UNIVERSITÉ DE PARIS
DOUBLE LICENCE MATHÉMATIQUES-INFORMATIQUE ANNÉE 2020/2021
PROGRAMMATION ORIENTÉE OBJET ET L'INTERFACE GRAPHIQUE

UN PROJET SIMILAIRE AU PET RESCUE SAGA

Étudiant 1
Nom et Prénom YE Junli

No. Étudaint 21962349

Email junli.ye@etu.u-paris.fr

Étudiant 2
Nom et Prénom LI Songqiao

No. Étudaint 22013160

Email songqiao.li@etu.u-paris.fr

Table des matières

1. 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. Le Pet Rescue Saga est basé sur SameGame où le joueur sélectionne des cases adjacentes de même couleur pour vider le plateau de jeu, en libérant les animaux au sommet des cases une fois qu'ils ont atteint le fond.

Dans le développement de ce projet, nous avons appliqué avec souplesse les connaissances et les détails du cours à la mise en œuvre des caractéristiques et de l'interface du jeu. De plus, nous avons approfondi notre base de connaissances sur l'utilisation de github pour le contrôle des versions de code. Dans le développement du projet, on a utilisé les outils de UML et Markdown en dehors du programme du cours.

En raison de l'épidémie de covid-19, nous n'avons pas pu communiquer et interagir face à face pendant les travaux du projet. Cependant, nous utilisons toujours des outils tels que les plateformes de travail en ligne et la vidéo-conférence pour assurer la continuité du projet.

FIGURE 1.1: L'interface du jeu "Pet Rescue Saga" de King.com

2. La planification de projet

2.1 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.

2.2 Les principales règles du jeu

Les principales règles du jeu sont divisées en deux parties.

La première partie concerne les règles de l'interface utilisateur (c'est-à-dire le cube) pour effectuer des opérations. L'utilisateur peut cliquer sur un carré avec la souris dans l'interface graphique. Cette case est la case correspondante pour laquelle il souhaite effectuer une opération d'élimination. Dans le jeu, il n'y a que des directions parmi les carrés, en bas, à gauche ou à droite. Si au moins un des carrés adjacents à ce carré lui est identique, c'est-à-dire a la même couleur, alors ces carrés identiques seront éliminés ensemble. Si aucune case adjacente de la case sélectionnée n'est identique, la sélection n'est pas valable et aucune case ne peut être éliminée. Une fois l'élimination des blocs terminée, l'interface lâchera les blocs restants pour combler les lacunes selon la règle du haut vers le bas et de droite à gauche.

Dans l'interface texte, l'utilisateur saisit les coordonnées de la position correspondante via le clavier pour compléter la sélection du carré. Les autres règles sont inchangées.

La deuxième partie de la règle concernant la fin du jeu. Après chaque tour, le système détermine si le jeu a réussi, c'est-à-dire si tous les "animaux" sont retournés au sol. Si tous les blocs qui peuvent être éliminés l'ont été et que les animaux ne sont pas revenus au sol, le jeu est perdu. L'utilisateur peut toujours continuer à sélectionner des blocs, mais cela n'entraîne aucune action. En d'autres termes, l'utilisateur ne peut que terminer le jeu.

2.3 Organisation générale et répartition des tâches

L'organisation du travail est la première préoccupation de l'équipe. Après nous être réunis sur zoom, nous avons partagé nos horaires afin de trouver des moments où on pourrait se réunir et travailler ensemble.

TABLE 2.1: Tableau de réparation 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)

Nous avions notre réunion hebdomadaire tous les vendredi soir de 19h à 19h30 afin d'assurer la continuité du projet. Pour les dernières semaines pendant le vacances de la fin d'année, nous avions trois séance de réunion visioconférence.

Après avoir rempli les déclarations des binôme à la fin des vacances de la Toussaint, notre programme n'a officiellement commencé que le 21 novembre en raison de l'impact des ajustements des cours causés par le coronavirus COVID-19. L'avancement de l'ensemble du projet est correctement suivi et géré sur la plateforme github.

Au cours des deux premières semaines (21 novembre - 30 novembre), nous avons déballé et disséqué les exigences du projet. Nous avons essentiellement établi les tâches pertinentes à accomplir et les avons réparties. Junli s'occupera de la logique de base du programme et de l'architecture globale, ainsi que de la rédaction de ce rapport et du fichier README.MD. Songqiao se concentrera sur la conception et la mise en œuvre de l'interface graphique utilisateur, ainsi que sur la création des diapositives qui seront utilisées dans la soutenance. De surcroît, vu qu'on est demandé de réaliser une version textuelle du jeux en outre de l'interface graphique et cette version ne concerne que les packages sauf view. Ainsi Junli s'occuperait également sur cette version simple.

2.4 Outils utilisés et introduction

Java

Java est le langage utilisé pour le cours POO cette année. Il est un langage de programmation orienté objet, qui a non seulement les divers avantages de conceprt de l'héritage multiple avec les classes abstraites ou les interfaces, mais a également les fonctionnalités de Swing pour manipuler les interfaces graphique. Le langage Java est un langage de programmation orienté objet statique représentatif, excellente implémentation de la théorie orientée objet, permettant aux programmeurs de penser élégamment à la programmation complexe.

Java est le langage de programmation de ce projet, *i.e.* les codes du programme sont tous réalisés en Java.

Markdown

Markdown est un langage de balisage léger créé en 2004 par John Gruber avec l'aide d'Aaron Swartz. Elle a été créée dans le but d'offrir une syntaxe facile à lire et à écrire. Un document balisé par Markdown peut être lu en l'état sans donner l'impression d'avoir été balisé ou formaté par des instructions particulières.[2] Dans le projet, nous avous utilisé le langage Markdown pour manipuler le ficher README.md.

