Université de Caen Normandie
U.F.R. Sciences
Département Informatique
Année 2020-2021

UNIVERSITÉ
CAEN
NORMANDIE

PROJET BATAILLE NAVALE

Réalisé par :
Ben Kabongo Buzangu
Martin Jules
Mamadou Mouctar Bah
Meryam El Boudouti

Introduction

Dans le cadre du cours de Complément POO, il a été demandé aux étudiants de la L2 Informatique de réaliser en groupe un projet informatique de bout en bout afin de parfaire leur savoir-faire et leurs compétences en travaux d'équipes.

Il s'agit ici de réaliser un jeu de **Bataille navale**, en utilisant le langage **Java** et sa bibliothèque intégrée **Swing**. La version du langage utilisée pour le projet est la 1.8.

Le patron d'architecture utilisé pour la conception de l'application est le **MVC** pour **Modèle - Vue - Contrôleur**. Nous avons commencé par implémenter le modèle avec les éléments nécessaires ainsi que la logique du jeu. Ensuite, nous avons conçu la partie vue-contrôleur en rajoutant une interface graphique.

Dans la suite de ce rapport, nous allons vous présenter cette conception et en expliquer les éléments constitutifs.

Chapitre 1

Diagramme des classes

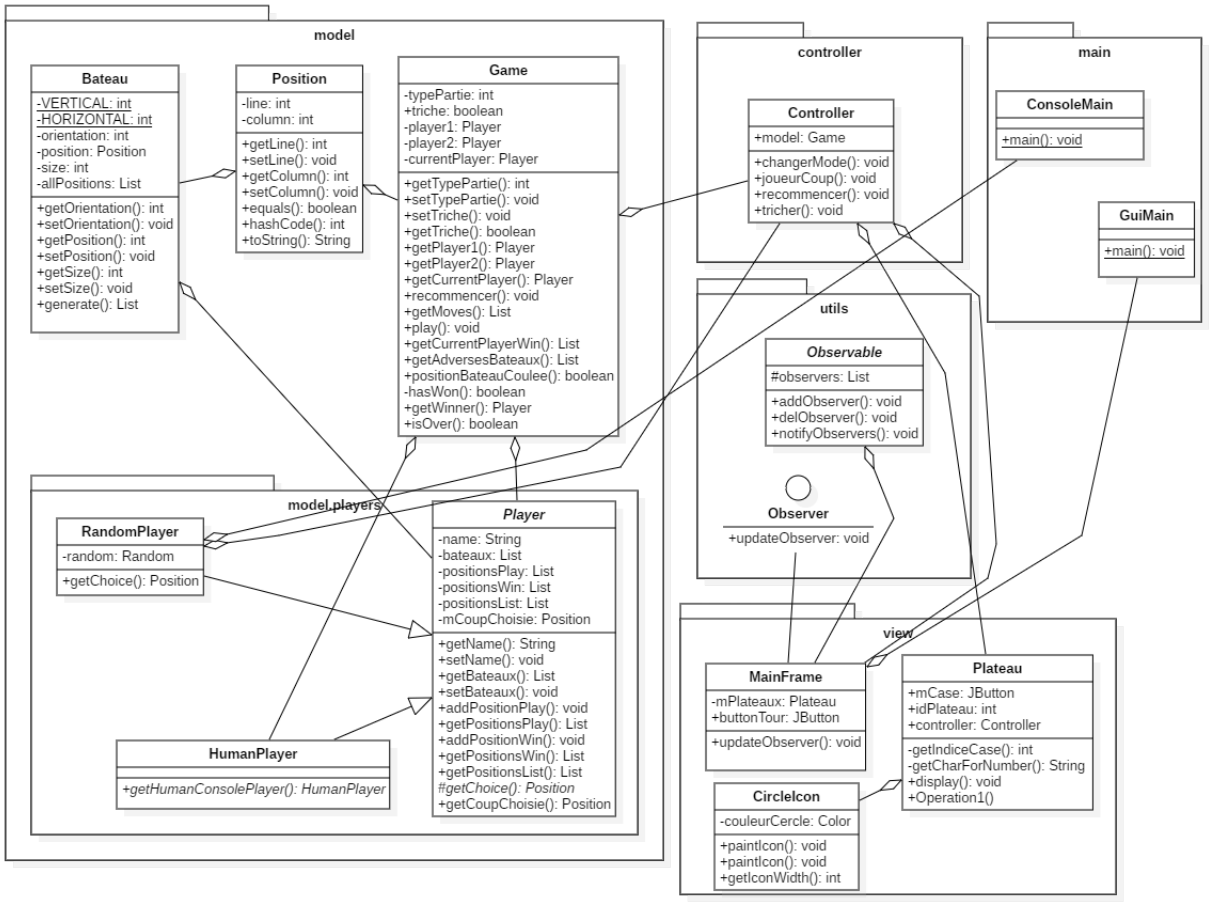

FIGURE 1.1 – Diagramme des classes

Chapitre 2

Modèle de l'application

Le modèle de l'application comprend les éléments constitutifs d'un jeu de bataille navale et la logique de ce dernier.

2.1 Gestion de la position

La notion de position est importante dans un jeu de bataille navale. La flotte des joueurs est donnée par une matrice à deux dimensions. Un joueur choisira donc systématiquement une position dans la flotte adverse sur laquelle tirer. Le tir est soit réussi si un pion du joueur adverse est touché ; soit un échec si aucun pion du joueur adverse n'est touché. Il a donc été nécessaire de mettre en place un mécanisme de gestion de positions.

Nous avons donc créé la classe **model.Position** afin de représenter une position. Une position est la donnée d'un numéro de ligne entre 0 et 9 et d'un numéro de colonne entre 0 et 9. Deux positions sont égales si leurs numéros de ligne et de colonne sont respectivement égales

2.2 Les joueurs

