Devoir de Contrôle Continu
Complément de POO

Jeu de Bataille Navale

Nguyen Phuong Vy VU - 21911658

L2 Informatique - Groupe 2A - Année 2020-2021

2 mai 2021

Table des matières

1 Introduction

1.1 Bataille Navale

Bataille Navale[1], également appelé Touché-Coulé, est un grand classique des jeux de sociétés pour deux joueurs. Son origine est dans un jeu français "L'Attaque" au début des années 1930.

Pour commencer ce jeu, deux joueur qui s'affrontent placer ses bateaux sur sa grille. Chacin a une flotte composée de cinq bateaux, qui sont un porte-avion, un croiseur, un contre-torpilleur, un sous-marin, un torpilleur et les bateaux ne doivent pas être collés entre eux, un navire est coulé si chacune de ses cases ont été touchées par un coup de l'adversaire.

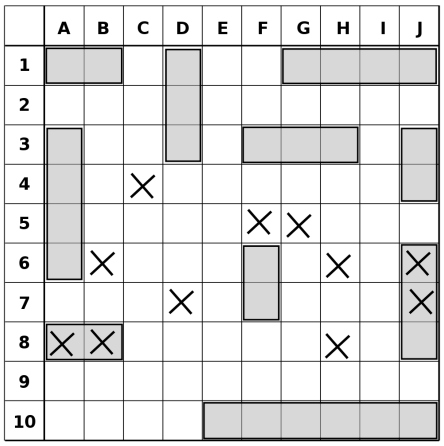

FIGURE 1 – Grille du jeu

Le gagnant est celui qui ales statégies à couler tous les navires de l'adversaire avant que tous les siens ne le soient.

1.2 But du projet

Dans ce projet, on a développé un jeu avec la conception Modèle-Vue-Contrôleur (MVC) qui permet de jouer sur une interface graphique. En effet, on a lancé un planning de ce projet en plusieurs temps :

— Lancer de la conception du projet (diagramme, librairie, méthode)
— Développer du moteur du jeu, selon les règles du jeu
— Développer l'interface et le contrôleur
— Optimiser et vérifier le code pour éviter les bugs

2 Organisation du projet

2.1 Hébergement du code

On a choisit le forge d'Unicaen pour faciliter la gestion du projet. Il permet de créer et d'administrer des dépôts sous le forge que l'on peut en faire avec terminal. De plus, le forge offre les autres fonctionnalités qui sont une gestion des permissions, une visualisation des différents commits, le suivi de l'activité du projet, etc.

2.2 Tableau des tâches

Classes	**Tâches**
Création des modèles	
Case	Contenir un point des coordonnées et un bateau
Ship	Construire un bateaux avec un poids
Création du plateau	
Board	Construire le plateau Placer les bateaux Créer la fonction de viser
Création de MVC	
Draw	Dessiner l'interface graphique Viser dans le plateau avec le souris
Main	Initialiser les bateaux Lancer le jeu

3 Architecture du projet

3.1 Directoires du projet

La structure du projet est non seulement de diviser le projet en des parties spécifiques, mais encore doit adopter la structure de Gradle. Il est structuré en trois parties.

```
rapport
gradle‣wrapper
src‣main‣java
  board
    Board.java
  model
    Case.java
    Ship.java
  view
    ViewController.java
  Main.java
```

On remarque les packages utilisés :
— board : initialiser le plateau avec les bateaux et les fonctions du jeu
— model : créer modèle du jeu
— view : dessiner le plateau et l'écouteur de souris