Github

GitHub est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git. Ce site est développé en Ruby on Rails et Erlang par Chris Wanstrath, PJ Hyett et Tom Preston-Werner. Par initiation du cours Pré-Projet, on a décidé d'utiliser le Github pour contrôler les versions du programme.

UML

Le Langage de Modélisation Unifié, de l'anglais *Unified Modeling Language (UML)*, est un langage de modélisation graphique à base de pictogrammes conçu comme une méthode normalisée de visualisation dans les domaines du développement logiciel et en conception orientée objet.[1] Le langage UML est utilisé dans ce programmme pour générer les graphes de classes *etc.*.

LaTeX

LaTeX est un langage et un système de composition de documents. Nous rédigeons nos rapports en utilisant du LaTeX pour les rendre plus professionnels.

3. Conception détaillée et mise en œuvre

3.1 Conception générale du programme : Class Diagrame

Inspirés par l'exercice de classe TD sur la façon de concevoir une structure de projet, nous avons décidé au tout début du projet d'identifier l'architecture globale du projet et les classes qui devaient être mises en œuvre en dessinant un diagramme de classe.

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* contient tous les objets nécessaires à ce projet, y compris la grille, le tableau, etc. Donc il joue le rôle de modèle. Cette partie gère les **données** de notre programme. Dans ce package, nous utilisons des classes abstraites et l'héritage pour exprimer la relation d'héritage entre les différentes tables.

Le package *view* est la partie se concentre sur **l'affichage**. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code Java de l'interface graphique comme Swing. Dans la version graphique du jeu, ce package permettra de garder le jeu proche de l'utilisateur. Dans la version graphique du jeu, ce package maintiendra un lien étroit entre le jeu et l'utilisateur. Toutefois, dans la version texte du jeu, nous n'appelons pas les méthodes dans ce paquet, c'est-à-dire que cela n'aura pas d'impact sur la version texte du jeu.

Le package *controller* gère la logique du code qui prend des **décisions**. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : Le contrôleur obtiendra l'entrée de l'utilisateur à partir de la vue, puis suivra la logique et appellera l'objet ou l'idée pertinente au sein de l'entité si nécessaire. Enfin, à la fin du calcul, une nouvelle situation est renvoyée à la vue pour l'afficher à l'utilisateur. Les codes logiques de base le plus important du jeu est également présent dans la classe Game du package controller.

Le diagramme de classe complet peut être retrouvé dans la pièce jointe.

3.2 Les objets du programme

Structure de l'héritage : Généricité et classe abstraite

L'architecture des objets est réalisé par l'héritage. Les carrés normaux qui indiquent des couleurs et les carrés d'animaux qui ne peuvent pas être éliminés sont tous deux des types de carrés. Dans ce cas, j'ai choisi d'utiliser une classe abstraite plus un type générique pour traiter ce problème.

```java
package entity;

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

```java
package entity;

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

```java
package entity;

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

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

Une fois que nous aurons fini de mettre en place tous les éléments, nous penserons à faire la mise en place du *Board*.

Board

La classe *Board* est l'élément le plus essentiel du package *entity*. Elle possèdes quatre champs décrivant les proproté du board en respectant la principe idée : Le board est représenté par un tableau de deux dimension (*an array of two dimensions*).