La bataille navale est un jeu dans lequel s'affrontent deux joueurs. Pour chacun, une certaine quantité d'informations importantes est à retenir afin de faire évoluer les parties de jeu.

Nous avons créé le package **model.players** afin de représenter les joueurs. La classe mère de toutes les classes de joueurs est la classe **model.players.Player**.

Chaque joueur sera initialisé par un nom et par une liste de bateaux.

Nous sauvegardons dans trois liste de positions importantes pour chaque joueur.
La liste des positions occupées contient toutes les positions occupées par le joueur dans sa propre flotte.
La liste de positions jouées contient toutes les positions que le joueur a jouée pour la flotte adverse. Cela évite à ce dernier de sélectionner des positions déjà jouées.
La liste des positions gagnées contient toutes les positions auxquelles le joueur a touché le joueur adverse.

Nous nous sommes assurées de représenter dans chacune des listes chaque position une et une seule fois. Ainsi, on conclura qu'un joueur a gagné une partie si la taille de la liste des positions gagnées est égale à la taille de la liste des positions occupées du joueur adverse.

Afin de pouvoir initialiser différents types de jeux, nous avons créé deux types de joueurs : le joueur humain (**model.players.HumanPlayer**) et le joueur aléatoire (**model.players.RandomPlayer**). Le joueur humain choisit lui-même les positions où jouer ; pour le joueur aléatoire, ces positions sont choisies aléatoirement.

2.3 Les bateaux

Dans un jeu de bataille navale, chaque joueur repartit ses bateaux sur différentes positions de la flotte. Nous avons opté pour une repartition aléatoire, pour tous les types de joueurs. Les bateaux possèdent une taille, une orientation et une liste de positions adjacentes. Nous avons implémenté la notion de bateaux dans la classe **model.Bateau**.

2.4 Gestionnaire de partie

Afin de gérer les parties en cours, nous avons créé la classe **model.Game**. Elle prend en paramètre deux joueurs, gère le tour de chacun et effectue les mises à jour adéquates des différentes listes de positions des joueurs au fur et à mesure du jeu.

