme et Structure de données :

I. Résolution du Taquin :

Le taquin est un puzzle carre constitue de NxN cases et le but consiste à remettre dans l'ordre les carreaux dans l'ordre à partir d'une configuration initiale quelconque.

II. Implémentation du problème :

Nous avons implémenté ce problème dans le langage JAVA :

Package utilisés :

JAVA Swing : Swing propose de nombreux composants dont certains possèdent des fonctions étendues, une utilisation des mécanismes de gestion d'événements performants (ceux introduits par le JDK 1.1) et une apparence modifiable à la volée (une interface graphique qui emploie le style du système d'exploitation Windows ou Motif ou un nouveau style spécifique à Java nommé Metal).

JAVA IO : Permet l'échange des informations, que ce soit pour recevoir des données d'une source ou pour envoyer des données vers un destinataire.

La source et la destination de ces échanges peuvent être de natures multiples : un fichier, une socket réseau, un autre programme, etc....

De la même façon, la nature des données échangées peut être diverse : du texte, des images, du son, etc...

JAVA awt Event : Permet des gérer plusieurs évènements a partir des interfaces EventListener.

Les interfaces EventListener permettent de définir les traitements en réponse à des événements utilisateurs générés par un composant. Une classe doit contenir une interface auditrice pour chaque type d'événements à traiter :

- ActionListener : clic de souris ou enfoncement de la touche Enter
- ItemListener : utilisation d'une liste ou d'une case à cocher
- MouseMotionListener : événement de souris
- WindowListener : événement de fenêtre

Les classes du projet :

1. Chrono.java : Pour lancer un chronomètre qui détermine le temps pour résoudre le jeu.
2. ControlButton.java : Pour contrôler des évènements spécifiques (gagnant, couleur…).
3. ControlGroup.java : Création de la fenêtre d'ouverture.
4. ControlMenu.java : Contrôler toutes les fonctionnalités du Menu.
5. Fenetre.java : Mise en forme de la fenêtre du jeu (tableau, Menu…).
6. Model.java : Vérification du jeu (mouvement, le joueur gagne ou pas) et lire/écrire les score
7. Musique.java : Pour lancer et arrêter la musique.

Quelques fonctionnalités des classes utilisées :

1. **Classe Fenêtre :**
   On déclare les différents types d'attribut du Menu (MenuBar, Menu, MenuItem), les Jpanel/Jlabel qui vont nous permette de créer un panneau comme gestionnaire de placement

(chronomètre, bordure des petits carreaux, table) et les boite des dialogues avec JOptionPane (pour enregistrer les scores).

```
//--------------------------LES ATTRIBUTS ----------------------------------------

protected Model model;
public static Random random = new Random();

//Menu
protected JMenuBar barMenu;
protected JMenu menuOpen;
protected JMenuItem menuMelanger;
protected JMenu menuScores;
protected JMenuItem score3;
protected JMenuItem score4;
protected JMenuItem score5;
protected JMenu taille;
protected JMenuItem tailleTrois;
protected JMenuItem tailleQuatre;
protected JMenuItem tailleCinq;
protected JMenu menuAide ;
protected JMenuItem commentJouer ;
protected JMenuItem Apropos ;

//Affichage du chrono avec son JPanel
protected JLabel labelChrono;
protected JPanel panelChrono;
private Chrono chrono;

//JPanel vide pour pouvoir utiliser le borderLayout ( les bordures )
protected JPanel panelNord;
protected JPanel panelWest;
protected JPanel panelEast;
protected JPanel panelSouth;

protected JPanel panelTab;   //JPanel contenant la table

protected JPanel panelGeneral;   //JPanel general

protected JButton[][] cellTab;     //Table avec les JButton

protected JOptionPane dialogVictoire;    //Boite de dialog pour enregistrer son score lors de la victoire

protected JOptionPane dialogScore;       //Boite de dialog pour afficher les scores

private ControlMenu controlMenu;      //Accees au controleur

private int tailleTab;    //Taille du tableau
```