```java
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;
```

Afin de garantir la sécurité du stockage des données, nous avons adopté le "principe de visibilité minimale", c'est-à-dire que tous les paramètres susmentionnés ont une visibilité aussi faible que possible.

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

TABLE 3.1: 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

méthode equals() reste possible. Ainsi on a aussi override la méthode equals() dans la classe Board.

3.3 La logique du jeux en base

Les codes de la logique du jeux en base est crés dans la classe Game du package Controller. Une calsse Game ne prend qu'un seul attribut en paramètre : un Board. Ce board-là est définit lorsqu'on commencer à jouer par l'appel du constrcteur par le lanceur, *i.e.* l'appel du constructeur par la classe Play.

Le processus d'élimination des carrés

Pour parvenir à l'élimination des blocages mutuels, nous déconstruisons l'ensemble du processus en trois étapes.

La première étape est réalisée par la fonction rangeOfElimination (Fonction 1), par laquelle nous calculons si l'un des carrés autour du carré spécifié remplit la condition d'élimination (c'est-à-dire que le carré peut être éliminé et que ce carré a la même couleur que le carré spécifié). Une fois le calcul effectué, la fonction retournera un tableau bidimensionnel, où une valeur vraie signifie que le carré à cette position peut être éliminé.

Dans la deuxième étape, la fonction de la méthode sera mise en œuvre dans la méthode eliminate(). Le programme éliminera les carrés pertinents en fonction du tableau obtenu lors de la phase précédente

Dans la troisième étape, les carrés vides sont éliminés par la méthode fall(). La méthode d'élimination des carrés blancs est légèrement plus compliquée que lors des deux phases précédentes. Nous y parvenons en introduisant une nouvelle méthode qui génère d'abord un nouveau tableau de ce tableau bidimensionnel dans l'ordre des colonnes. Ce tableau est ensuite transformé en un arrayList<Block> et tous les éléments à valeur nulle du tableau sont ordonnés jusqu'à la fin. Après avoir reconverti la liste obtenue après l'alignement en un tableau, nous mettons ainsi en œuvre la chute du carré sur chaque colonne.