3.2 Diagramme des classes

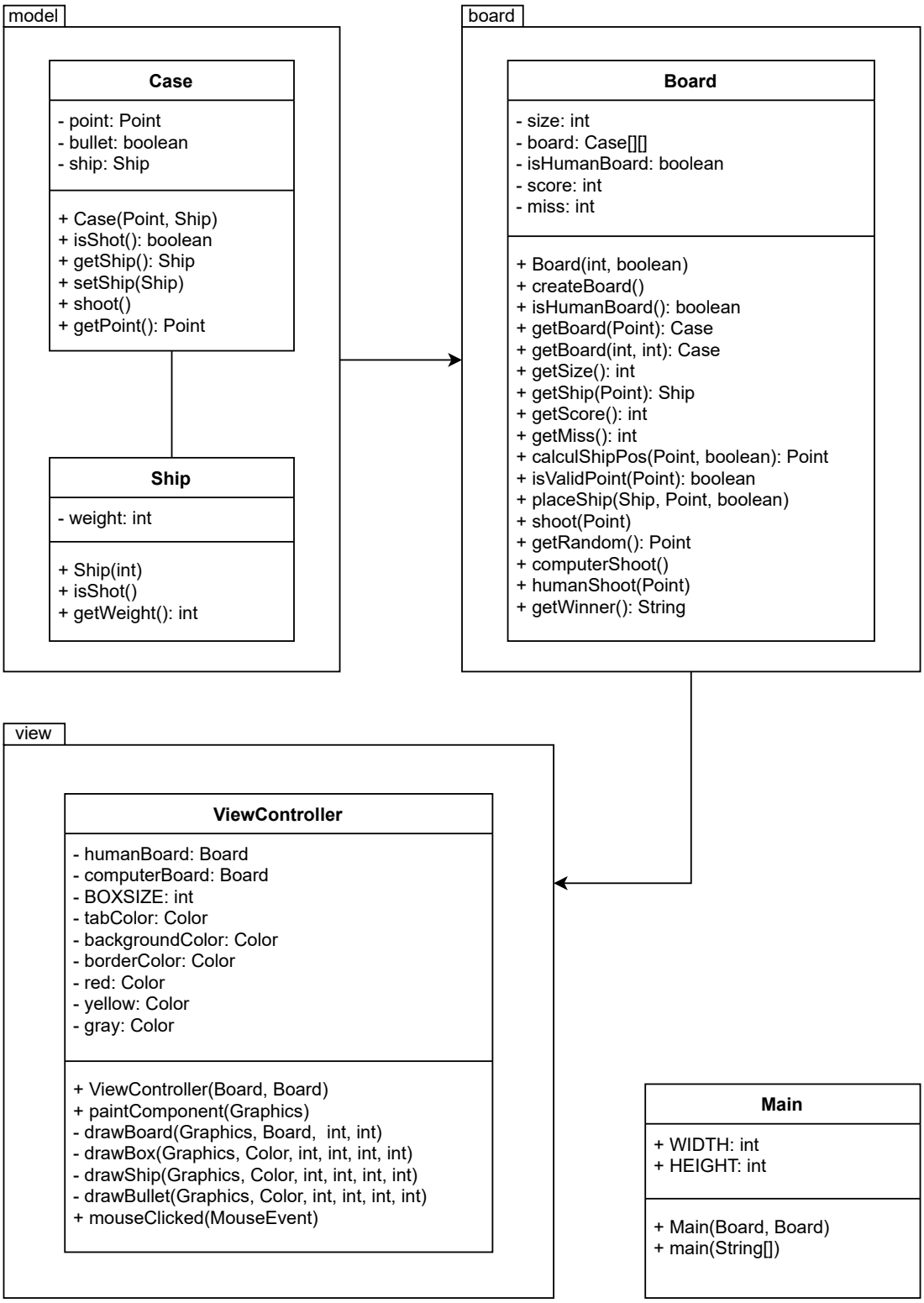

FIGURE 2 – Diagramme des classes

4 Développement du projet

4.1 Création du plateau

Au début, on choisit la taille du plateau est de 10x10, 100 carrées. Donc, on crée la classe des cases nommées **Case** qui sera une petite carrée du plateau. Il y a des cases qui contiennent des bateaux ou sont nulles. On initialise les bateaux dans la classe **Ship** avec les poids et une case contient un poids du bateau.