La création du constructeur Fenêtre nous permette d'avoir la forme de la fenêtre et effectuer quelques spécifications à notre jeu (le nom, la taille...) et c'est grâce à la bibliothèque Jframe.

```
//---------------------------------LE CONSTRUCTEUR --

public Fenetre(Model model, int tailleTab){

    this.model = model;

    initAttribut(tailleTab);
    creerWidnetVersionGeneral();
    creerMenu();
    setSize(600, 600);
    setTitle("Taquin");
    display();
    setResizable(false);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


}
```

Après avoir initialiser tous les attributs, on doit créer une table avec des petites cases avec des bordures numérotes selon sa taille et on doit aussi mettre en place le Menu qui à partir d'une méthode addActionListener va nous permette de choisir des options pour ce jeu.

```java
//---------------------------------CREATION DE LA TABLE----------------------
public void creationTable(){

    //taille de la table
    cellTab = new JButton[tailleTab][tailleTab];
    System.out.println("taille = " + cellTab.length);

    //initilaiser les cases par les nombres
    int nbr  = 0;

    try {

        //creation du tableau
        for (int i = 0; i < cellTab.length; i++){
            for(int j = 0; j < cellTab[i].length; j++) {
                String stringSurCase = Integer.toString(nbr);
                cellTab[i][j] = new JButton(stringSurCase);
                cellTab[i][j].setBackground(Color.getHSBColor(178, 81, 53));
                nbr++;
            }
        }

        //la 1ere cellule est vide
        cellTab[0][0] = new JButton();
        cellTab[0][0].setBackground(Color.getHSBColor(178, 81, 53));

    }
    catch (NullPointerException | IndexOutOfBoundsException e){
        System.out.println("Probleme");
    }

}
```

```java
//Creation du menu
public void creerMenu(){

    taille.add(tailleTrois);
    taille.add(tailleQuatre);
    taille.add(tailleCinq);

    menuScores.add(score3);
    menuScores.add(score4);
    menuScores.add(score5);

    menuOpen.add(taille);
    menuOpen.addSeparator();
    menuOpen.add(menuMelanger);
    menuOpen.addSeparator();
    menuOpen.add(menuScores);

    menuAide.add(commentJouer);
    menuAide.addSeparator();
    menuAide.add(Apropos) ;

    barMenu.add(menuOpen);
    barMenu.add(menuAide);
    setJMenuBar(barMenu);


}
```

```java
public void setControlMenu(ActionListener listener){
    tailleTrois.addActionListener(listener);
    tailleQuatre.addActionListener(listener);
    tailleCinq.addActionListener(listener);
    menuMelanger.addActionListener(listener);
    score3.addActionListener(listener);
    score4.addActionListener(listener);
    score5.addActionListener(listener);
    commentJouer.addActionListener(listener);
    Apropos.addActionListener(listener);

}
```

A l'aide d'un attribut Random on va mélanger les cases au début de chaque partie.

```java
//Melange les cases
public void melangerCase(){

    int k = 0, posX = 0, posY = 0;

    while (k < 100){

        int mouvAlea = random.nextInt(4) + 1;

        if (mouvAlea == 1){
            if (posX != 0){
                model.verifMouvement(cellTab, posX - 1, posY);
                posX = posX - 1;
            }
        }

        if (mouvAlea == 2){
            if (posY != cellTab.length - 1){
                model.verifMouvement(cellTab, posX, posY + 1);
                posY = posY + 1;
            }
        }

        if (mouvAlea == 3){
            if (posX != cellTab.length - 1){
                model.verifMouvement(cellTab, posX + 1, posY);
                posX = posX + 1;
            }
        }

        if (mouvAlea == 4){
            if (posY != 0){
                model.verifMouvement(cellTab, posX, posY - 1);
                posY = posY - 1;
            }
        }

        k++;

    }
```

