UNIVERSITÉ
CAEN
NORMANDIE

2ème Licence Informatique

Groupe 2B

RAPPORT

Jeu de Bataille Navale

AUTEURS :

AMANDINE CHAMPY

FAICAL SID AHMED RAHMANI

NOLWEN GALLIER

MICHEL KAZADI KALALA

20 Avril 2021

Table des matières

1 Introduction

Dans le cadre de l'unité d'enseignement Complément de Programmation Orientée Objet il nous a été demandé de réaliser une **bataille navale** suivant un modèle MVC (Model View Controller). Cet exercice est pour nous un moyen d'approfondir nos connaissances en Java ainsi que d'en acquérir des nouvelles.

Le modèle MVC est un pattern architectural destiné aux interfaces graphiques. Ce modèle sépare les **données (model)** de **l'interface graphique (view)** et **la logique de contrôle (controller)** c'est à dire la logique concernant les actions de l'utilisateur.

Les règles de la bataille navale sont les suivantes : " La bataille navale est un jeu de société dans lequel deux joueurs doivent placer des « navires » sur une grille tenue secrète et tenter de « toucher » les navires adverses. Si un bateau adverse est impacté, le tir apparaît d'une façon spécifique par rapport à un tir raté .On dit qu'un navire est coulé si chacune de ses cases a été touchées par un coup de l'adversaire. Le gagnant est celui qui parvient à couler tous les navires de l'adversaire avant que tous les siens ne le soient. "

La réalisation de ce projet en MVC se découpe en deux parties :

— L'implémentation du programme en mode console

— L'implémentation de l'interface graphique

2 Organisation du projet

2.1 Répartition des tâches

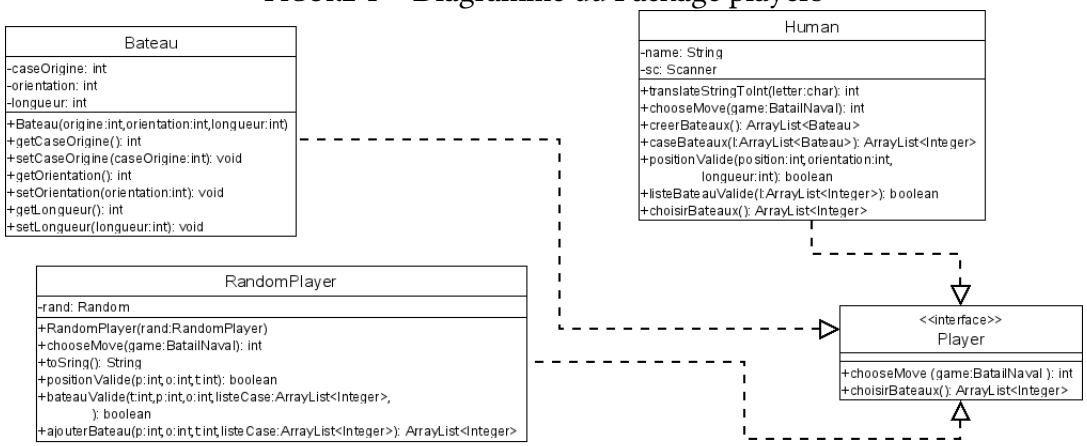
Afin de produire le jeu dans un temps précis nous avons formé un groupe avec Amandine, Nolwen, Michel et Faical. Nous nous sommes réparti en deux groupes de travail. Nolwen et Amandine se sont concentrés sur l'implémentation du jeu en mode console, jouable directement en ligne de commande tandis que Michel et Faical ont entrepris la réalisation de l'interface graphique. L'implémentation en mode console ayant été finie bien avant l'interface graphique, Nolwen et Amandine se sont donc aussi penchés sur la réalisation de l'interface graphique en MVC.

2.2 Architecture du programme

Concernant l'architecture du programme, nous avons opté pour 5 packages contenant le code source :

players : qui comme son nom l'indique contient toutes les classes concernant les types de joueurs.

FIGURE 1 – Diagramme du Package players



— **games** : figure qui contient les classes relatives au bon fonctionnement du jeu.

