

Conception et développement d'une application de jeu de taquin

INTRODUCTION

Dans cette section, nous introduirons notre projet axé sur le développement d'une application de jeu de puzzle en Java, avec un accent particulier sur le célèbre jeu de taquin. Notre objectif principal est de créer une application interactive et divertissante qui offre une expérience de jeu engageante pour les utilisateurs .

À travers ce rapport, nous visons à fournir une vue d'ensemble complète du processus de conception et d'implémentation de notre application de jeu de puzzle. Nous détaillerons les principes de l'architecture logicielle MVC (Modèle-Vue-Contrôleur) utilisée dans notre application, en expliquant comment elle facilite la création d'une application modulaire et facile à maintenir.

Les lecteurs peuvent s'attendre à découvrir divers aspects de notre projet, notamment :

- La présentation du jeu de taquin et de ses mécanismes de jeu.
- Une analyse des objectifs et des exigences de notre application, mettant en évidence ses fonctionnalités clés telles que les niveaux de difficulté et la personnalisation de la grille.
- Une exploration de l'architecture logicielle MVC, expliquant comment elle facilite la gestion des données, la logique métier et l'interface utilisateur.
- Une revue du code source, mettant en évidence les choix de conception et les interactions entre les composants.
- Une explication de l'algorithme utilisé pour résoudre le puzzle.
- Une présentation de l'interface utilisateur, avec des captures d'écran commentées illustrant les fonctionnalités.
- Un examen des tests effectués pour valider le bon fonctionnement de l'application.

En résumé, ce rapport offre une vision complète de notre projet, combinant théorie, pratique et évaluation critique dans le contexte universitaire.

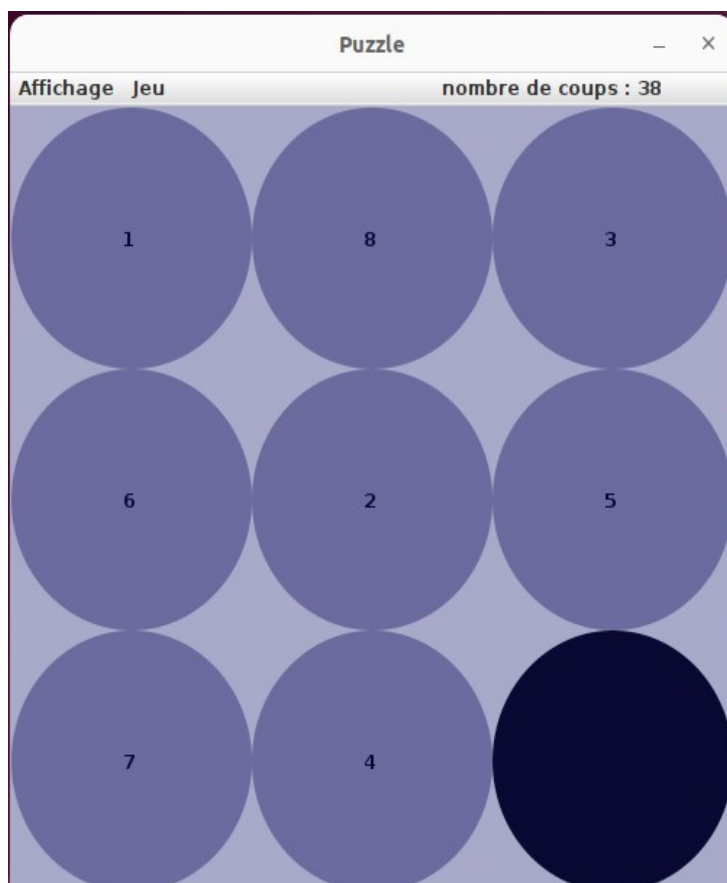
Présentation de notre Puzzle de Taquin

Le jeu de puzzle, également connu sous le nom de jeu de taquin, est un jeu classique de logique et de réflexion. Voici comment il fonctionne :

- Le plateau de jeu est composé d'une grille de cases, avec une case vide.
- Chaque case contient une partie de l'image à reconstituer, sauf la case vide.
- Le joueur déplace les pièces adjacentes à la case vide pour réorganiser les pièces.
- L'objectif est de reconstituer l'image en déplaçant les pièces dans le bon ordre.
- Le jeu peut proposer différents niveaux de difficulté avec des images plus complexes ou des contraintes supplémentaires.