Par la suite, quatre méthodes de création des boites de dialogues seront utilisées dans la classe ControlMenu.java.

```java
//Creation de la fenetre de dialog
public void creerBoiteDialogVictoire(){

    dialogVictoire = new JOptionPane();
    String nomJoueur = JOptionPane.showInputDialog(null, "Bravo vous avez fini, mettez votre nom pour enregistrer votre chrono.", "Victoire", JOptionPane.QUESTION_MESSAGE);
    model.setNomJoueur(nomJoueur);

}

//---------------------------AFFICHER LES SCORES-----------------------------------------------------

public void creerBoiteDialogScore(String categorie, String score, String fichier) throws IOException {

    model.rangerScore(fichier);
    dialogScore = new JOptionPane();
    JOptionPane.showMessageDialog(null, "Meilleur score : " + categorie + "\n\n" + score, "Meilleur score", JOptionPane.PLAIN_MESSAGE);

}

//---------------------------AFFICHER A PROPOS-----------------------------------------------------

public void creerBoiteDialogApropos(String string) {

    JOptionPane.showMessageDialog(null, string," A propos",JOptionPane.PLAIN_MESSAGE);

}

//---------------------------AFFICHER COMMENT JOUER-----------------------------------------------------

public void creerBoiteDialogCommentJouer(String string) {

    JOptionPane.showMessageDialog(null, string," Comment Jouer ? ",JOptionPane.PLAIN_MESSAGE);

}
```

2. Classe ControlMenu :

Une méthode actionPerformed est mise dans cette classe pour permette l'utilisateur de choisir les options de Menu (lire les scores, mélanger les petits carreaux pour commencer, savoir comment jouer, avoir une idée sur les meilleurs scores, choisir la taille…).

```
public void actionPerformed(ActionEvent e) throws IndexOutOfBoundsException{

    //Choisir la taille de la table


    if (e.getSource() == fenetre.tailleTrois){
        fenetre.undisplay();
        fenetre = new Fenetre(model, 3);
        fenetre.setControlMenu(this);
        fenetre.display();


    }
    if (e.getSource() == fenetre.tailleQuatre){

        fenetre.undisplay();
        fenetre = new Fenetre(model, 4);
        fenetre.setControlMenu(this);
        fenetre.display();


    }

    if (e.getSource() == fenetre.tailleCinq){
        fenetre.undisplay();
        fenetre = new Fenetre(model, 5);
        fenetre.setControlMenu(this);
        fenetre.display();

    }

    //Melanger

    if (e.getSource() == fenetre.menuMelanger) {
        fenetre.undisplay();
        controlButton = new ControlButton(model, fenetre);
        model.setEtatPartie(true);
        fenetre.melangerCase();
        fenetre.remetCouleur();
        fenetre.setLocationRelativeTo(null);
        fenetre.display();
        fenetre.menuMelanger.removeActionListener(this);
    }
```

```
        //Comment Jouer ?

        if (e.getSource() == fenetre.commentJouer) {
            fenetre.creerBoiteDialogCommentJouer("Le jeu taquin se compose d'un cadre de tuiles\n" +
                    "carrées numérotées dans un ordre aléatoire avec\n" +
                    "une tuile manquante. Pour résoudre le casse tête,\n" +
                    "les joueurs doivent placer les tuiles dans l'ordre\n" +
                    "en effectuant des mouvements de glissement qui\n" +
                    "utilisent l'espace vide. ");
        }

        // A propos
        if (e.getSource() == fenetre.Apropos) {
            fenetre.creerBoiteDialogApropos("Réaliser par : \n  Adnen Mrad ");
        }
```