Le graphe dessous décrit détaillément le logique comment on réalise "un tour" du jeux :

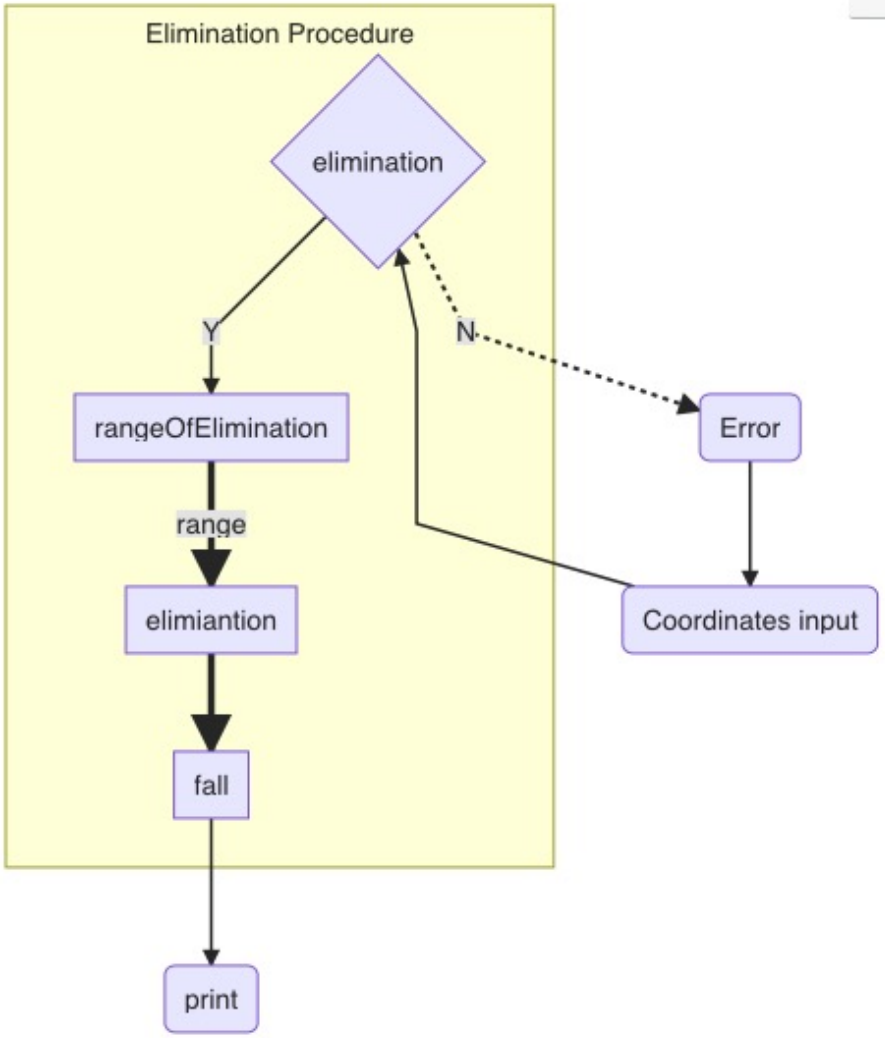


FIGURE 3.1: Logique d'Elimination

3.4 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.

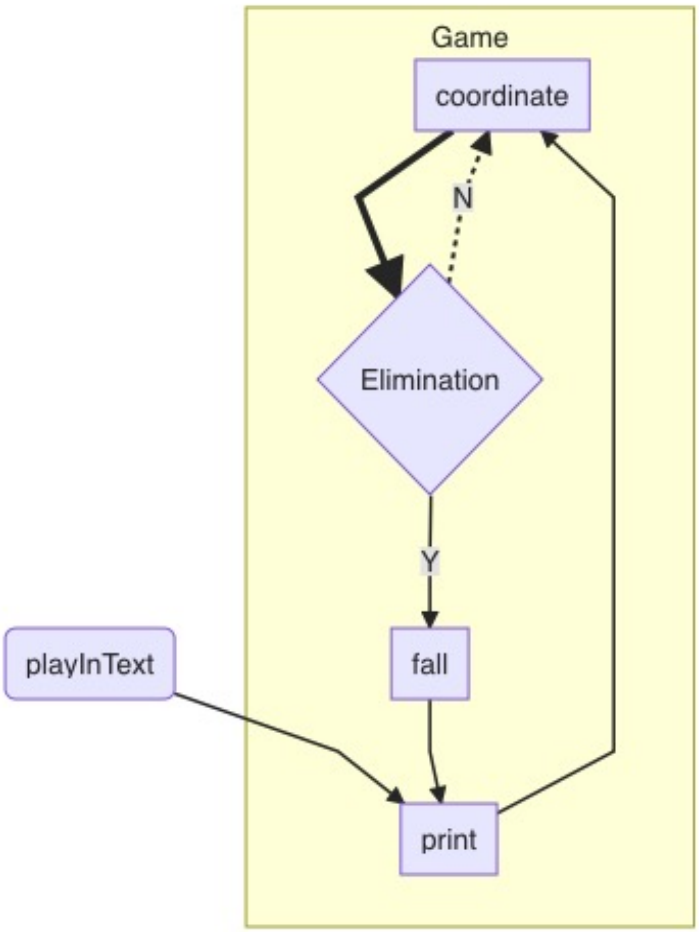


FIGURE 3.2: PlayInText

On a un capture d'écran de la version texte deu jeux :

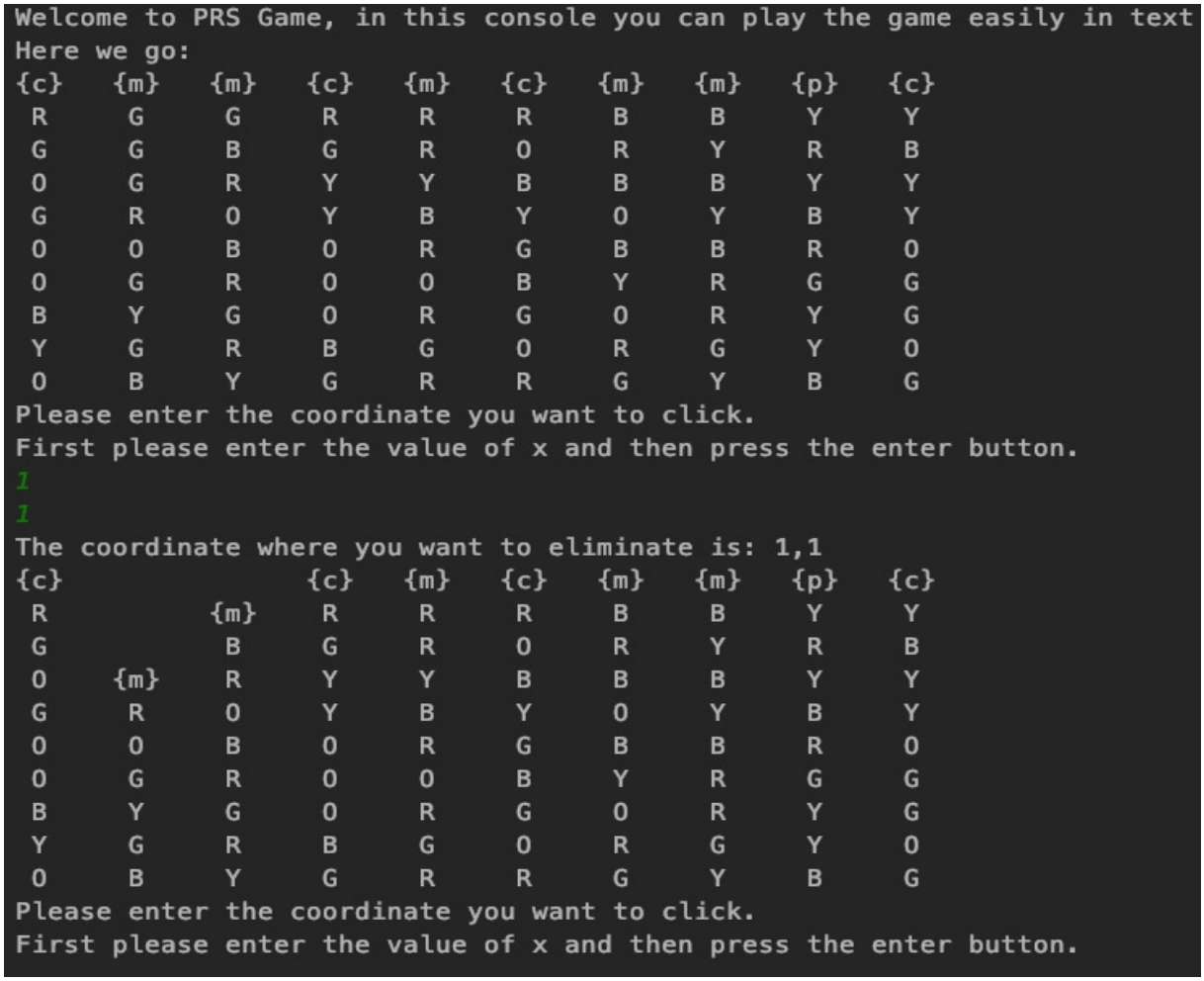


FIGURE 3.3: L'intereface au Terminal à la version text

3.5 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.

Vu que les parties d'interface physique est nouvelle pour le cours, on a bien utilisé les connaissances vus dans ce semestre. Par exemple, les expressions lambda sont utilisées très souvent lorsqu'on implémentait les boutons et fenêtres. On a également essayer d'assurer le "Thread Safe" pour les procédures plus complexe.

La fenêtre principal se compose de deux parties : la moitié supérieure est l'interface visuelle du jeu, et la moitié inférieure comprend quatre boutons de fonction.

Difficult

Ce bouton permet l'utilisateur de choisir l'un des difficultés parmi les trois choix. Lorsque l'utilisateur cliquetai sur ce bouton, la fenêtre renverrait un nouveau sous-fenêtre qui demande l'utilisateur de choisir la difficulté.

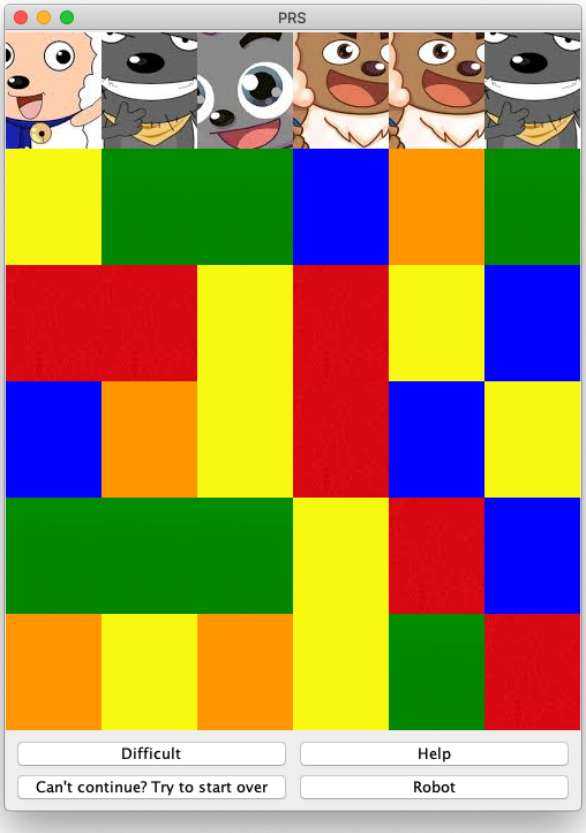


FIGURE 3.4: L'interface principal

FIGURE 3.5: Sous-fenêtre "Difficult"

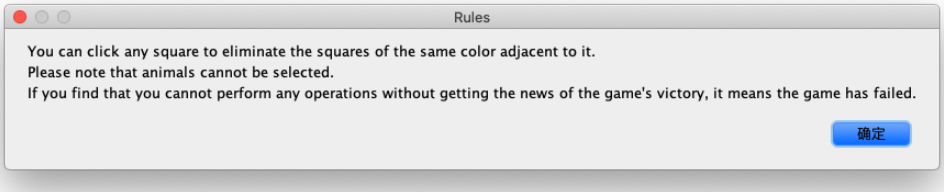


FIGURE 3.6: Sous-fenêtre "Help"

Help

Ce bouton renverrait un nouveau fenêtre introduisant les règles du jeux en quelques phrases simples quand l'utilisateur le clique.

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. Il est important de noter que la fonction de bot n'est pas conçue pour effectuer toutes les actions en une seule fois, mais par étapes, afin de garantir la participation de l'utilisateur et de lui permettre d'envisager plus facilement l'utilisation du bot à une étape particulière. Bien sûr, si l'utilisateur veut que le bot termine le jeu en entier, il peut continuer à déclencher le bouton "robot" jusqu'à ce que le bloc ne puisse plus être éliminé. Un autre objectif de cette conception est que l'utilisateur puisse utiliser cette fonction comme une aide : lorsqu'il ne sait pas par où commencer, il peut choisir cette fonction.

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());
```

