Un Projet Similaire au Pet Rescue Saga

Étudiant 1: YE Junli

Étudiant 2: LI Songqiao

Université de Paris
Double Licence Mathématiques-Informatique Année 2020/2021

January 12, 2021

Table des manières

Introduction

Le projet PRS est le projet semestriel du cours POOIG de L2 Informatique de l'Université de Paris. Ce projet est basé sur le langage java et son but principale est de réaliser un jeu comme le jeu très connu Pet Rescue Saga de King.com.

Figure: L'interface du jeu "Pet Rescue Saga" de King.com

Table des manières

Objectifs de conception et fonctions accomplies

Concevez un mini-jeu similaire à la saga classique du sauvetage d'animaux domestiques où l'utilisateur interagit avec le jeu par le biais d'une interface graphique. Dans la fenêtre, l'interface d'initialisation comporte deux types de carrés : les carrés normaux et les carrés animaux. L'utilisateur peut éliminer les carrés normaux en cliquant dessus pour ramener tous les animaux "au sol", c'est-à-dire au bas du formulaire.

En plus de l'interface graphique, une interface textuelle est nécessaire, où l'utilisateur peut terminer le jeu dans le terminal en utilisant des instructions de commande et des entrées au clavier.

Organisation générale et répartition des tâches

	Junli	Songqiao
Travail de l'analyse	Analyse des besoins du projet Mise en place d'une structure globale Version textuelle du jeux Achever la conception détaillée des packages ENTITY et CONTROLLER	Conceptualiser le processus d'interaction avec l'utilisateur Compléter la conception détaillée du package VIEW
Travail du code	package entity package controller Codes LaTex pour générer le rapport Codes UML pour les graphes et le fichier README.md	pakage View Codes LaTex pour générer le rapport Codes UML pour les graphes et le fichier
Travail du text	Conception et rédaction du rapport Fichier README.md dans le projet	La partie UI du rapport Diaposition pour la soutenance (S'il y en a besoin)

Table: Tableau de réparation des tâches

Outils utilisés et introduction

Java

Markdown

UML

textbfLaTeX

Table des manières

Conception générale du programme: Class Diagrame

La conception globale du projet utilise une structure MVC qui permet de faire fonctionner les choses grâce à trois "package" distincts mais inter-connectés.

Le package entity

Le package view

Le package controller

Les objets du programme

Structure de l'héritage: Généricité et classe abstraite

```
package entity;

public abstract class Block<T>{
    // Code
}
```

```
package entity;

public class NormalBlock extends Block<Color>{
    // Code
}
```

```
package entity;

public class AnimalBlock extends Block<Animal>{
    // Codes
}
```

Les objets du programme

On manipule égalememt un BlockFactory, i.e. Usine des blocs où on peut y produre un block aléatoirement lorsqu'on commencer à jouer.

Les objets du programme

La classe Board est l'élément le plus essentiel du package entity.

```
static final int WIDTH = 10;
static final int HEIGHT = 10;
// Les deux chiffres decrivent la dimension de tableau
private static BlockFactory factory = new BlockFactory();
private Block[][] blocks;
```

Les objets du programme

Color et Animal: Utilisation de l'Enum

Pour décrire les couleur et les animaux, on a utilisé les classes Enum qui n'a pas fait partie du cours de ce semestre en POO. Une énumération est en fait une classe, d'où cette appellation de classe énumération. Cette classe étend la classe Enum, qui elle-même étend la classe Object, comme toutes les classes en Java.

Les valeurs d'une énumération sont les seules instances possibles de cette classe. Dans notre projet, par exemple, Color comporte cinq instances: B, R, Y, G, O. On peut donc comparer ces instances à l'aide d'un == de façon sûre, même si la comparaison à l'aide de la méthode equals() reste possible. Ainsi on a aussi override la méthode equals() dans la classe Board.