Remarque : Pour le Menu Scores (prenons l'exemple de Item Menu Score 3) on va tout d'abords trier les scores et prendre le nom et le meilleur score des trois premiers joueurs, on crée une boite de dialogue qui va nous permette d'afficher nos résultats.

```
//Lire les scores

if (e.getSource() == fenetre.score3){

    try{
        model.rangerScore("score3.txt");

        String meilleurNom1, meilleurNom2, meilleurNom3;
        String meilleurScore1, meilleurScore2, meilleurScore3;
        if(!model.getMeilleurScore3().get(0).equals("")){
            meilleurNom1 = model.getMeilleurScore3().get(0)[0];
            meilleurScore1 = model.getMeilleurScore3().get(0)[1];
        }
        else{
            meilleurNom1 = "";
            meilleurScore1 = "";
        }
        if(!model.getMeilleurScore3().get(1).equals("")){
            meilleurNom2 = model.getMeilleurScore3().get(1)[0];
            meilleurScore2 = model.getMeilleurScore3().get(1)[1];
        }
        else{
            meilleurNom2 = "";
            meilleurScore2 = "";
        }
        if(!model.getMeilleurScore3().get(2).equals("")){
            meilleurNom3 = model.getMeilleurScore3().get(2)[0];
            meilleurScore3 = model.getMeilleurScore3().get(2)[1];
        }
        else{
            meilleurNom3 = "";
            meilleurScore3 = "";
        }

        fenetre.creerBoiteDialogScore("3x3", "1- " + meilleurNom1 + "=>" + meilleurScore1
                + "\n2- " + meilleurNom2 + "=>" + meilleurScore2
                + "\n3- " + meilleurNom3 + "=>" + meilleurScore3, "score3.txt");
    } catch (IOException e1) {
        e1.printStackTrace();
    }

}
```

3. Classe ControlGroup :

Cette classe a le rôle de créer la fenêtre d'ouverture :

```
public class ControlGroup {

    private Model model;
    private Fenetre fenetre;
    private ControlButton controlButton;
    private ControlMenu controlMenu;

//------------creation de la fenetre d'ouverture------------

    public ControlGroup(Model model) {

        this.model = model;

        fenetre = new Fenetre(model, 4);

        controlMenu = new ControlMenu(fenetre, model);

        fenetre.display();

    }

}
```

4. Classe ControlButton :

A partir de cette classe on va contrôler quelques évènements dans le jeu :

- A chaque fois l'utilisateur clique sur un carreau on vérifie tout d'abord la possibilité de faire le mouvement. Si c'est le cas on ferme cette fenêtre, on fait le réglage des couleurs et on initialise la fenêtre tout dépend du mouvement.

```
for(int i = 0; i < fenetre.cellTab.length; i++){
    for(int j = 0; j < fenetre.cellTab[i].length; j++) {
        if (e.getSource() == fenetre.cellTab[i][j]) {
            fenetre.undisplay();
            model.verifMouvement(fenetre.cellTab, i, j);
            fenetre.remetCouleur();
            fenetre.setLocationRelativeTo(null);
            fenetre.display();
        }
    }
}
```

- On lance le chronomètre selon l'état de la partie.

```
if (model.getEtatPartie()){
    fenetre.getChrono().start();
    model.setEtatPartie(false);
}
```

- Si le joueur a gagné le jeu une boite de dialogue s'ouvre pour permette le joueur d'écrire son nom et par la suite enregistrer son score selon la taille qui a choisi au début du jeu.

```
if (model.estGagne(fenetre.cellTab)){

    System.out.println("Victoire");
    fenetre.getChrono().stopChrono();
    fenetre.creerBoiteDialogVictoire();
    model.setStringChrono(fenetre.getChrono().toString());
    System.out.println(fenetre.getChrono().toString());
    model.afficheNomJoueur();


    for (int i = 0; i < fenetre.cellTab.length; i++){
        for (int j = 0; j < fenetre.cellTab[i].length; j++){
            fenetre.cellTab[i][j].removeActionListener(this);
        }
    }

    if(fenetre.cellTab.length == 3){
        try {
            model.ecrireScore("score3.txt");
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
    if(fenetre.cellTab.length == 4){
        try {
            model.ecrireScore("score4.txt");
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }
    if(fenetre.cellTab.length == 5){
        try {
            model.ecrireScore("score5.txt");
        } catch (IOException e1) {
            e1.printStackTrace();
        }
    }

}
```