Ensuite, on construit la classe de plateau **Board** avec les cases avec le point en correspond en défaut (pas de bateau) et avec le boolean pour spécifier le plateau. Après avoir créé les cases sur le plateau, on fait une méthode de placer les bateaux `placeShip(Ship, Point, boolean)`. On note avec le boolean `true`, le bateau veut placer en vertical et vice versa, le boolean `false` est en horizontal.

Finalement, après avoir créé deux plateaux sur la classe **Main**, l'un de joueur et l'autre d'ordinateur. On place les bateaux sur chaque plateau.

Ce sont des bateaux et des poids vont initialiser :
— 1 Porte-avions (poids de 5)
— 1 Croiseur (poids de 4)
— 1 Contre-torpilleurs (poids de 3)
— 1 Sous-marin (poids de 3)
— 1 Torpilleur (poids de 2)

4.2 Viser sur le plateau

Selon aux règles du jeu, on fait la méthode de viser les bateaux par rapport au boolean des cases. Si un case est visé dont le boolean devient `true`. Quand on vise sur un bateau, le poid sera perdu et si le poid d'un bateau est de zéro, on dit ce bateau est coulé. En outre, on fait une méthode de viser par ordinateur qui utilise la classe **Random** avec le boucle pour trouver les points disponibles.

4.3 Design Pattern Modèle-Vue-Contrôleur (MVC)

Dans cette partie, on construit l'interface du jeu par la librairie Java Swing qui a été inclus au JDK depuis la version 1.2, cette libraire est moins compliquée d'installer que la librairie JavaFX, avec la conception Modèle-Vue-Contrôleur (MVC). Et on souhaite de ne pas lancer la ligne de commande quand on joue le jeu.

Java Swing[2] fait partie de la bibliothèque Java Foundation Classes (JFC). C'est un API dont le but est similaire à celui de l'API AWT, mais son fonctionnement et son utilisation sont complètement différentes.

Modèle-Vue-Contrôleur ou MVC[3] qui est composé de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs (un motif d'architecture logicielle destiné aux interfaces graphiques).

4.3.1 Modèle

Un modèle est les données à afficher. Par conséquent, on développe deux classes de modèle qui sont **Case** et **Ship**.

4.3.2 Vue

Une vue présente l'interface graphique. On dessine la grille de plateau en utilisant l'opérateur `instanceof` **Case** qui permet de tester si un objet possède, pour déterminer une carrée. On décide des bateaux coulés sont les rectangles rouge, des obus sont les cercles, les obus sur une case nulle est de couleur yellow et sur un case de bateau est de couleur rouge. On va dessiner avec les fonctions principales `fillOval` et `fillRect` de la classe **Graphics**

En résumé, on indique les remarques de l'interface graphique sur l'image ci-dessous.