La logique du jeux en base

Table: Les méthodes dans la classe Game et ses fonctions

Méthode	Valeur retourne	Fonctions réalisées
Game( )		Constructeur
print( )	void	afficher le tableau, pour la version textuelle
eliminate( )	void	méthode principale du programme 1. calculons le domaine d'elimination 2. eliminer les blocks 3. réduire la place après avoir terminé l'élimination
rangeOfElimination( )	boolean[][]	retourner un tableau de boolean de deux dimension selon le règle d'elimination
fall( )	void	faire descendre les block pour complir les blocks null
deleteNullBlock( )	Block[]	supprimer les blocks null
isWin( )	boolean	Déterminer si le jeu est gagné

La logique du jeux en base

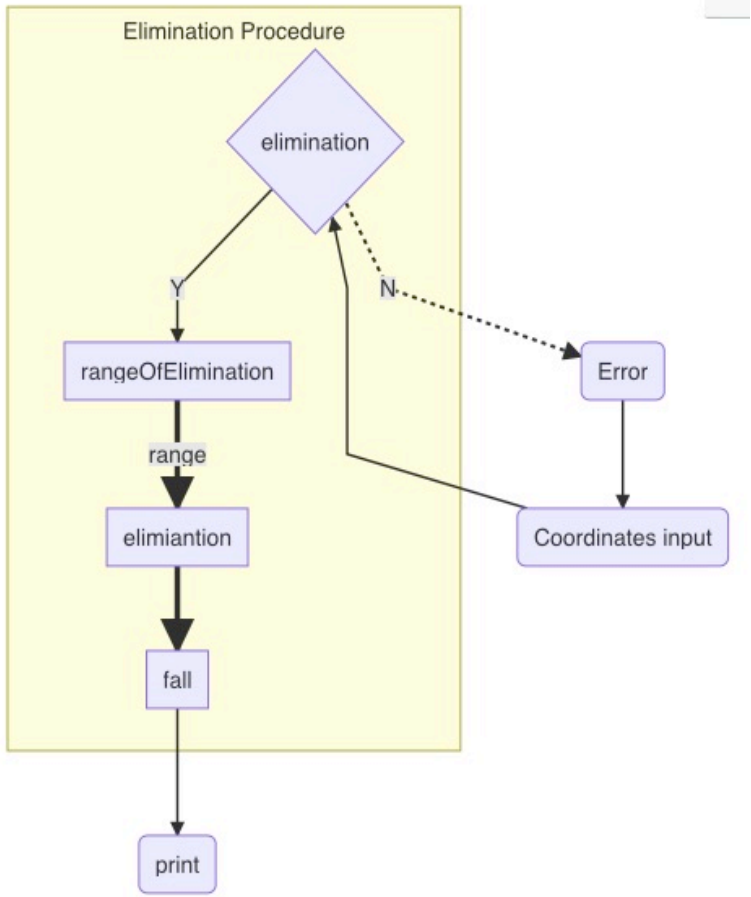


Figure: Logique d'Elimination

Version texte du jeux

Nous implémentons la version texte du jeu en définissant une méthode playInText() dans la classe de Play. L'utilisateur peut alors simplement lancer le jeux à partir de leur Terminal et intéragir avec le machine grâce à son clavier.

Le déroulement de base de la version texte du jeu est le suivant.

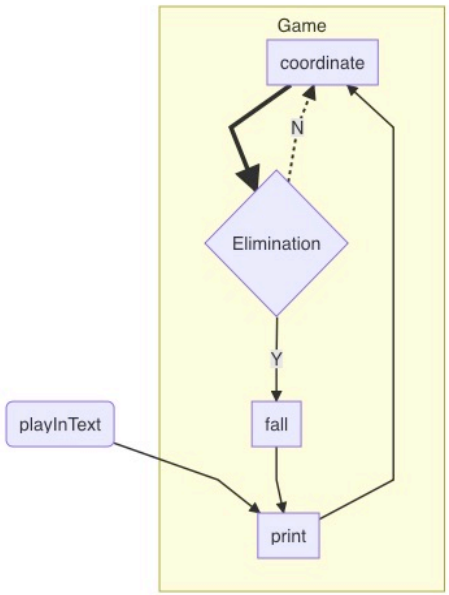


Figure: PlayInText

Interface utilisateur

Dans le projet, on réalise l'interface graphique à l'aide de Swing et AWT. Ce package contient alors deux class, même on a bien y implémenté les classes internes pour bien représenter leurs relations. Le joueur peux s'intéragir avec la machine depuis son souris.