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
                }
        }
}
```

Thread-Safe

Actuellement, les systèmes d'exploitation traditionnels sont multi-tâches, c'est-à-dire que plusieurs processus s'exécutent simultanément. Par souci de sécurité, chaque processus ne peut accéder qu'à l'espace mémoire qui lui est alloué, mais ne peut pas accéder aux autres processus, ce qui est garanti par le système d'exploitation.Il y aura une zone publique spéciale dans l'espace mémoire de chaque processus, généralement appelée le tas. Tous les threads du processus peuvent accéder à cette zone, qui est la cause potentielle du problème.

Pour poster une action à executer dans la file des événements, le main doit invoquer la méthode EventQueue.invokeLater(Runnable) où Runnable est un interface fonctionel.[3] Dans notre projet, on a également réalisé le thred-safe en utilisant le bloc du code suivant dans le class "main" (la class play du package controller).

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

4. Fonctionnement du projet

Selon les exigences du projet, nous avons implémenté deux interfaces différentes de jeu, version texte et version image, basées sur la même logique sous-jacente. Afin de faciliter l'installation et l'utilisation de l'utilisateur, nous avons également conçu une interface de démarrage.

Pré-requis

Bien sur, ce jeux marche sur l'environnemt de Java donc un Java JRE est demandé pour le premier bas. En plus, lorsque l'utilisateur a téléchargé successivement le fichier .zip du projet, il pourra simplement entrez dans le chemin de ce projet par son console, ex. Terminal.

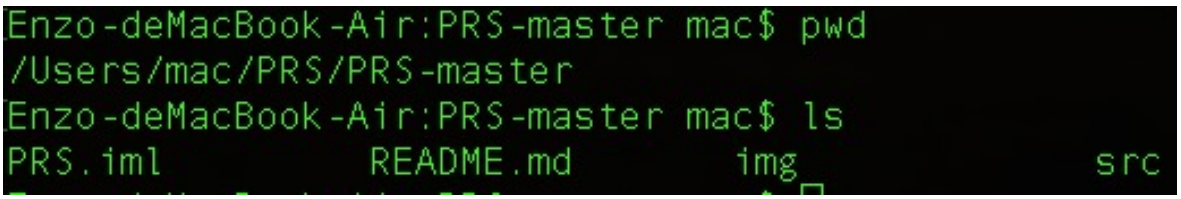


FIGURE 4.1: Le fichier PRS.zip dans le console

L'utilisateur doit les instructions données dans le fichier README.md pour compiler le projet et ensuite il peut démarrer le proejet tout simplement.

FIGURE 4.2: Compiler et exécuter le jeux

L'interface de démarrage

