Reda BOUZIDI
Chaïma JAIDANE

Projet POO

1. Introduction

Java est un langage de programmation orienté objet qui s'appuie sur des structures de données appelées *objet*, et des mécanismes tels que l'encapsulation, l'héritage ou encore le polymorphisme

C'est pourquoi dans le cadre de notre projet nous avons essayé d'illustrer au mieux les principes propres à la programmation orienté objet et à Java plus particulièrement.

2. Kakuro

Kakuro est un jeu du même type que « nombres fléchés ». Le jeu comporte une grille avec des cases remplies non modifiable (case en noir), des cases avec des nombres, des cases vides à remplir avec des nombres compris entre 1 et 9 et pour finir des cases d'indication pouvant contenir deux valeurs, la somme des cases contiguës de la colonne sous la case d'indication et/ou la somme des cases contiguës de la ligne à droite de la case d'indication.

La seule contrainte du jeu est que les colonnes ou lignes contiguës ne peuvent contenir qu'une seule fois chaque valeur de 1 à 9.

3. Structures des programmes

Interfaces

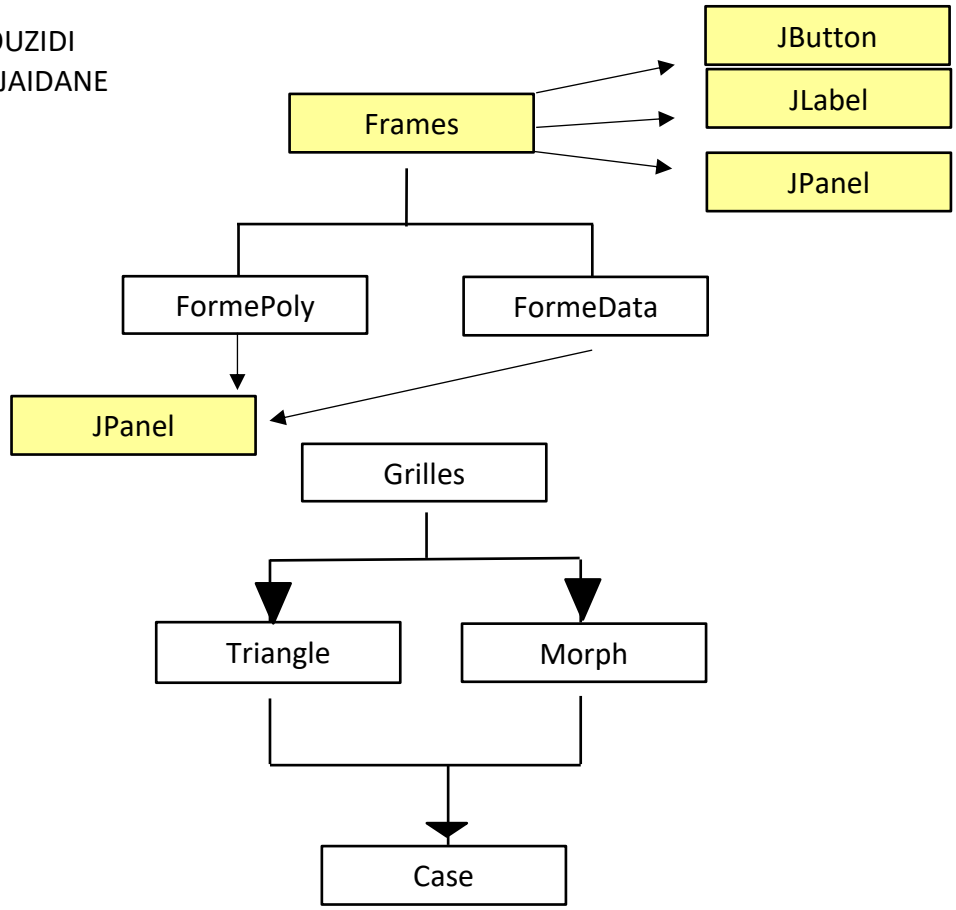
L'interface crée est Frames.
Les interfaces issues des API sont JFrame, JPanel.

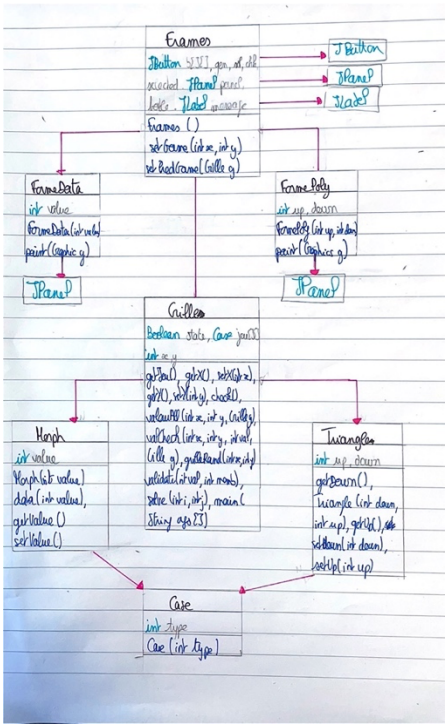
Classes

Les classes sont FormeData et FormePoly qui implemente l'interface Frames.
Nous avons aussi les classes Morph,Triangle,Case qui implémente la classe Grilles.