Figure: L'interface principal

Interface utilisateur

Difficult:

Figure: Sous-fenêtre "Difficult"

Help:

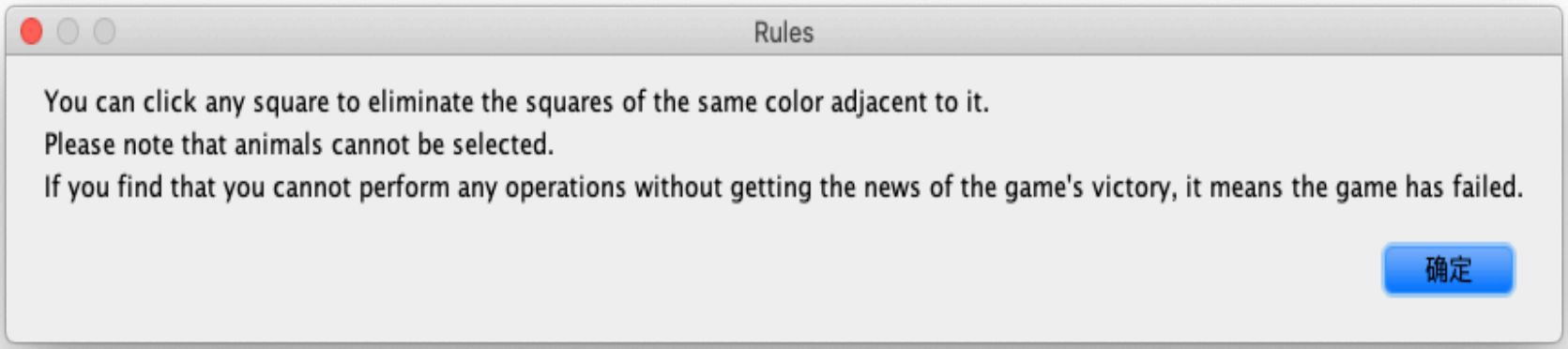


Figure: Sous-fenêtre "Help"

Interface utilisateur

Restart:

Comme le jeux ne s'arrête pas même il n'y pas plus de block à éliminer, le joueur, peut, simplement de recommencer le jeux dans n'importe quel moment au cours du jeux. i.e. , on va créer un nouveau Game et l'afficher.

Robot:

Dans la version GUI du jeu, nous avons également une fonction spéciale : le robot, où l'utilisateur clique sur le bouton "robot" et un robot effectuera une étape à la place de l'utilisateur.

Les quatres boutons dessus sont réalisés par expressions lambda comme dessous:

```
jb0.addActionListener((ActionEvent e) -> new DimensionChoose());
jb1.addActionListener((ActionEvent e) -> new RulesPanel());
jb2.addActionListener((ActionEvent e) -> restart());
jb3.addActionListener((ActionEvent e) -> robotPlay());
```

Interface utilisateur

En plus, pour les deux sous-fênetres simlpes, on les a réalisé par les classes internes qui rend les relations plus caires.

```
public class View extends JFrame {
    static class RulesPanel extends JPanel {
        public RulesPanel() {
            // Code
        }
    }

    class DimensionChoose extends JPanel {
        public DimensionChoose() {
        // Code
        }
    }
}
```

Interface utilisateur

Thread-Safe:

```
EventQueue.invokeLater(() -> {
    try{
        view.setVisible(true);
    }catch (Exception e){
        e.printStackTrace();
    }
});
```

Pré-requis

```
Enzo-deMacBook-Air:PRS-master mac$ pwd
/Users/mac/PRS/PRS-master
Enzo-deMacBook-Air:PRS-master mac$ ls
PRS.iml         README.md       img             src
```

Figure: Le fichier PRS.zip dans le console

```
Enzo-deMacBook-Air:src mac$ java controller/Play
Hello! Welcome to PRS game, I hope you can have a nice day!
Bonjour ! Bienvenue à PRS Games et je vous souhaite une excellente journée !
您好！ 欢迎来到PRS游戏， 希望您可以度过愉快的一天！
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
Please enter 1 for English and press enter;
Pour le français, tapez 2 et appuyez sur enter;
中文请输入3并按回车
```