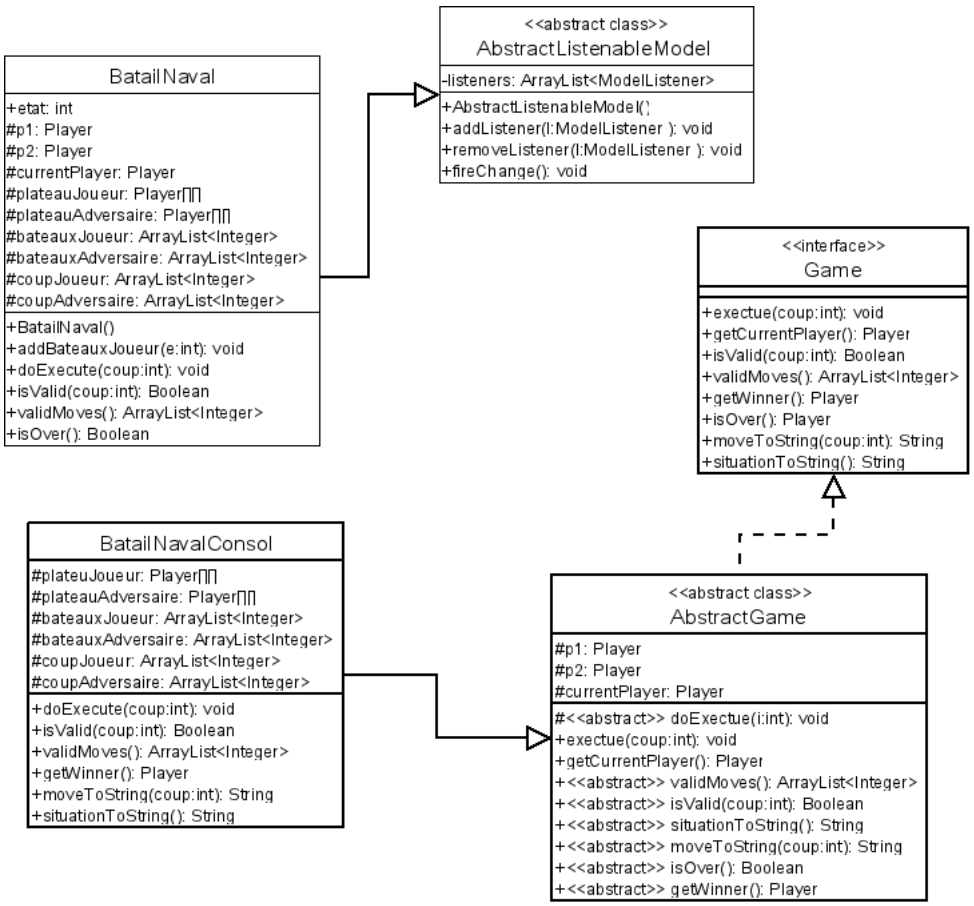
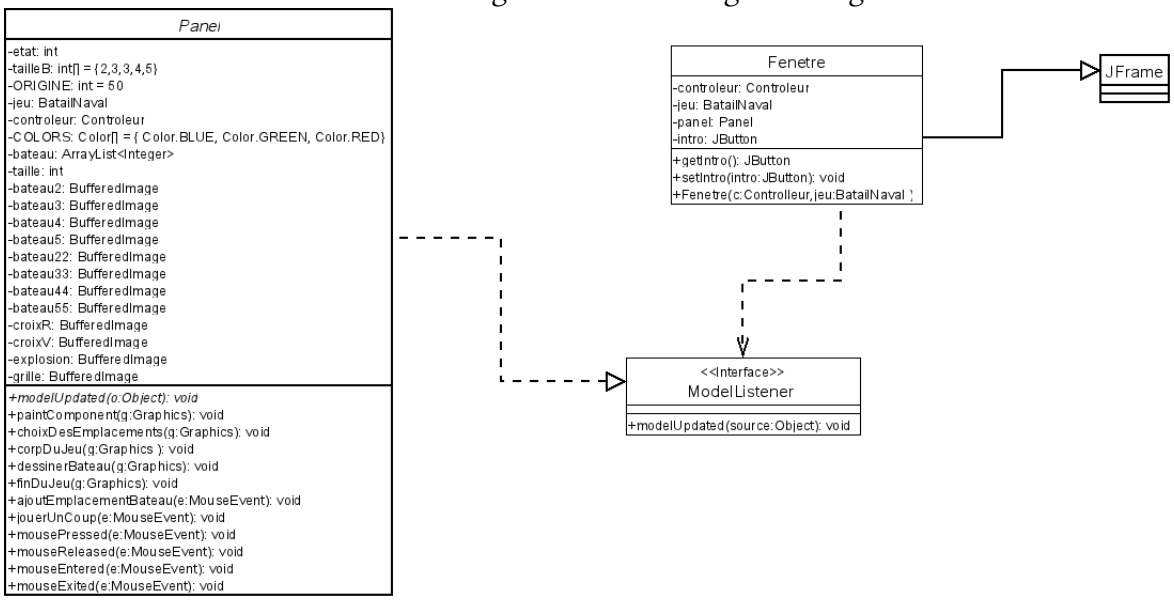