5. Classe Chrono :

On va créer un objet chrono à partir de cette qui va nous permette de lancer/arrêter un chronomètre afin de savoir le temps nécessaire pour gagner le jeu et par la suite on peut savoir les meilleurs scores.

```
//----------------Start le jeu-------------------------
public void run(){

    startChrono = true;
    double time = 0;

    while(startChrono){

        try{
            sleep(10);
            time = time + 0.01;
            DecimalFormat df = new DecimalFormat("######0.00");
            strChrono = df.format(time);
            labelChrono.setText("Chronometre : " + strChrono);
        }
        catch (InterruptedException e){
           stopChrono();
        }

    }

}
```

```
//--------------------Stop le jeu-------------------
public void stopChrono(){

    interrupt();

}
```

6. Classe Musique :

A partir de cette classe et en important la bibliothèque javazoom.player on peut créer un objet de la classe Player qui implémente un lecteur simple pour la lecture d'un flux audio MPEG. Une méthode close est mise en évidence pour la fermeture de la musique

```
//-----------------Lancer la musique----------------------
@Override
public void run() {

    try {
        do {
            FileInputStream buff = new FileInputStream(fileLocation);
            player = new Player(buff);
            player.play();
        } while (loop);
    } catch (Exception ioe) {
        System.out.println(ioe.toString());
    }
}
```

```
//--------------Stop la musique-------------------------
 public void close(){
     loop = false;
     player.close();
     this.interrupt();
 }
```

7. Classe Model :

A Comme toutes les classes, on a besoin des accesseurs qui sont des méthodes qui nous permettre d'accéder aux variables de nos objets en lecture et des méthodes mutateurs nous permettra d'en faire en même en écriture.

```
//-----------------------------ACCESSEUR D'ECRITURE---------------------------------

public boolean getEtatPartie() {
    return etatPartie;
}

public ArrayList<String[]> getMeilleurScore3() {
    return meilleurScore3;
}

public ArrayList<String[]> getMeilleurScore4() {
    return meilleurScore4;
}

public ArrayList<String[]> getMeilleurScore5() {
    return meilleurScore5;
}

//---------------------------ACCESSEUR DE LECTURE-------------------------------------

public void setEtatPartie(boolean etatPartie){
    this.etatPartie = etatPartie;
}

public void setNomJoueur(String nomJoueur){ this.nomJoueur = nomJoueur;}

public void setStringChrono(String strChrono){
    score = strChrono;
}
```

