RAPPORT PROGRAMMATION OBJET

Bataille-Navale

22008897 : KERMEZIAN AXEL
22009099 : IHANNOUBA NIHAL
22011592 : KRIMI IBRAHIM

4 mai 2021

Table des matières

1 Introduction

Le module de programmation objet permet de mettre en pratique et de perfectionner les connaissances acquises en programmation objets lors du premier semestre. Chaque groupe composé au maximum de quatre personnes , doit réaliser une application implémentant une interface graphique en MVC . Le jeu choisi est le suivant : *L*a bataille navale.

Règles du jeu :

« La bataille navale, appelée aussi touché-coulé, 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. Le gagnant est celui qui parvient à couler tous les navires de l'adversaire avant que tous les siens ne le soient. On dit qu'un navire est coulé si chacune de ses cases a été touchées par un coup de l'adversaire.»

Le projet comporte trois phases de développement importantes :

* Premièrement : Les fonctions implémentant les règles du jeu.
* Deuxièmement : Le placement aléatoire des bateaux des joueurs à chaque nouvelle exécution du jeu.
* Troisièmement : Le développement de l'interface graphique en MVC.

2 Organisation

2.1 Répartition des tâches

Dans un premier temps le groupe c'est concerté pour définir l'implémentation du projet ainsi que la traduction de ses règles en pseudo code.
L'implémentation de l'interface a été négligée lors de cette étape et reportée lorsque le jeu seras implémenté correctement.L'écriture d'une interface graphique est une première pour l'ensemble des membres du groupe.

La concertation étant terminée , le groupe a tout d'abord opté pour la génération d'un tableau de type "Case" qui serons différentes selon la situation du jeu. Ce tableau feras office de flotte pour chaque joueur. La flotte de chaque joueur subis un tirage aléatoire spécifique et non un placement manuel.*(voir page 7)*.

L'intégration des règles ainsi que le déroulement du jeu a été effectué dans une classe spécifique.
Étant composé de trois membres cela a impliqué une répartition des tâches réfléchis.
Krimi Ibrahim et Ihannouba Nihal ont implémentés les différentes classes visant au bon déroulement du tirage aléatoire de la flotte et de sa composition.

Kermezian Axel était chargé d'implémenter les règles du jeu selon le cahier des charges.
Une fois ces tâches achevées, impliquant un jeu fonctionnel, le groupe c'est de nouveau réunis afin d'implémenter l'interface graphique ainsi que ses différentes vues. Cette étape a été effectuée en cohésion par la totalité du groupe.

3 Architecture

3.1 Répartition des classes

La répartition des classes a été étudiée minutieusement.
Quatre Packages au total ont permis de séparer les différentes classes selon leur utilisation.

1. **Le package games** : contient les classes propre au développement de l'interface graphique ainsi que le déroulement du jeu.
   - Mer : implémente les fonctions permettant le respect des règles et le déroulement du jeu ;
   - MerGui : Définis l'interface graphique ;
   - VueMer : Dessine les flottes graphiquement ;
   - VueMerLabel : Affiche l'état de la partie ainsi que la représentation des flottes dans le terminal ;
   - EcouteurModel : Interface permettant de redéfinir l'affichage d'une vue dans les différentes classes implémentant celle-ci ;
   - AbstractModelEcoutable : Classe abstraite agissant sur les objet de type EcouteurModel ;
   - Aleatoire : Définis l'aléa ainsi que son positionnement ;
   - PositionCoord : Classe implémentant des fonctions ciblant les coordonnées d'un clique à la souris.

2. **Le package players** : contient les classes définissant les joueurs.
   - Player : Classe abstraite définissant un joueur ;
   - Humain : Définis les interactions propre à l'humain ;
   - JoueurRandom : Définis les interactions propre au joueur random

3. **Le package cell** : contient les classes composant les flottes des joueurs.
   - Case :Class abstraite définissant une classe ;
   - CaseVide ;
   - CaseBateau ;
   - NonToucher ;
   - Toucher ;
   - Couler ;