Figure: Compiler et exécuter le jeux

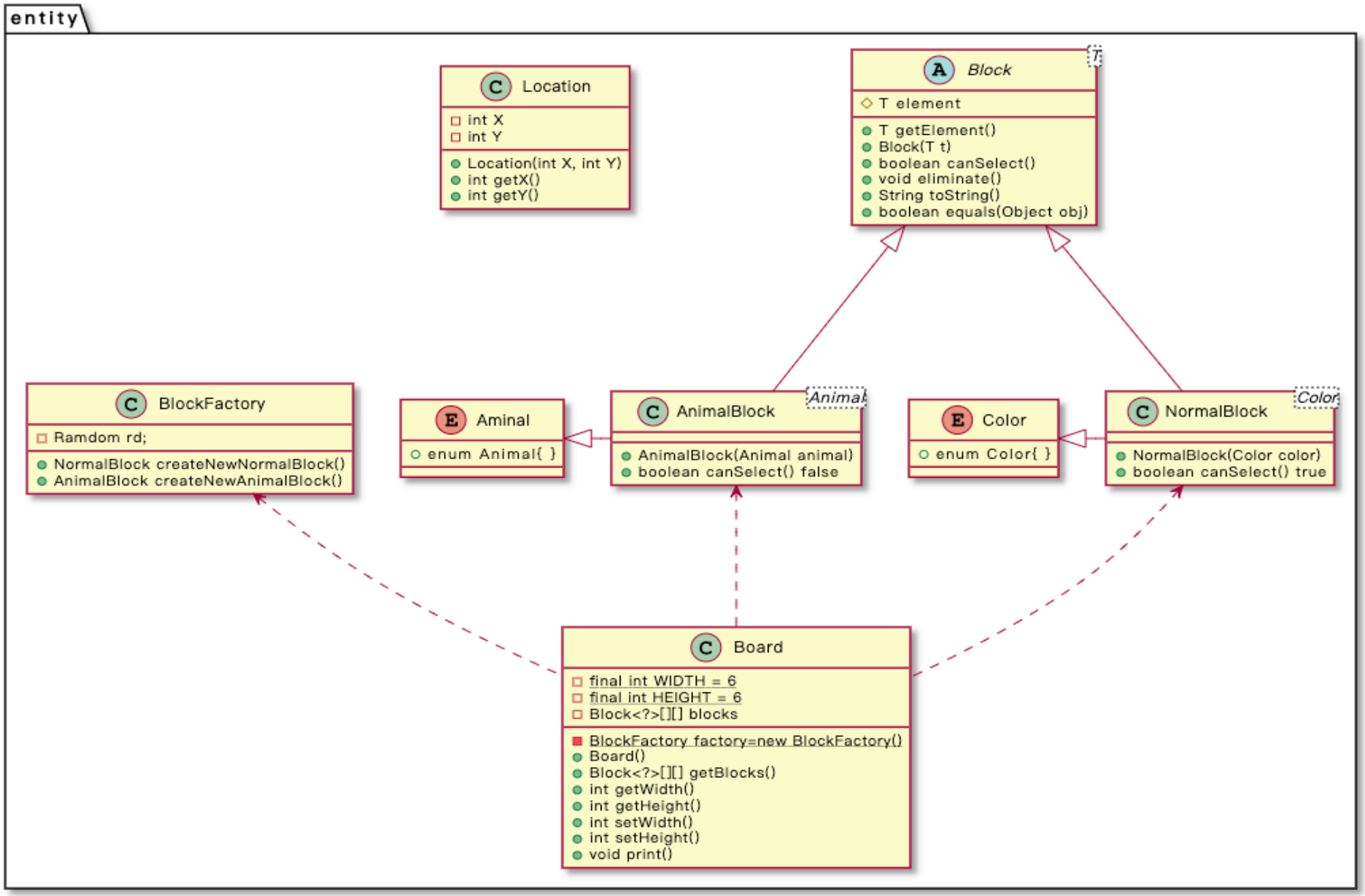
Interface utilisateur

L'interface de démarrage

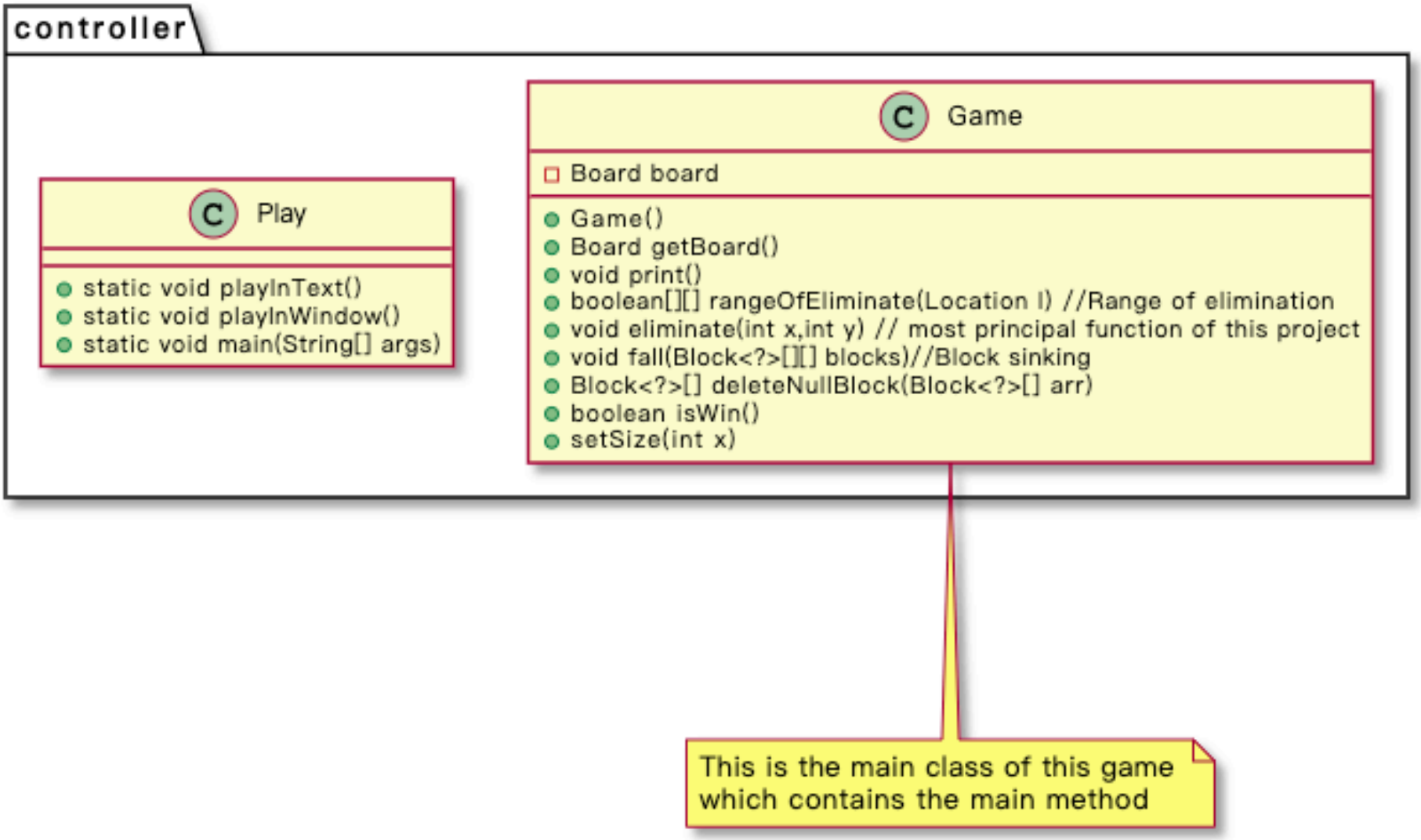