L'utilisateur entre maintenant dans l'interface de démarrage. Il doit tout d'abord choisir la langue parmi les trois langues offert dans le console : soit anglais, soit français, soit chinois. Ensuite, le jeu demandera le nom de l'utilisateur et dit bonjour à l'utilisateur. Après avoir terminé toutes les procédures ci-dessus, le programme demandera à l'utilisateur quelle version du jeu il souhaite choisir et le démarrera.

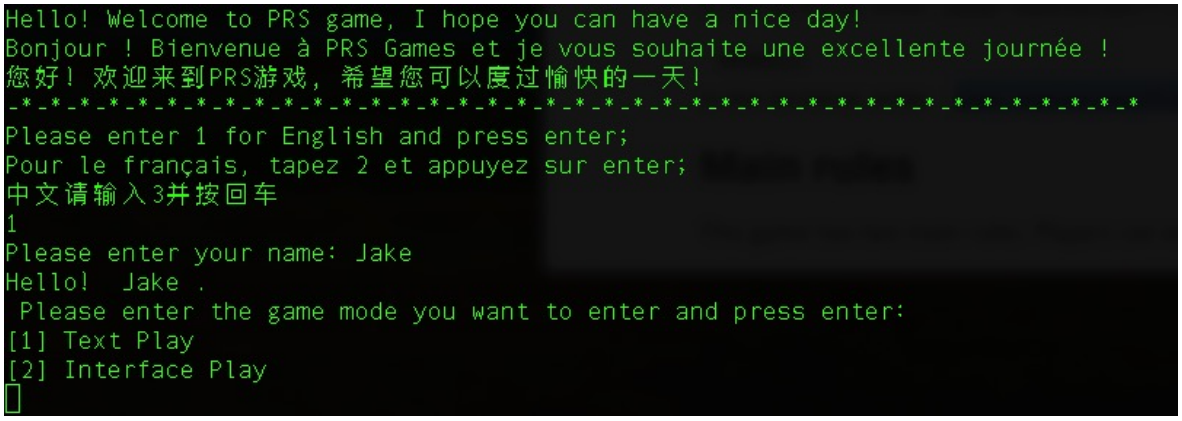


FIGURE 4.3: Choisir le mode de jeu

Il est à noter que nous avons conservé la sortie texte en mode graphique dans la version de développement du programme. C'est-à-dire que lorsque l'utilisateur sélectionne le carré dans l'interface graphique pour l'éliminer, le jeu sortira de manière synchrone la position du clic de l'utilisateur et le résultat du calcul sous forme de texte dans le terminal. Le but principal du maintien de cette fonction est de permettre aux développeurs (ou utilisateurs de test) de visualiser les résultats de calcul correspondant aux actions correspondantes, et de vérifier si le rafraîchissement de l'interface graphique est correct.

5. Conclusions

5.1 Revoir le projet

C'est la première fois de notre vie que nous essayons de développer un projet Java complet (si le fichier d'instructions initial est ignoré). Pour être honnête, le développement de ce projet nous pose encore certains défis, ce qui nous rappelle également que nous devons continuer à solidifier les connaissances de base dans l'étude suivante, mais aussi réfléchir à comment les utiliser de manière flexible. Le fonctionnement du projet n'est pas seulement le travail d'une personne, mais la cristallisation des efforts d'une équipe. Par conséquent, nous devons avoir de grandes capacités de communication et d'expression.

En revenant sur le développement de l'ensemble de ce projet, nous avons fait de notre mieux pour utiliser les connaissances requises par le cours de POO-IG de ce semestre : objets, encapsulation, polymorphisme, héritage, classes internes, exception, expression lambda, générécité, interfaces graphiques, etc. En plus du contenu du cours, afin de mieux compléter ce projet, nous avons également appris nous-mêmes l'utilisation des langages uml, markdown et latex, et avons produit les résultats correspondants.