4. **Le package main** : Classe Main

3.2 Diagramme

FIGURE 1 – Diagramme Package games

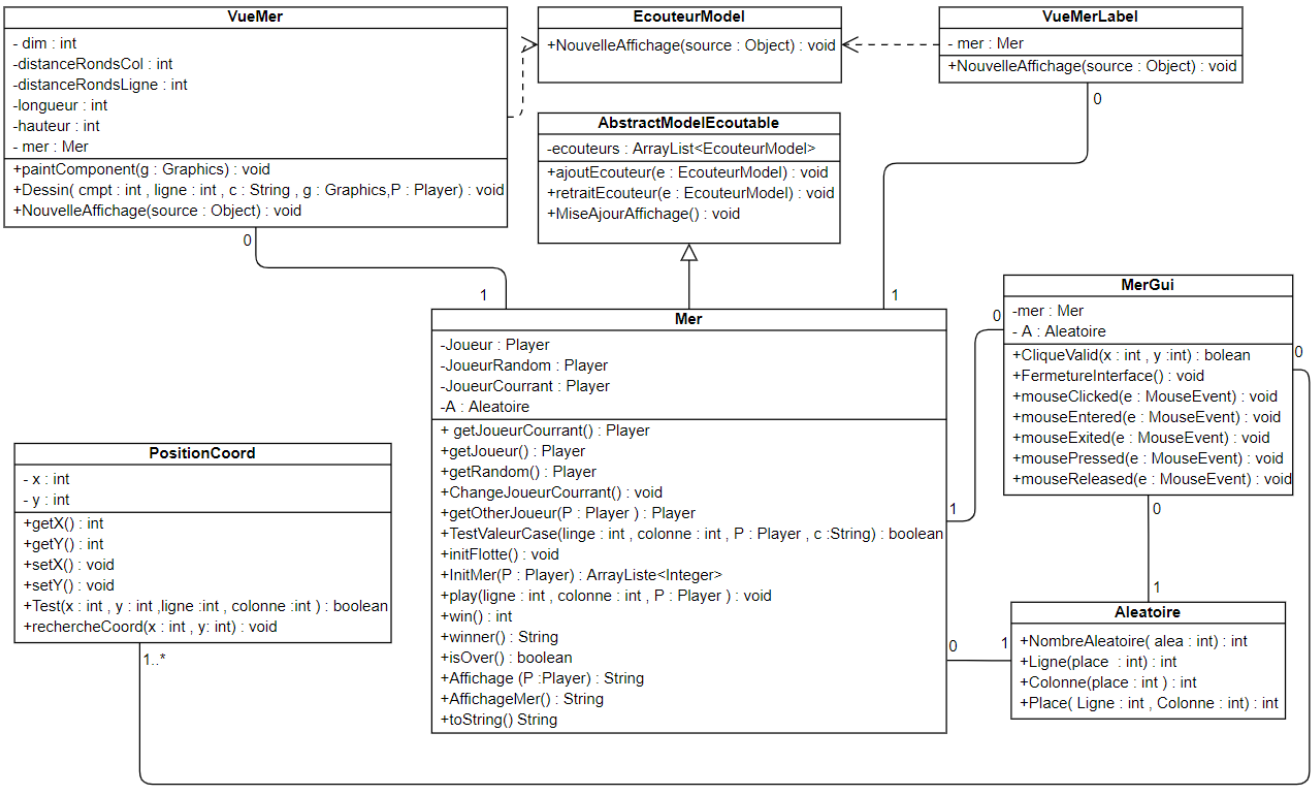


FIGURE 2 – Diagramme Package cell