FIGURE 2 – Diagramme du Package games

— **affichage** : qui est constitué de classes qui permettent l'affichage de l'interface graphique. par rapport un MVC le modèle correspond aux packages players et games, la vue au package affichage, et contrôleur au package main.

FIGURE 3 – Diagramme du Package affichage



— **main** : qui contient la classe exécutable.

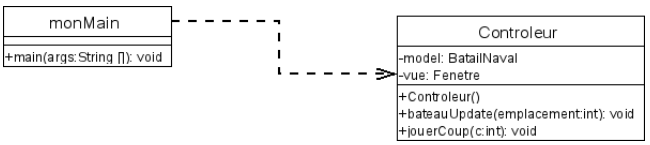
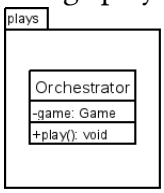


FIGURE 4 – Diagramme du Package main

— **plays** : contenant la classe orchestrant le jeu en mode console.

FIGURE 5 – Diagramme du package plays et de la classe Orchestrator



Notre répertoire contenant notre code source est donc construit de la manière suivante :

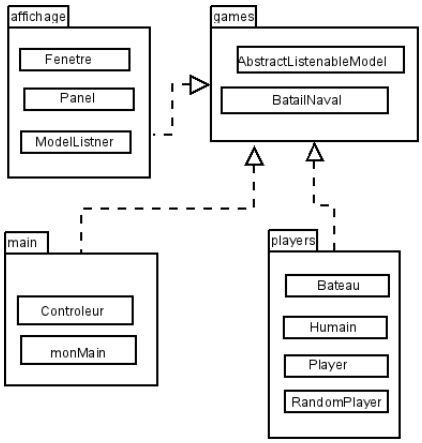


FIGURE 6 – Diagramme des Packages

3 Comment marche notre programme

Quelque soit le mode choisi, console ou interface graphique, notre programme utilise dans l'ensemble les mêmes classes à quelques exceptions près. Nous avons tout de même dû construire deux classes bataille Navale, une pour chaque mode, car notre classe pour le mode console hérite de la classe AbstractGame et la classe pour le mode graphique hérite de la classe AbstractListenable-Model hors une même classe ne peut pas hériter de deux classes. Nous allons donc voir les classes communes aux deux modes avant de parler des différences par la suite.

Pour le fonctionnement du jeu, nous avons créé les méthodes suivantes : $doExecute()$, $isValid()$, $validMoves()$,$isOver()$ et $getWinner()$.

Ces dernières sont toutes indispensable au fonctionnement du jeu et sont utilisées peu importe le mode. La méthode $doExecute()$ se charge d'effectuer un coup choisi par un joueur sur le plateau, $validMoves()$ renvoi la liste des coups valides. Grâce à cette méthode couplée à la méthode $isValid()$ nous pouvons vérifier si un coup choisi par un joueur est valide et peut être joué ou non et pour finir la méthode $isOver()$ vérifie si la partie est terminée. Si la partie est terminée alors la méthode $getWinner()$ nous retourne le joueur qui gagnant.