En résumé, le jeu de puzzle est un défi de logique où le joueur doit réorganiser les pièces pour former l'image originale, offrant ainsi une expérience engageante pour les joueurs de tous âges.

Dans notre barre de menu principale, nous avons deux menus principaux : "Affichage" et "Jeu". Dans le menu "Affichage", les utilisateurs ont la possibilité de choisir entre deux options : "Chiffres" et "Image". L'option "Chiffres" permet de basculer l'affichage vers un puzzle de chiffres, tandis que l'option "Image" permet de basculer l'affichage vers un puzzle d'image.



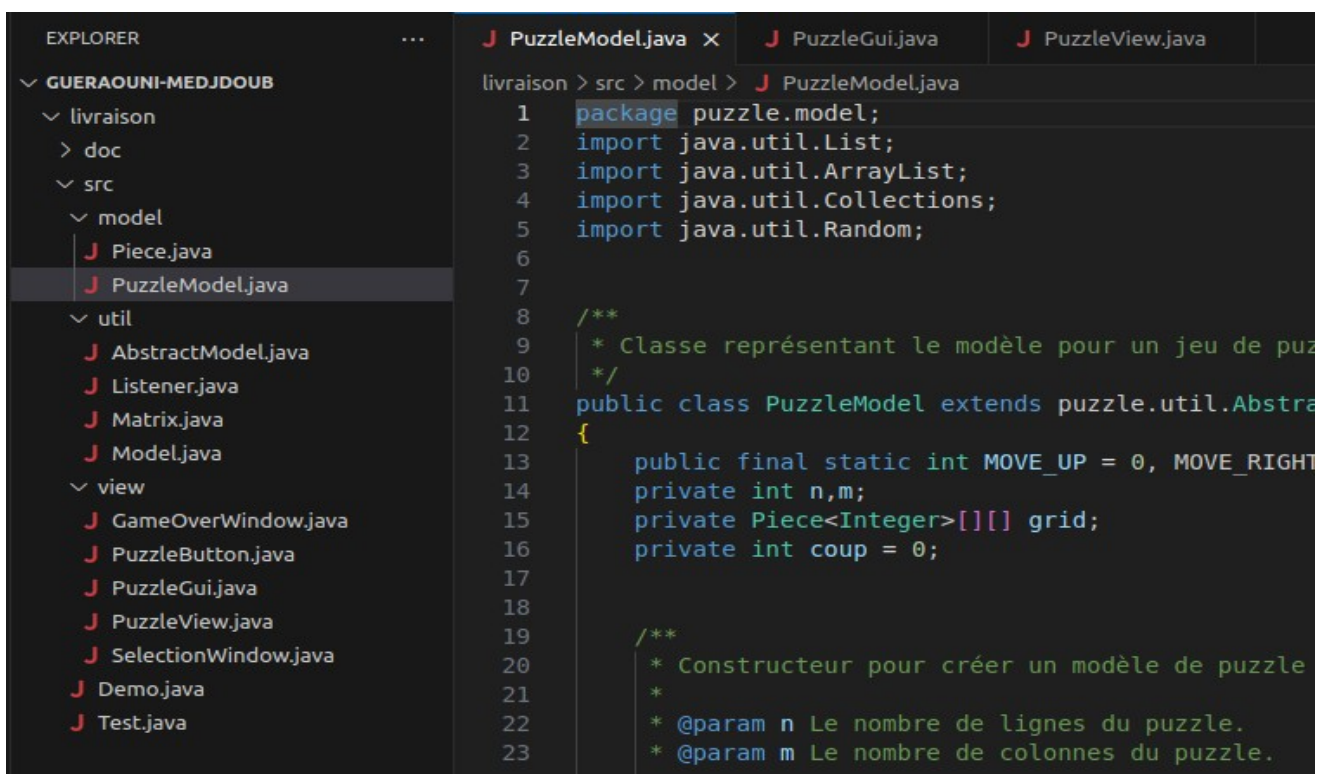
Architecture logicielle et Interface utilisateur