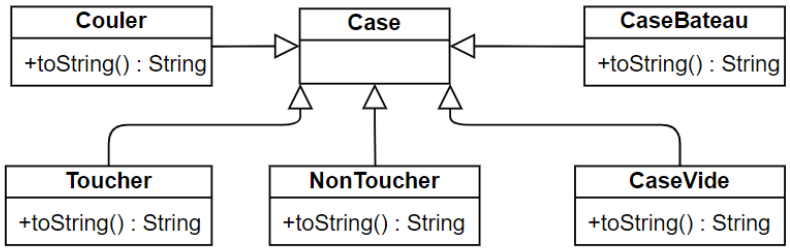


FIGURE 3 – Diagramme Package player

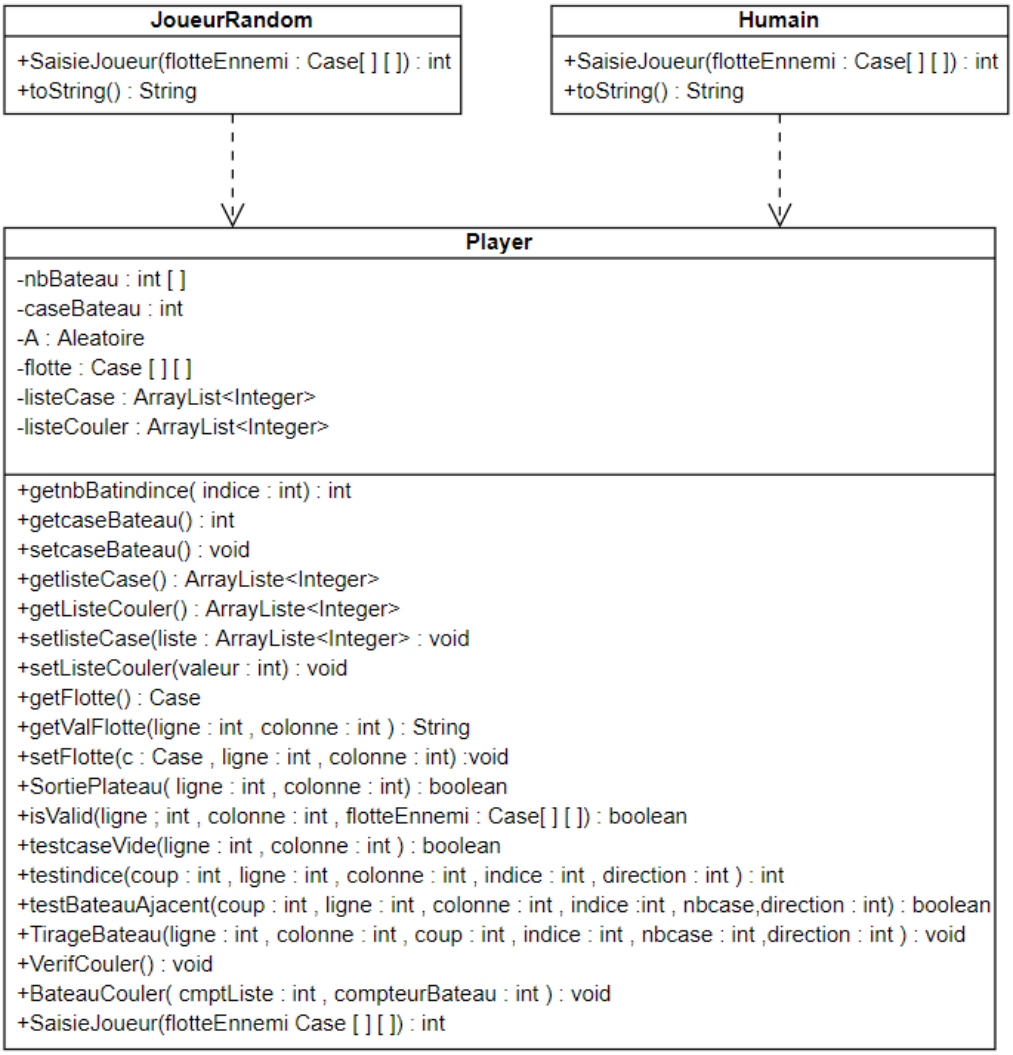
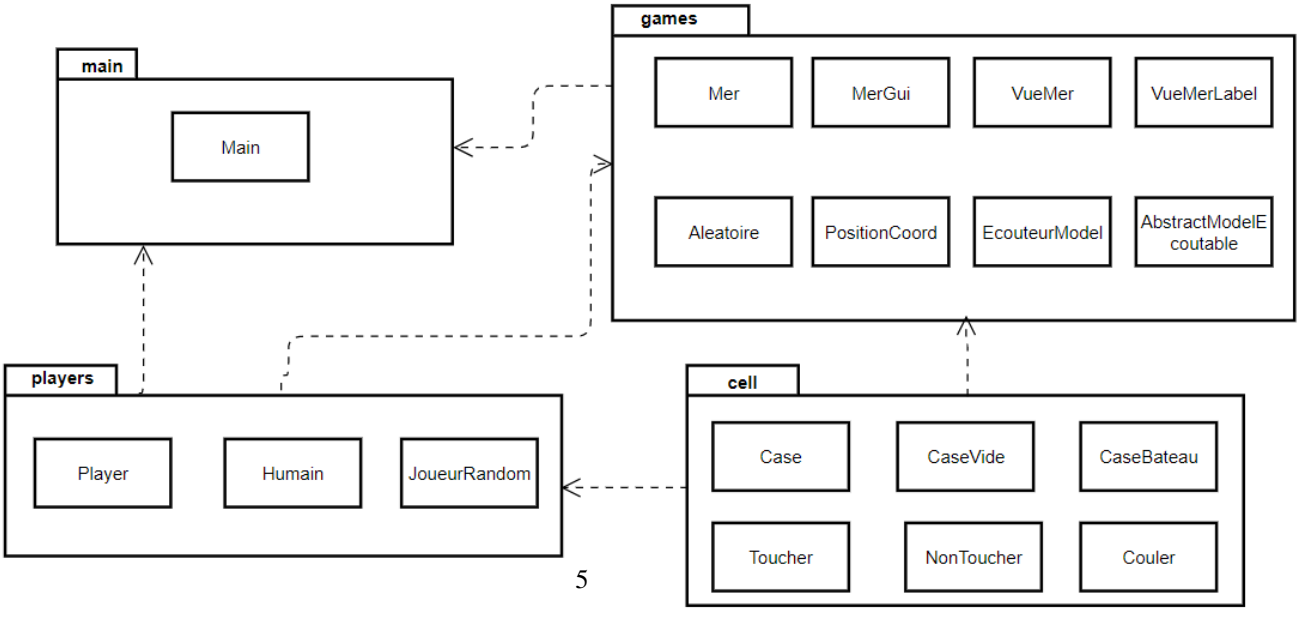