3.1 Jeu en mode console

Pour le mode console, la plus grosse particularité est que la situation du jeu, c'est à dire le plateau, doit être visible depuis la console. Nous retrouvons donc la méthode $situationToString()$ qui affiche le plateau avec l'emplacement des bateaux du joueur humain et les cases jouées marquées par une croix. Tout ça est visible grâce à une suite de caractères précis :

- ~ = case vide
- 0 = bateau
- x = case jouée
- @ = bateau touché

La deuxième particularité de ce mode est que le joueur humain doit indiqué en début de partie où il veut placer ses bateaux en indiquant l'emplacement de la première case du bateau et son

orientation. Idem lorsque le joueur doit jouer, il doit indiquer la case par son numéro de ligne et par la lettre de sa colonne, comme la bataille Navale traditionnelle.

3.2 Jeu avec interface graphique

Pour le mode avec l'interface graphique, l'expérience de jeu de l'utilisateur est simplifié et plus agréable car tout ce fait visuellement et par un simple clique de la souris. Le jeu se découpe en trois phases que l'on retrouve aussi dans l'interface graphique. La première est le *choix des emplacements des bateaux*, le joueur voit uniquement sa grille s'afficher à l'écran et doit cliquer sur les cases où il souhaite placer ses bateaux. Ensuite la deuxième phase de jeu consiste à trouver les bateaux de l'adversaire avant que ses propres bateaux soient découverts. Il voit donc apparaître deux grilles, celle de gauche représente son plateau avec ses bateaux apparents et celle de droite représente le plateau de l'adversaire. Pour effectuer un coup le joueur doit simplement indiquer la case en cliquant dessus.

La troisième phase ne nécessite pas d'interaction avec le joueur, elle se contente d'afficher le nom du vainqueur de la partie. La principalement différence avec le mode console réside dans le fait de devoir cliquer sur la case que l'on sélectionner au lieu de rentrer dans la console le numéro de la ligne puis la lettre de la colonne de la case que l'on souhaite sélectionner.

3.3 Les similitudes derrières les différences

Que ce soit en mode console ou graphique, le jeu est orchestré par une classe. Pour le mode console ça sera la classe Orchestrator et pour l'interface graphique ça sera la classe Controlleur. Malgré ce que l'on pourrai penser les deux modes sont très similaires dans le fonctionnement. en effet ils sont basés sur le même système car les cases sont associés à un entier dans les deux modes. En mode console, cet entier est calculé grâce à l'entrée du numéro de ligne et la lettre de la colonne par l'utilisateur. En mode console il est calculé avec les coordonnées récupérées lors d'un clique de l'utilisateur sur l'écran. Le reste du jeu ne diffère pas selon le mode.

4 Retour sur le projet

Globalement notre jeu est abouti. En effet il suit les consignes données, il possède un jeu exécutable en console et une interface graphique suivant le schéma MVC.
Nous avons volontairement choisi ce type de graphisme simplement en référence au fait que ce jeu est très vieux et que les premières interfaces de celui ci n'étaient pas plus évoluées que celle-ci.

4.1 Améliorations envisagées

Même si les règles du jeu sont respectées nous aurions pu imaginer des niveaux de difficultés, par exemple un nombre de coup limité pour parvenir à trouver tous les bateaux adverses, ou encore jouer contre le montre.

4.2 Problèmes rencontrés

Nos connaissances sur le modèle MVC étant très basiques, nous avons rencontrés certaines difficultés pour comprendre comment celui-ci devait fonctionner. De plus ne travaillant pas sur les mêmes machines Nolwen et Amandine se sont rendu compte que l'interface graphique n'était pas identique d'une machine à l'autre, nous avons du améliorer l'interface graphique pour que celle-ci ne dépende pas d'un seul type de machine.