L'architecture Modèle-Vue-Contrôleur (MVC) divise une application en trois parties (comme vu en cour) :

- **Modèle (Model)** : Gère les données et la logique métier.
- **Vue (View)** : Présente les données à l'utilisateur et reçoit ses interactions.
- **Contrôleur (Controller)** : Fait le lien entre le modèle et la vue, gérant les actions de l'utilisateur et mettant à jour le modèle et la vue en conséquence.

Maintenant, regardons comment cela s'applique à notre application de jeu de puzzle :

- **Modèle (Model)** : Dans ton code, la classe PuzzleModel représente le modèle de notre jeu. Elle gère la logique métier du jeu, comme la manipulation des pièces du puzzle, les mouvements du joueur et la vérification des conditions de victoire.
- **Vue (View)** : La classe PuzzleView agit comme la vue de notre application. Elle est responsable de l'affichage du plateau de jeu, des pièces du puzzle et des éléments d'interface utilisateur tels que les boutons et les menus.
- **Contrôleur (Controller)** : Les classes PuzzleGui et PuzzleButton jouent le rôle de contrôleurs dans notre application. PuzzleGui gère les actions de l'utilisateur, comme le choix des niveaux de difficulté et le redémarrage du jeu ainsi que le switch entre l'affichage chiffre ou images, tandis que PuzzleButton gère les interactions avec les pièces individuelles du puzzle, comme les clics pour les déplacer.



The screenshot shows an IDE with a project named 'GUERAOUNI-MEDJDOUB'. The 'src' directory contains 'model' and 'util' subdirectories. The 'model' directory contains 'PuzzleModel.java' and 'PuzzleGui.java'. The 'util' directory contains 'AbstractModel.java', 'Listener.java', 'Matrix.java', and 'Model.java'. The 'view' directory contains 'GameOverWindow.java', 'PuzzleButton.java', 'PuzzleGui.java', 'PuzzleView.java', 'SelectionWindow.java', 'Demo.java', and 'Test.java'. The 'PuzzleModel.java' file is open, showing the following code:

```
1 package puzzle.model;
2 import java.util.List;
3 import java.util.ArrayList;
4 import java.util.Collections;
5 import java.util.Random;
6
7
8 /**
9  * Classe représentant le modèle pour un jeu de puzzle
10  */
11 public class PuzzleModel extends puzzle.util.AbstractModel {
12     {
13         public final static int MOVE_UP = 0, MOVE_RIGHT = 1, MOVE_LEFT = 2, MOVE_DOWN = 3;
14         private int n,m;
15         private Piece<Integer>[][] grid;
16         private int coup = 0;
17
18
19         /**
20          * Constructeur pour créer un modèle de puzzle
21          *
22          * @param n Le nombre de lignes du puzzle.
23          * @param m Le nombre de colonnes du puzzle.
```

En résumé, l'architecture MVC permet de séparer clairement les responsabilités dans notre application de jeu de puzzle, ce qui la rend plus modulaire, plus facile à maintenir et à étendre.

l'interface utilisateur de notre application de jeu de puzzle a été conçue pour offrir une expérience interactive et engageante aux utilisateurs. Voici une explication plus détaillée de chaque composant de l'interface, basée sur le code :

1. **PuzzleGui :**

- `buildMenuBar()`: Cette méthode crée et retourne la barre de menus de l'interface utilisateur. Elle contient des éléments tels que les options d'affichage, les options de jeu et l'affichage du nombre de coups.
- `newModel()`: Cette méthode est appelée pour créer un nouveau modèle de puzzle en fonction du niveau de difficulté sélectionné par l'utilisateur.
- `changeModel()`: Cette méthode met à jour le modèle de puzzle de la vue avec un nouveau modèle. Elle est utilisée pour réinitialiser le jeu avec un nouveau modèle lorsque nécessaire.
- `changeLevel(int newLevel)`: Cette méthode met à jour le niveau de difficulté du jeu en fonction de la sélection de l'utilisateur.
- `changeCoups(int nombre)`: Cette méthode met à jour l'affichage du nombre de coups dans la barre de menus en fonction du nombre de coups effectués par le joueur pendant le jeu.