Figure: Choisir le mode de jeu

Table des manières

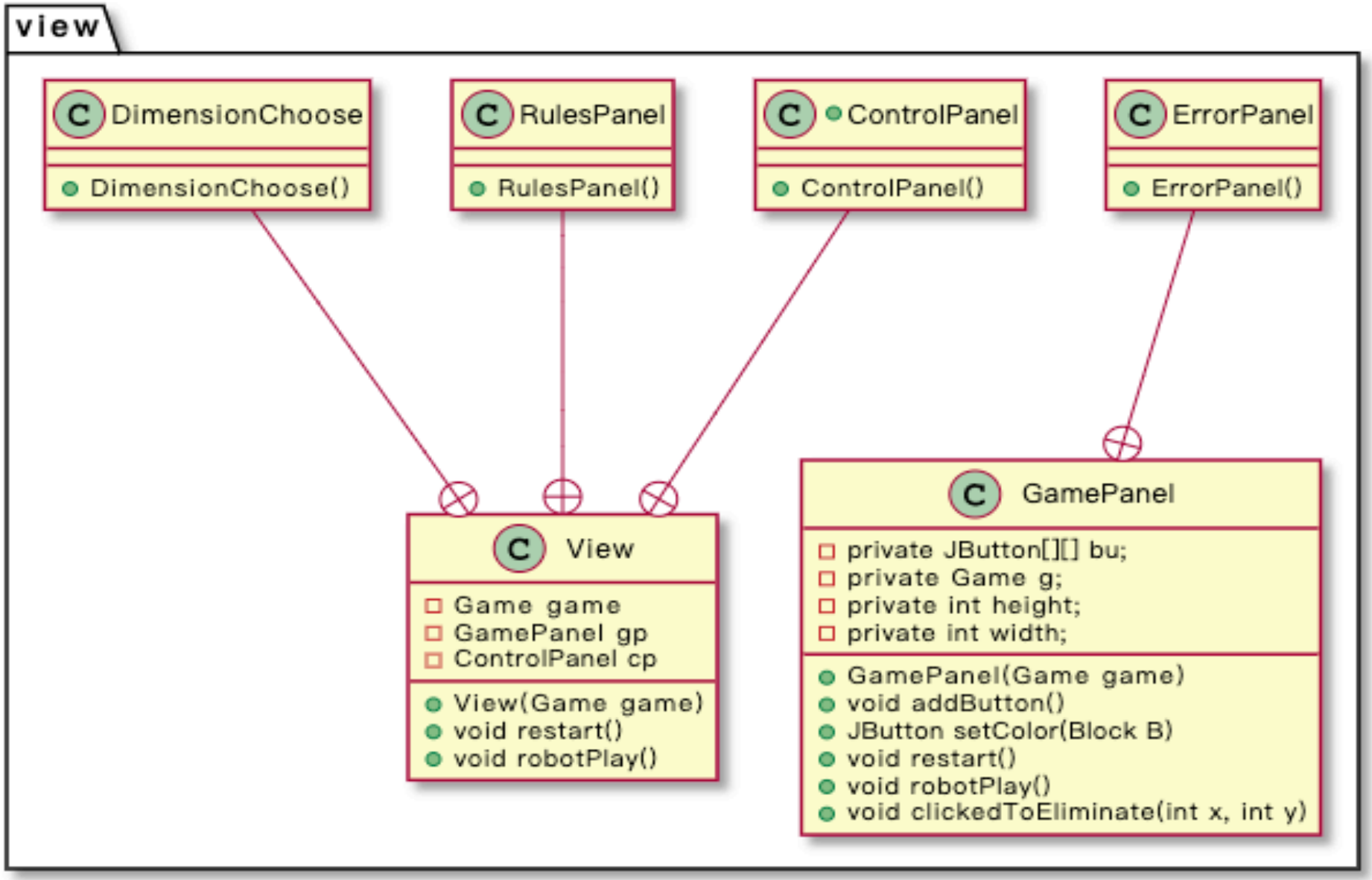
Conclusions Appendix : Diagramme de classes



Conclusions et Appendix : Diagramme de classes



Conclusions et Appendix : Diagramme de classes



Conclusions et Appendix : Diagramme de classes

Merci !

Bonne journée!