Une méthode « verifeMouvement » qui va nous permette d'organiser les cases selon le mouvement effectuer par le joueur et par la suit la méthode « estGagne » qui va vérifier si le joueur a bien mis les carreaux en ordres ou pas encore (à l'aide d'un autre tableau.).

```java
public void verifMouvement(JButton[][] tabCase, int positionX, int positionY) {

    System.out.println("x = " + positionX + ", y = " + positionY);
    String temp;

    if (positionX != 0) {

        if (tabCase[positionX - 1][positionY].getText().equals("")){
            temp = tabCase[positionX - 1][positionY].getText();
            tabCase[positionX - 1][positionY].setText(tabCase[positionX][positionY].getText());
            tabCase[positionX][positionY].setText(temp);
            return;
        }

    }

    if (positionX != tabCase.length - 1) {

        if (tabCase[positionX + 1][positionY].getText().equals("")){
            temp = tabCase[positionX + 1][positionY].getText();
            tabCase[positionX + 1][positionY].setText(tabCase[positionX][positionY].getText());
            tabCase[positionX][positionY].setText(temp);
            return;
        }

    }

    if (positionY != tabCase.length - 1) {

        if (tabCase[positionX][positionY + 1].getText().equals("")){
            temp = tabCase[positionX][positionY +1].getText();
            tabCase[positionX][positionY + 1].setText(tabCase[positionX][positionY].getText());
            tabCase[positionX][positionY].setText(temp);
            return;
        }

    }

    if (positionY != 0) {

        if (tabCase[positionX][positionY  - 1].getText().equals("")){
            temp = tabCase[positionX][positionY - 1].getText();
            tabCase[positionX][positionY - 1].setText(tabCase[positionX][positionY].getText());
            tabCase[positionX][positionY].setText(temp);
            return;
        }

    }
```

```java
public boolean estGagne(JButton[][] tabCase){

    //creation d'un tableau pour veifier si les cases sont dans l'ordre
    String[][] tabString = new String[tabCase.length][tabCase.length];
    int nbr = 0;
    for (int i = 0; i < tabString.length; i++){
        for (int j = 0; j < tabString[i].length; j ++){
            tabString[i][j] = String.valueOf(nbr);
            nbr++;
        }
    }
    tabString[0][0] = "";

    //compteur des cases dans l'ordre
    int cptOrdre = 0;

    //verification des cases
    for (int i= 0; i < tabCase.length; i++){
        for (int j = 0; j < tabCase[i].length; j++){
            if (tabCase[i][j].getText().equals(tabString[i][j])){
                cptOrdre++;
                System.out.println("cptOrdre:"+cptOrdre+" nbrCase:"+tabCase.length*tabCase.length);
            }
            else{
                cptOrdre = 0;
                System.out.println("cptOrdre:"+cptOrdre+" nbrCase:"+tabCase.length*tabCase.length);
            }
        }
    }
    if (cptOrdre == tabCase.length*tabCase.length){
        System.out.println("Dans l'ordre");
        return true;
    }
    else{
        System.out.println("Pas dans l'ordre");
        return false;
    }

}
```

A la fin de cette classe on trouve trois méthodes responsable a la lecture, écriture, rangement du score dans des fichiers selon la taille de la fenêtre choisie : tout d'abord un crée un objet a partir de la classe File. Par la suite grâce à « BuffredWrite » et « BufferedRead » on peut écrire et lire les noms et les scores du joueurs à partir des fichiers crées.

```
    //Ecriture
    public void ecrireScore(String fichier) throws IOException {

        System.out.println("Fichier : "+fichier);
        File file = new File(fichier);
        System.out.println("File : "+file);
        System.out.println(score);

        BufferedWriter bw = new BufferedWriter(new FileWriter(file, true));

        bw.write(nomJoueur);
        bw.write("_");
        bw.write(score);
        bw.newLine();
        bw.close();

    }
//Lecture des scores dans les fichiers textes
public void lireScore(String fichier) throws IOException {

    System.out.println("Je lis:"+fichier);

    String s;

    File file = new File(fichier);

    BufferedReader br = new BufferedReader(new FileReader(file));

    if (fichier.equals("score3.txt")){
        while ((s = br.readLine()) != null) {
            lireScore3.add(s);
        }
    }

    if (fichier.equals("score4.txt")){
        while ((s = br.readLine()) != null) {
            lireScore4.add(s);
        }
    }

    if (fichier.equals("score5.txt")){
        while ((s = br.readLine()) != null) {
            lireScore5.add(s);
        }
    }

    br.close();

    for (int i = 0; i < lireScore3.size(); i++){
        System.out.println("Score3 ligne " + i + " : " + lireScore3.get(i));
    }

    for (int i = 0; i < lireScore4.size(); i++){
        System.out.println("Score4 ligne " + i + " : " + lireScore4.get(i));
    }

    for (int i = 0; i < lireScore5.size(); i++){
        System.out.println("Score5 ligne " + i + " : " + lireScore5.get(i));
    }

}
```

III. Résultats :