2. **SelectionWindow :**

- `actionPerformed(ActionEvent e)`: Cette méthode est appelée lorsque l'utilisateur clique sur le bouton de validation pour personnaliser la taille de la grille. Elle récupère les valeurs des champs de texte pour les lignes et les colonnes, les valide et met à jour la taille de la grille de jeu en conséquence.

3. **PuzzleView :**

- `modelUpdated(Object source)`: Cette méthode est appelée chaque fois que le modèle est mis à jour. Elle est responsable de mettre à jour l'affichage en fonction des modifications apportées au modèle, en rafraîchissant le contenu de la grille du puzzle.

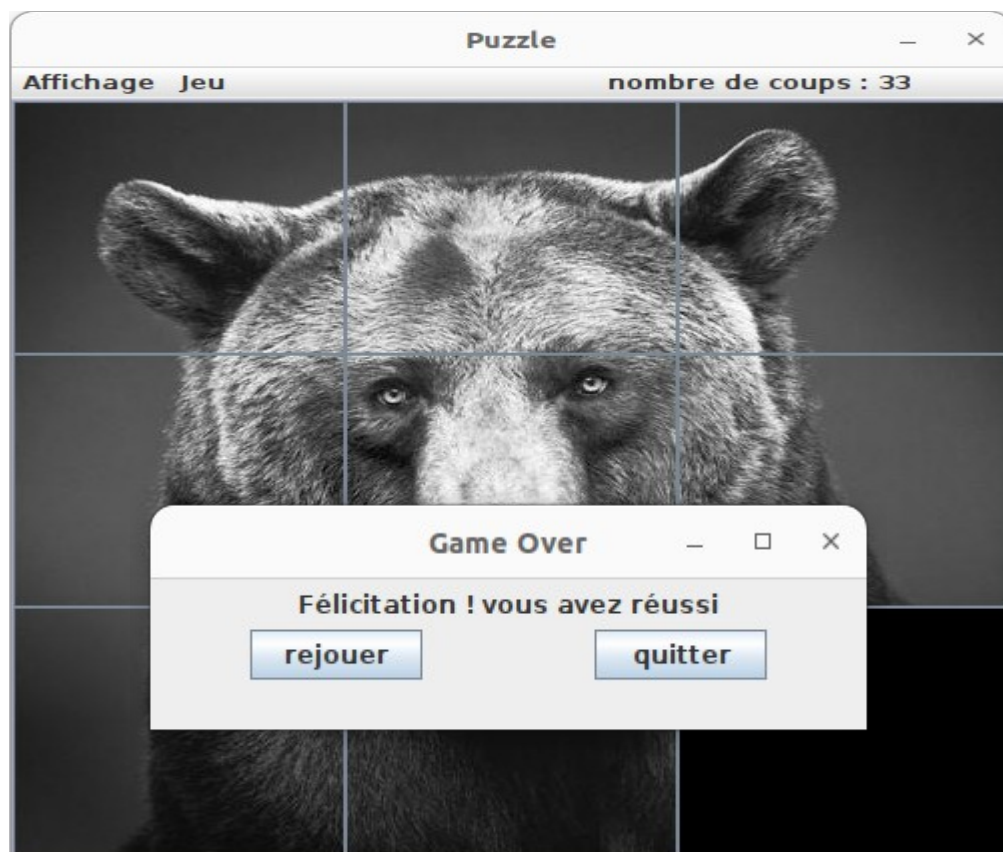
- `paintComponent(Graphics g)`: Cette méthode est responsable du rendu de la vue du puzzle. Elle dessine les pièces du puzzle et les éléments d'interface utilisateur tels que les boutons.