C'est la classe observable de notre modèle. Elle renseigne ses observateurs des changements qui surgissent au cours d'une partie lancée, du début à la fin de cette dernière.

2.5 Mise en place du pattern observateur

Le pattern observateur fait parti de la démarche MVC. La classe **Game** représente la classe modèle de notre application. Elle hérite de la classe **utils.Observable**. Les classe implémentant l'interface **utils.Observer** peuvent donc observer une partie en cours et être mises au courant des mises à jour. La vue de notre interface graphique implémente cette interface.

2.6 Jeu en console

Nous avons créé une version console du jeu, accesible à partir de **main.ConsoleMain**.

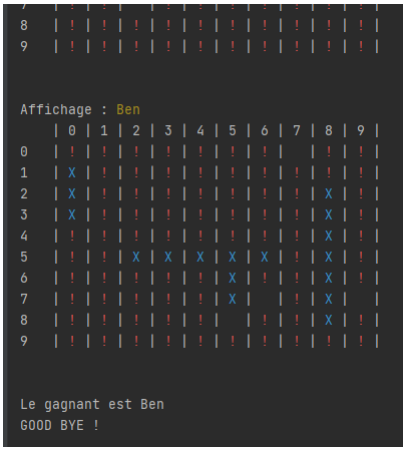

FIGURE 2.1 – Jeu console

Chapitre 3

Interface graphique

L'interface graphique de notre application a été implémentée dans le package **view**. La classe de base **MainFrame** implémente l'interface **utils.Observer**. Au fur à mesure des coups de chaque joueur, l'interface est mise à jour.

Trois modes de jeu ont été implémentés : le mode huamin contre joueur aléatoire, le mode avec deux joueurs aléatoires et le mode avec des joueurs humains.

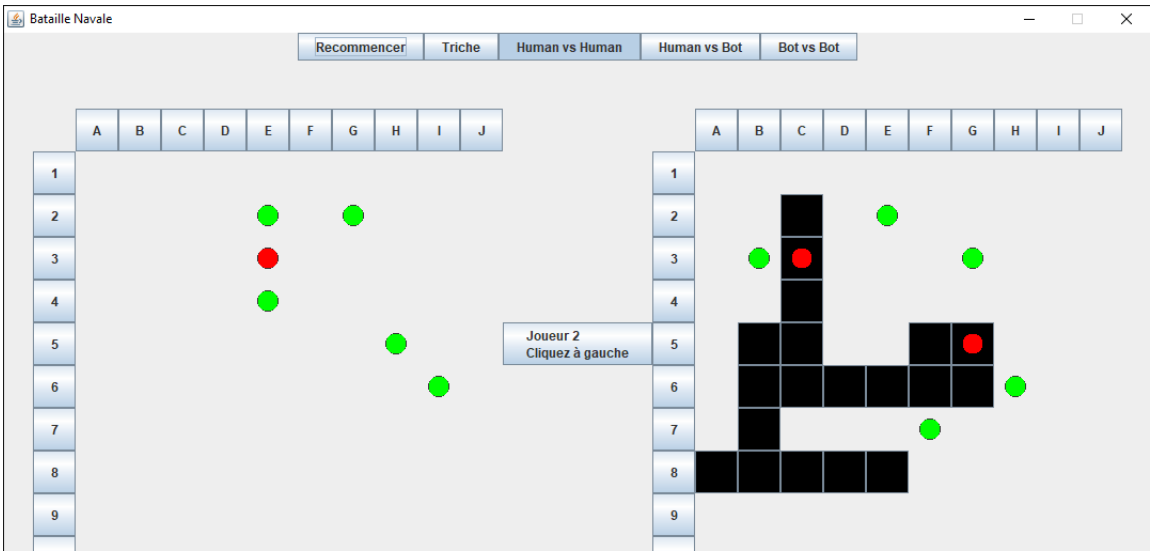

FIGURE 3.1 – Interface graphique