L'ensemble de notre programme comporte encore de nombreux défauts.

Dans la classe *Game*, la complexité de l'algorithme de calcul le zones d'élimination est élevé. Cela oblige le programme à prendre beaucoup de mémoire pour effectuer ce calcul pendant le processus en cours, ce qui prend du temps. Nous pouvons encore faire quelques recherches sur cette question : comment rendre cet algorithme plus efficace ?

De plus, dans la conception préliminaire du jeu, nous espérons ajouter une musique de fond et des effets sonores de fonctionnement. C'est une fonctionnalité très intéressante. Mais nous n'avons pas pu terminer cette fonction après de nombreuses tentatives, nous avons donc dû l'annuler. C'est également devenu un gros regret dans le jeu.

L'ensemble du projet est développé via github pour la gestion des versions et les mises à jour de progression. Maintenant, ce projet a été rendu public sur github. En même temps, à l'aide de fonction "insights" du github, on peut analyser les tâches faits par chacun dans l'équipe.(Voir 5.1)

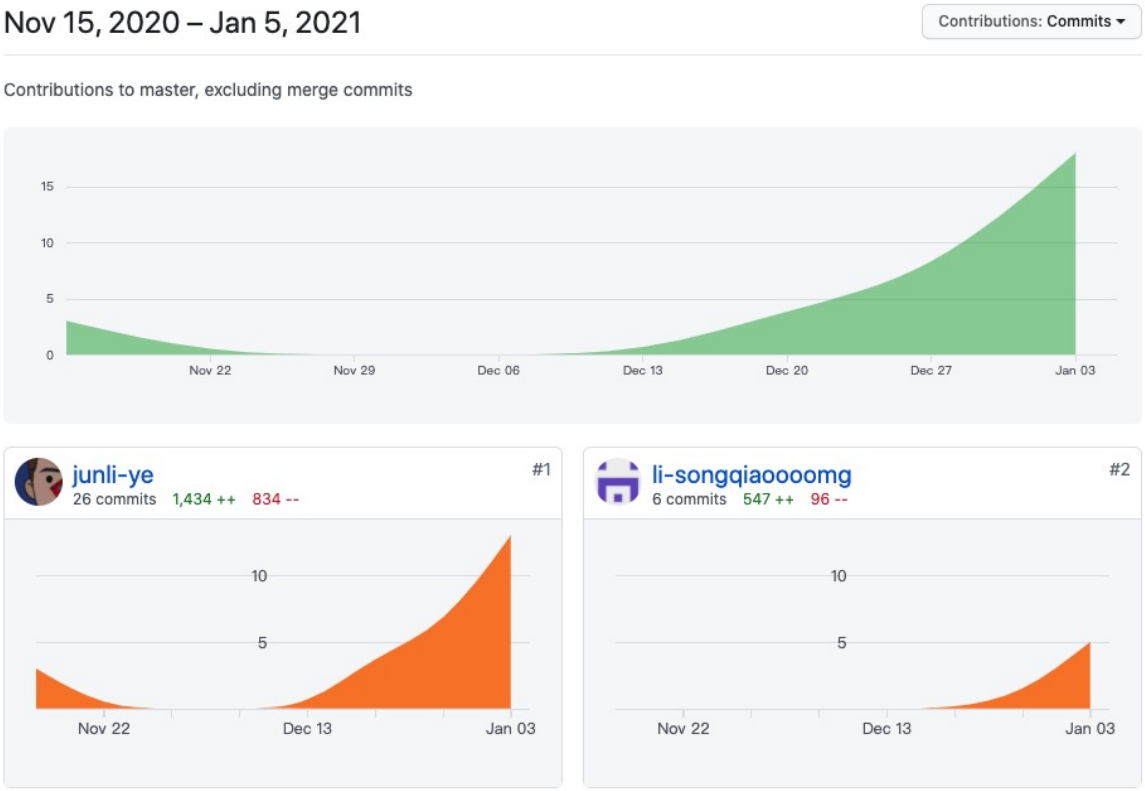


FIGURE 5.1: Les commits contribués entre 15 nov. 2020 et 5 jan. 2021

5.2 Apport du projet

Songqiao

Par ce projet , nous travaillons ensemble , discussion notre idée . Nous comme vrai développeurs dans un team dans un entreprise. En ce cours, on maîtrise les outils comme git, uml et latex. Et il améliore notre niveau de discussion quand on face un projet .En plus ça nous apprend comme coder une application qui réponds aux besoins des utilisateurs .

Personnellement , c'est le premier projetcar je n'ai pas app. Ça m'apprend beaucoup de mon camarade Junli,pas seulement connaissance sur POO ,mais aussi comment organiser un projet et trouver les outils .

Junli