4. **PuzzleButton :**

- `mouseEntered(MouseEvent e)`: Cette méthode est appelée lorsque la souris entre dans la zone du bouton. Elle déclenche l'état de survol du bouton, ce qui peut entraîner un changement visuel si le bouton est conçu pour répondre au survol de la souris.
- `mouseExited(MouseEvent e)`: Cette méthode est appelée lorsque la souris quitte la zone du bouton. Elle désactive l'état de survol du bouton, rétablissant son apparence par défaut.
- Ces méthodes jouent un rôle essentiel dans le fonctionnement global de l'interface utilisateur, assurant une interaction fluide et intuitive avec l'application de jeu de puzzle

5. **GameOverWindow :**

nous avons intégré une fonctionnalité essentielle consistant en l'ouverture d'une nouvelle fenêtre pop-up qui s'affiche lorsque le joueur termine avec succès le jeu de puzzle. Elle annonce la fin du jeu et permet au joueur de choisir entre rejouer ou quitter l'application



La fenêtre est construite avec deux boutons, l'un pour rejouer et l'autre pour quitter le jeu. Lorsque le joueur clique sur l'un des boutons, un événement est déclenché, et la fenêtre réagit en conséquence.

6. *Personnalisation :*

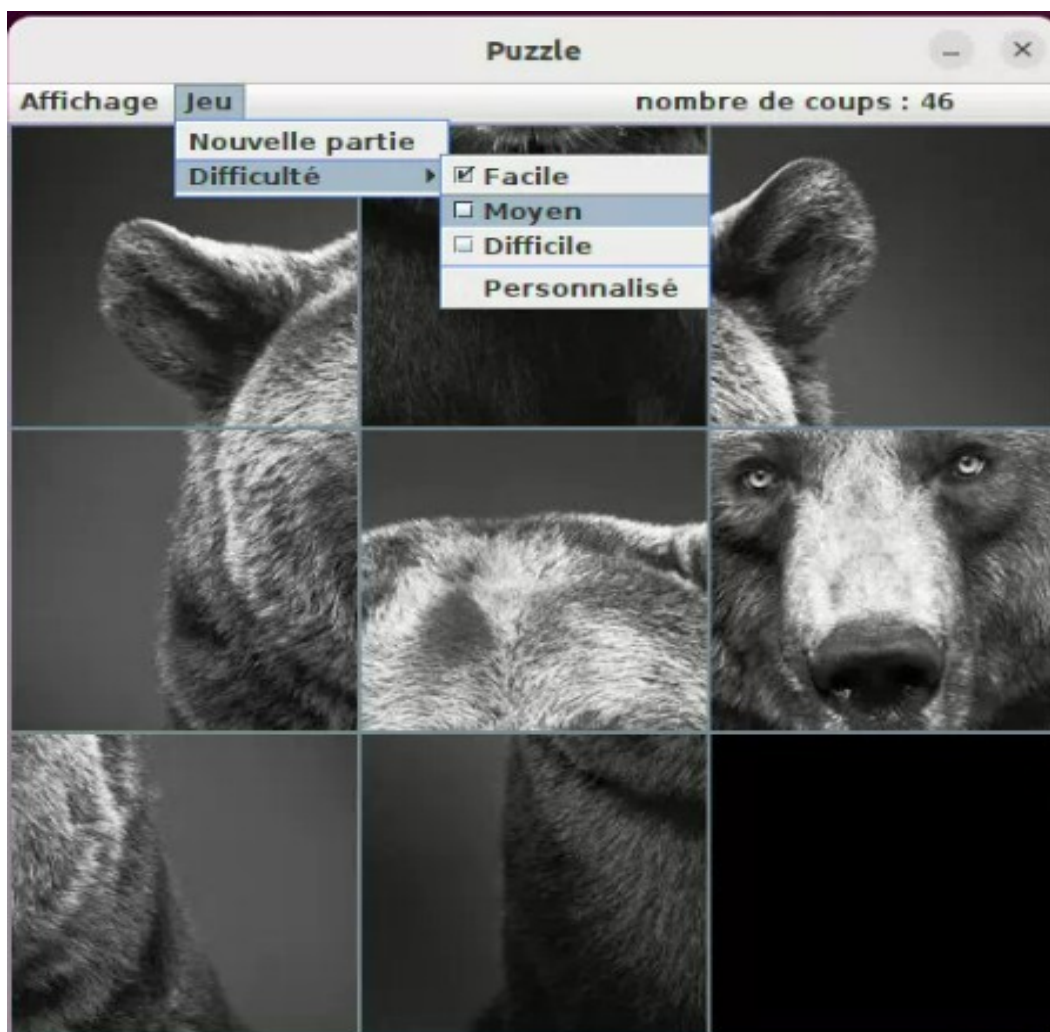
nous avons également intégré une fonctionnalité permettant à l'utilisateur de personnaliser la taille de la grille de jeu, avec une dimension minimale de 3x3 et maximale de 10x10. Cette fonctionnalité offre une flexibilité supplémentaire à l'utilisateur, lui permettant de choisir la taille qui convient le mieux à ses préférences de jeu.