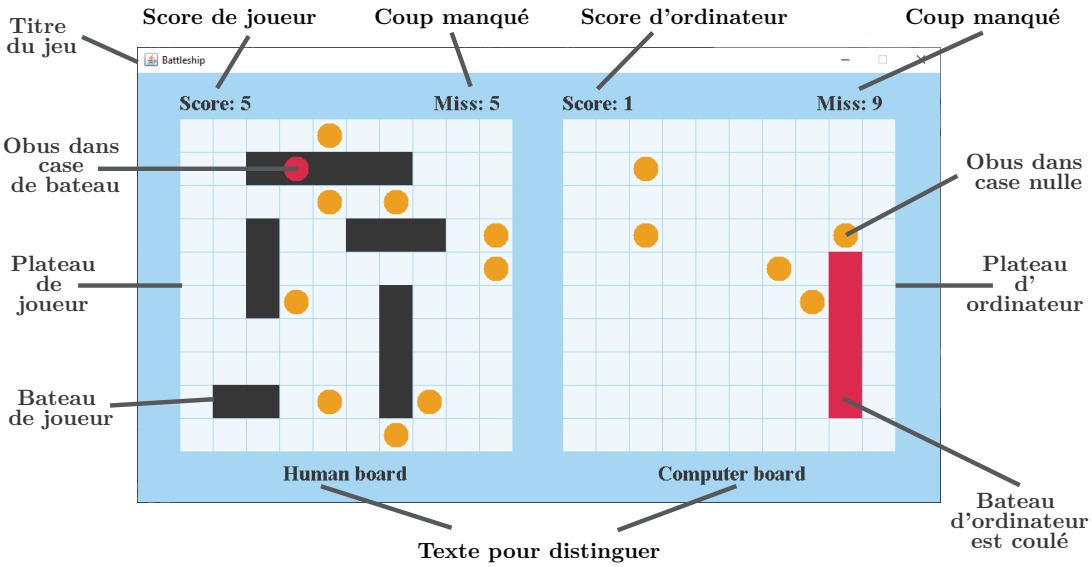

FIGURE 3 – Remarques de l'interface graphique

4.3.3 Contrôleur

Un contrôleur contient la logique concernant les actions effectuées par l'utilisateur. Pour faire les contrôleurs, on utilise l'interface **MouseListener** dans le package **java.awt.event**.

Premièrement, on calcule le point de case par rapport les coordonnées de souris. On propose la formule pour calculer le point (x, y) de robot de la coordonnée (xPos, yPos) :

$$x = (xPos - PADDING)/BOXSIZE$$
$$y = (yPos - PADDING)/BOXSIZE$$

Deuxièmement, pour viser sur le plateau avec le souris, on met les conditions des coordonnées pouvant fonctionner dans les zones autorisées de la souris et utilise la méthode `mouseClicked(MouseEvent)` pour appeler la fonction de viser.

5 Instruction du jeu

5.1 Lancer le jeu

Il y a beaucoup du logiciel de construction automatisée d'application qui automatisent la création d'applications exécutables à partir du code source sur la marché comme ANT, Maven et Gradle. Sur ce projet, on décide d'utiliser le Gradle, car c'est simple pour écrire le script et pour lancer.

Entrez la commande `./gradlew run` pour lancer le jeu. Ça fonctionne avec le Linux Shell et Windows Powershell.

5.2 Jouer le jeu

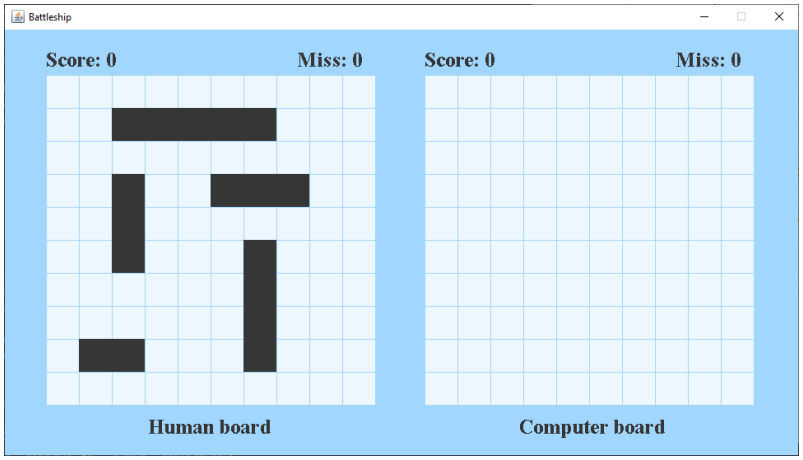

FIGURE 4 – Initialisation du plateau

On clique sur le plateau d'ordinateur pour viser. Quand tous les bateaux sont coulés, le jeu est fini.