Pour moi, bien que le projet m'ennuie beaucoup surtout dans les derniers jours avant le dépôt, mais par contre il m'a beaucoup aidé à réviser tout les connaissances sur Java. J'ai appris à télétravailler ensemble en utilisant les outils comme Zoom etc.(même on l'a déjà fait l'année dernière), à équilibrer la répartition des tâches, à la rechercher des informations, à lire des manuels pour apprendre un petit langage pour m'aider. Bien sûr, je pense que notre projet n'est pas bon : il y a encore des exceptions et des regrets sur les méthodes qui ne se sont pas réalisés. Cependant, je pense que la plus grande importance de ce projet est de m'apprendre à travailler en équipe et à communiquer. Je réalise également que la ponctualité est une qualité très importante dans le travail d'équipe.

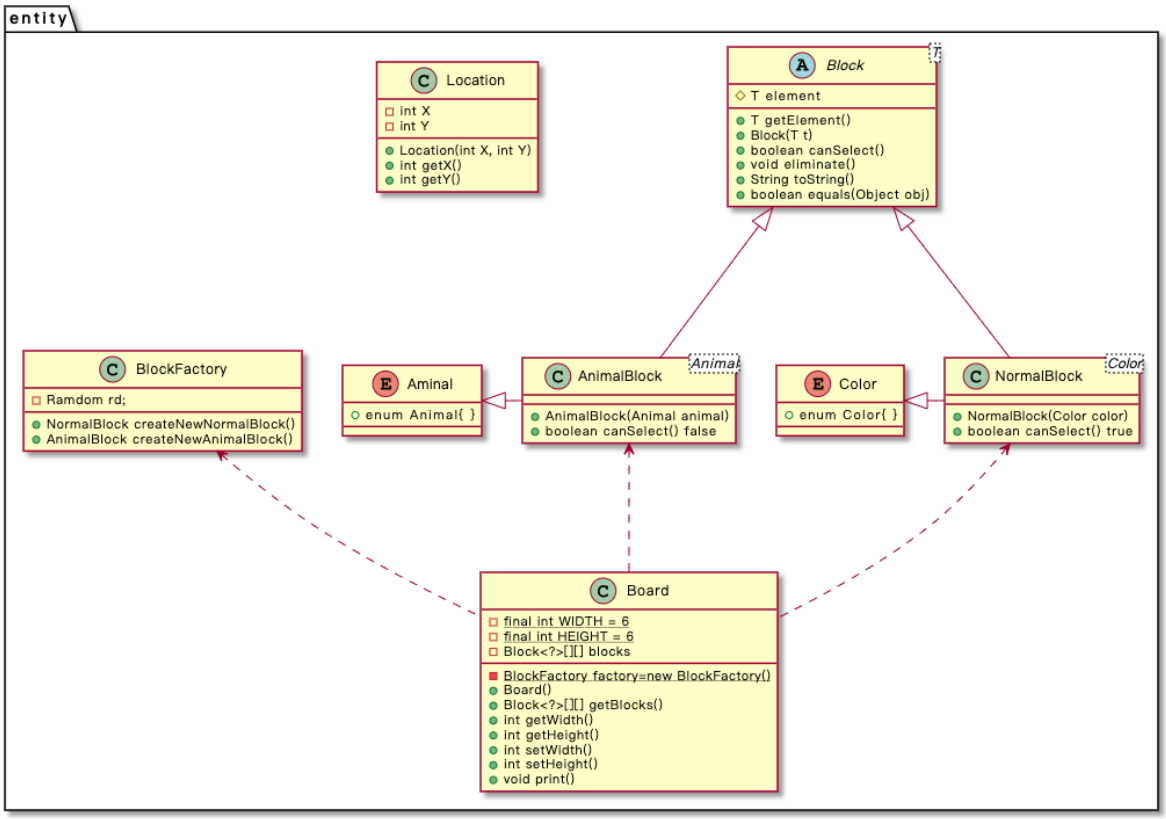
References

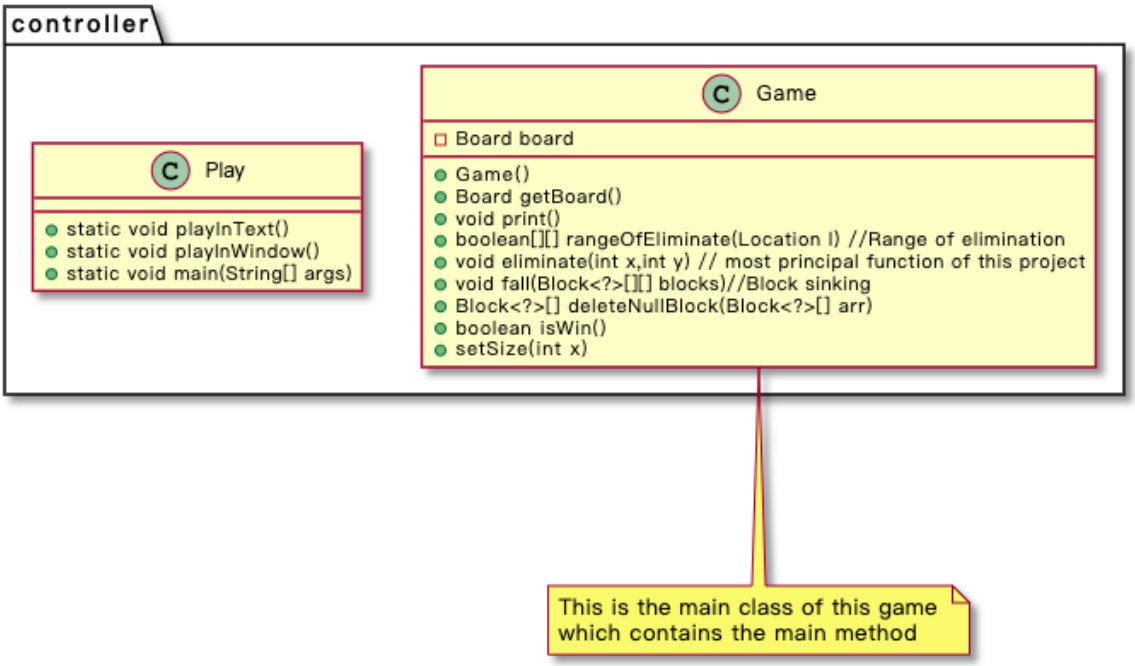
[1] Drawing uml with plantuml, plantuml language reference guide. page Z1, 2020. [Cited on page 4]

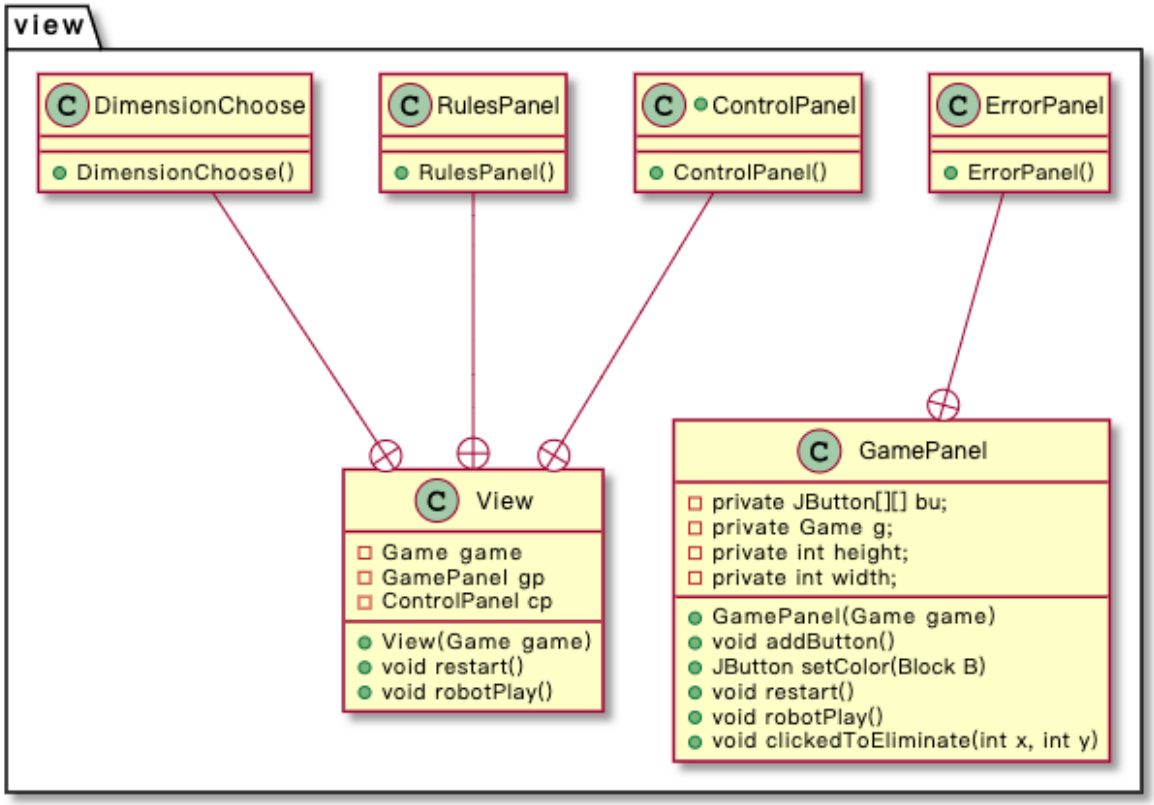
[2] John Gruper. Markdown cheatsheet. [Web Page]. https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet/ Accessed Jan. 2, 2021. [Cited on page 4]

[3] Cristina Sirangelo. Interface graphique (cours de poo-ig). 2020. [Cited on page 13]

Appendix : Diagramme de classes




controller
Play
static void playInText()
static void playInWindow()
static void main(String[] args)
Game
Board board
Game()
Board getBoard()
void print()
boolean[][] rangeOfEliminate(Location l) //Range of elimination
void eliminate(int x,int y) // most principal function of this project
void fall(Block<?>[][] blocks)//Block sinking
Block<?>[] deleteNullBlock(Block<?>[] arr)
boolean isWin()
setSize(int x)
This is the main class of this game
which contains the main method


view
DimensionChoose
DimensionChoose()
RulesPanel
RulesPanel()
ControlPanel
ControlPanel()
ErrorPanel
ErrorPanel()
View
Game game
GamePanel gp
ControlPanel cp
View(Game game)
void restart()
void robotPlay()
GamePanel
private JButton[][] bu;
private Game g;
private int height;
private int width;
GamePanel(Game game)
void addButton()
JButton setColor(Block B)
void restart()
void robotPlay()
void clickedToEliminate(int x, int y)