7. Niveaux de difficultés :

Dans le menu "Jeu", les utilisateurs ont accès à deux fonctionnalités principales. Tout d'abord, l'option "Nouvelle partie" leur permet de lancer une nouvelle partie du jeu. Ensuite, ils peuvent également choisir la difficulté du jeu en sélectionnant parmi les options prédéfinies : "Facile", "Moyen" et "Difficile". De plus, ils ont la possibilité de personnaliser la taille de la grille en choisissant l'option "Personnalisé", ce qui leur permet de spécifier le nombre de lignes et de colonnes pour créer un puzzle sur mesure.

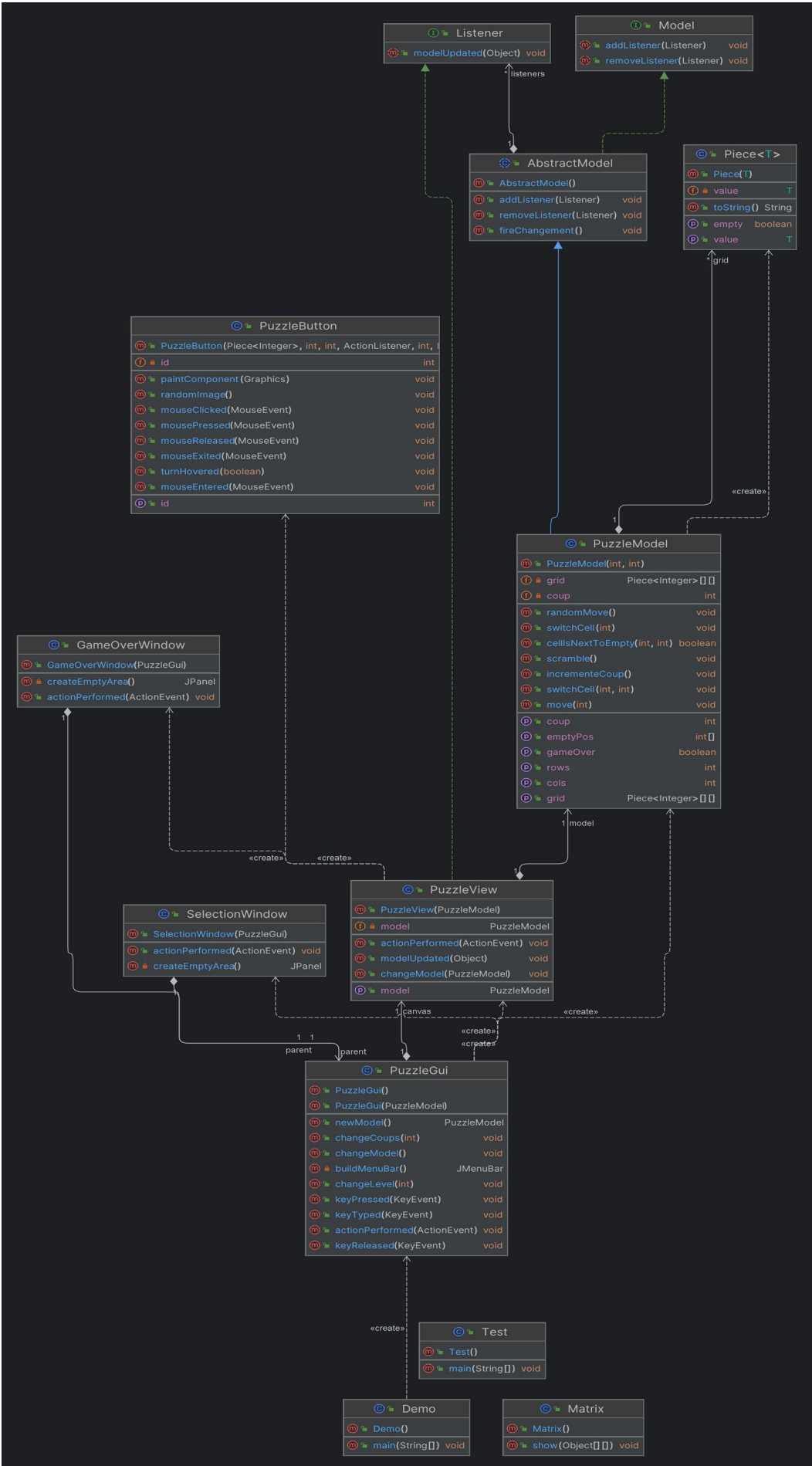
Cette organisation de la barre de menu offre aux utilisateurs un accès facile aux différentes fonctionnalités du jeu et leur permet de personnaliser leur expérience de jeu selon leurs préférences.