6 Captures d'écran

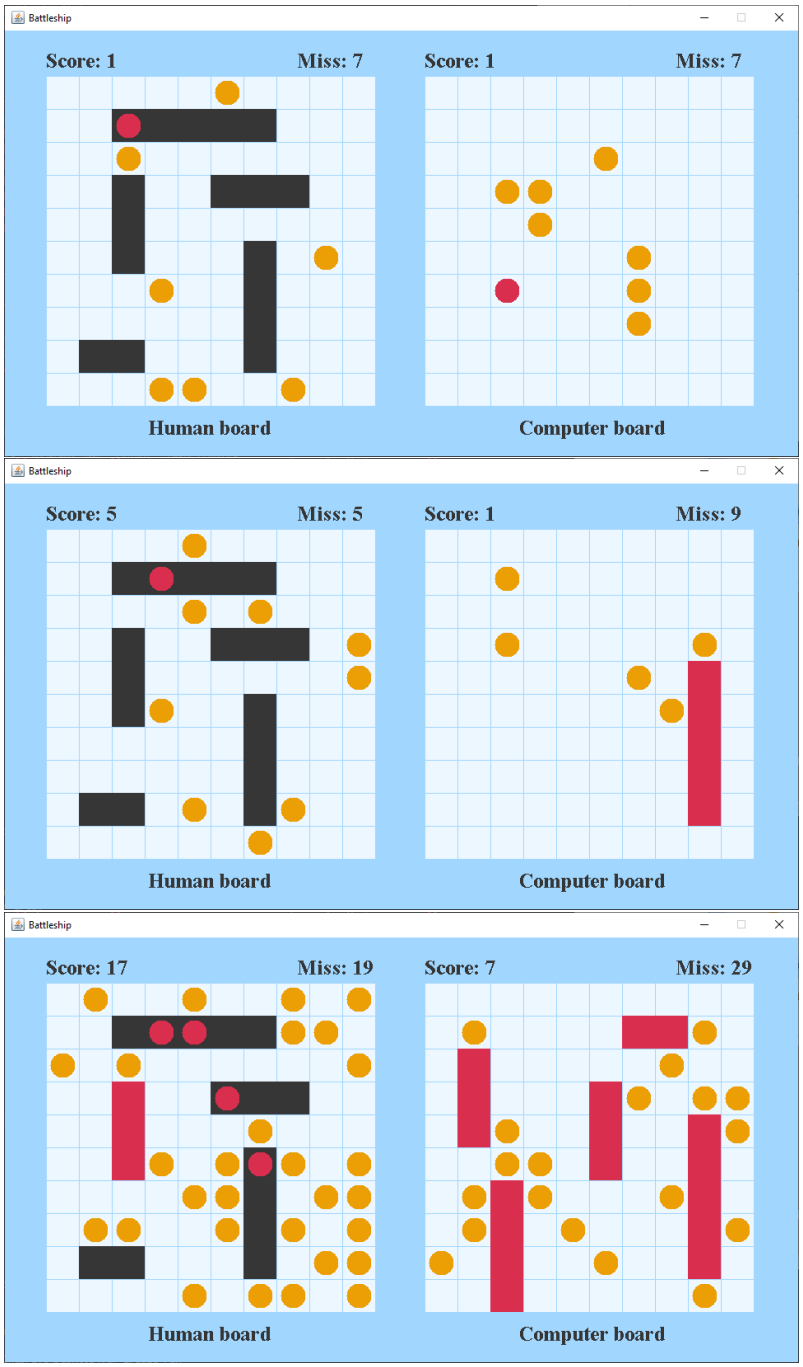

FIGURE 5 – Jeu de Bataille Navale

7 Conclusion

Après ce projet, j'ai mieux compris la libraire Java Swing, le Java Build Tools Gradle, et le jeu de Bataille Navale, je n'ai pas connu ce jeu en précédent. En autre, j'ai avancé la capacité de débugger, d'optimiser le code avec la conception MVC.

Enfin, je veux remercier M. RANAIVOSON de me conseiller pendant les cours de travaux pratiques ainsi que M. MATHET pour ses cours magistraux. Merci d'avoir pris le temps de lire ce rapport.

Références

[1] Wikipédia. Bataille navale.
`https://fr.wikipedia.org/wiki/Bataille_navale_(jeu)`.

[2] Jean-Michel DOUDOUX. Le développement d'interfaces graphiques avec swing.
`https://www.jmdoudoux.fr/java/dej/chap-swing.htm`.

[3] Wikipédia. Modèle-vue-contrôleur.
`https://fr.wikipedia.org/wiki/Modèle-vue-contrôleur`.