Diagramme de classe/ UML

Reda BOUZIDI
Chaïma JAIDANE





Reda BOUZIDI
Chaïma JAIDANE

Nous avons décidé de modéliser notre code de la sorte afin de faciliter la compréhension du code. En effet, nous avons créé par exemple pour l'interface graphique deux classes, qui gèrent chacune un type de case rencontré dans le jeu. Pour ensuite que ces dernières aident à l'implémentation de la classe Frames, qui génèrent l'ensemble du jeu soit les cases à compléter par le jouer, les boutons pour manipuler le jeu ainsi que la modification de la taille de la grille.
De même pour la classe Grille, nous avons créé 3 sous classes pour faciliter l'implémentation de Grille. La classe Case permet d'associer un type à une case, donc les classes Triangles et Morph sont créés pour respectivement associer une valeur au case indicatives et ***aux cases avec des nombres.*** La décomposition est faite de la sorte pour faciliter la gestion des différents types de cases.

4. Description des classes et de l'interface

Dans cette partie, nous traiteront les éléments du programme que nous avons jugé nécessaire de détailler en plus de leurs commentaires.

Tout ce qui n'apparaît pas dans cette partie est simplement expliqué par les commentaires.

Grille :

La classe grille étant la classe principale, gérant le jeu de façon qu'aucune erreur ne puisse se faire. Pour cela nous avons décidé d'implémenter plusieurs fonctions qui sont les suivantes :

- **Checker :**

La fonction checker permet de voir si les conditions de victoires sont remplies et sinon retourne en fonction de l'erreur retrouvé un entier :

  - 0 pour une victoire
  - 1 en cas de non-respect de la règle de duplication
  - 2 en cas d'invalidité dans la somme
  - 3 en cas de cases incomplètes

L'algorithme cherche chaque case correspondante a Triangle et en fonctions des valeurs de la case, elle cherche horizontalement à droite ou verticalement en bas les cases Morph et vérifie si elles correspondent aux contraintes.

- **Copie :**

Copie intégralement la Grille donné en paramètre tout en mettant les valeurs modifiables a 0

- **valeurAll :**

Demande en entrée des coordonnées (correspondantes a un Morph) et teste pour la case en questions toutes les valeurs de 1 à 9, puis sélectionne celle qui peuvent être posé sans causer une duplication de valeurs.

- **grilleRand(Première fonctionnalité ajouté) :**

La fonction statique grilleRand permet de retourner une grille de taille x*y avec des valeurs au hasard. L'algorithme en premier lieu, insère dans les premières cases/2éme cases (pour soucis de grandes cases) des Triangles. Par la suite, à partir de ces dernières elle insère d'autres

Triangles dans des cases plus loin au hasard. Si deux triangles se croisent, elle crée un double triangle demandant des valeurs a droit et en bas.
En revanche, si un triangle demande une valeur à droite (*resp en bas*) et qu'il trouve un autre triangle à droite (*resp en bas*) il supprime le triangle.
À la suite des triangles placés, la fonction place des Morph dans les cases correspondantes jusqu'à trouver un triangle ou arriver à la taille maximale de la grille.
Cependant, grâce au random la fonction va générer des cases de type Data. Les chances sont néanmoins assez basses.

FormePoly :
Classes représentant un triangle dans l'interface graphique, elle consiste à utiliser la fonction **public void paint(Graphics g)** pour dessiner à partir du JPanel lui-même la case , ainsi que ses valeurs

FormeData :
Classes représentant un Morph de type data dans l'interface graphique, elle consiste à utiliser la fonction **public void paint(Graphics g)** pour dessiner à partir du JPanel lui-même la case , ainsi que sa valeur
Frames :
La classe frame représente l'interface graphique en elle-même, elle utilise toutes les autres classes pour faire fonctionner le jeu en harmonie
SetGame :
Définie un jeu de taille x*y de façon random, elle utilise la fonction setPredGame et grilleRand pour créer le niveau et l'interface.
setPredGame :
Initialiseur de l'interface graphique, elle crée un petit panel de 100 pixel de hauteur, puis crée une grille de taile x*y , prenant une taille proportionnelle au nombre de cases arrivant jusqu'à presque la taille de l'écran.
Ensuite elle crée une grille clone pour pouvoir garder les anciennes valeurs de la grille initial.
Chaque bouton a un eventListener correspondant à la variable gen correspond à l'eventListener générant une nouvelle grillle random. La variable sol correspond à l'eventListener montrant la solution, chk correspond à l'eventListener qui regarde si la solution est bonne et hard correspond à l'eventListener qui augmente la difficulté (**deuxième fonctionnalité ajouté**) ensuite, les boutons b[i][j] ont tous un event listener unique , permettant de changer la valeur de l'élément correspondant à la grille et au bouton(si le mode difficile n'est pas activé)

5. La répartition du travail

Afin de répartir de manière équivalente le travail, nous avons décidé que Reda BOUZIDI allait se charger de la partie algorithmique, c'est-à-dire des classes Grilles, Triangles, Morph et Case. La partie interface a donc été faite par Chaïma JAIDANE. Soit les classes Frames, FormeData et FormePoly.