Principe de l'algorithme

1. **Sélection des mouvements possibles** : L'algorithme commence par identifier les mouvements possibles à partir de la position actuelle du puzzle. Dans la méthode `randomMove()` de la classe `PuzzleModel`, une liste des cellules adjacentes à la case vide est générée. Cela est réalisé en parcourant les cellules adjacentes et en les ajoutant à la liste si elles se trouvent à l'intérieur des limites de la grille.
2. **Choix aléatoire d'un mouvement** : Une fois que la liste des mouvements possibles est établie, l'algorithme choisit aléatoirement l'une des cellules adjacentes pour effectuer le mouvement. Cela est réalisé en utilisant un générateur de nombres aléatoires pour sélectionner un index dans la liste des cellules adjacentes.
3. **Exécution du mouvement** : Une fois le mouvement sélectionné, l'algorithme met à jour la position du puzzle en échangeant la case vide avec la case sélectionnée. Cela est réalisé dans la méthode `switchCell()` de la classe `PuzzleModel`, qui échange les positions des cases dans la grille.
4. **Répétition du processus** : Après avoir effectué le mouvement, l'algorithme peut répéter le processus en sélectionnant de nouveaux mouvements à partir de la nouvelle configuration du puzzle. La méthode `scramble()` de la classe `PuzzleModel` utilise cette approche pour mélanger le puzzle en effectuant un certain nombre de mouvements aléatoires.

DIAGRAMME DES CLASSES



CONCLUSION

Dans cette conclusion, en tant qu'étudiants en informatique, nous avons réalisé un projet de développement d'une application de jeu de puzzle en Java. Ce projet nous a permis d'appliquer nos connaissances théoriques à un projet pratique.

Nous sommes fiers d'avoir réussi à concevoir une application fonctionnelle et attrayante, malgré les défis rencontrés. La mise en œuvre du modèle MVC a été un succès, rendant notre code clair et modulaire.

Notre application offre une expérience utilisateur immersive avec plusieurs modes de jeu et niveaux de difficulté. Cependant, des améliorations restent possibles, notamment en termes de performances et de fonctionnalités supplémentaires.

En résumé, ce projet nous a permis de développer nos compétences en développement logiciel et nous sommes reconnaissants envers nos enseignants pour leur soutien. Nous sommes impatients de poursuivre notre exploration dans le domaine de l'informatique.