FIGURE 4 – Diagramme des Packages



4 Fonctionnalité

4.1 Tirage Aléatoire de la flotte

La flotte de chaque joueur est soumis à un tirage totalement aléatoire des bateaux inclus dans celle-ci.
La possibilité de tirer une flotte identique à la précédente est faible.Le détails de ce tirage est détaillé plus bas.

Détail de la composition d'une flotte

Une flotte est un tableau de type Case de taille 10x10. Celle-ci contient au total cinq bateaux qui peuvent être positionnés verticalement ou horizontalement.Le choix de cette direction est tirée aléatoirement pour chaque bateau.

Description d'une "Case"

Nom de la Case	**toString()**
Case	(classe abstraite pas de redéfinition)
CaseVide	" "
CaseBateau	" - "
NonToucher	" ! "
Toucher	" X "
Couler	" C "

La redéfinition du toString dans chaque classe étendant la classe Case permet une simplification de l'affichage des flottes.

Case formant une flotte

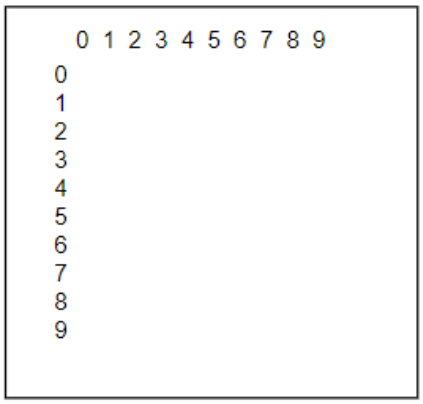
Les cinq bateaux, possède un nombre de "Casebateau" bien définis.La totalité des CaseBateau sur une flotte est de 17 à l'initialisation.Celle-ci permettent la formation des bateaux sur une flotte.
Les Casebateau en question ont des coordonnées définis par une "place" au sein de la flotte. Cette place est définis très simplement : Un tableau de 10x10 contient 100 places. La première place est le nombre 0 ce qui implique la valeur 99 pour la dernière place
La classe aléatoire permet une conversion indice ligne et colonne de la flotte en une place et inversement une place en ligne ainsi que colonne.
Les "places" au sein de la flotte ayant comme valeur la classe CaseBateau sont contenu dans une liste triée selon le bateau ayant le plus petit nombre de CaseBateau.

Position et nombre de place étant des CaseBateau dans la liste

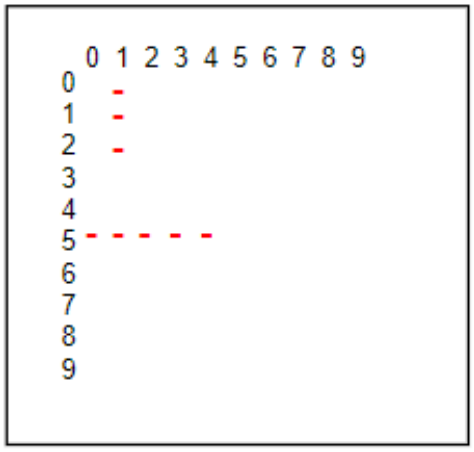
Ordre d'ajout bateau	**Nombre de case sur la flotte**
1er	2
2eme	3
3eme	3
4eme	4
5eme	5

Chaque joueur possède un tableau de constante étant de valeur : "2,3,3,4,5". Ce tableau permet un ajout et parcours spécifique de la listeCase des "places" étant des CaseBateaux tirés aléatoirement.Cela implique une initialisation de la flotte uniquement via cette liste.
Également, lorsqu'un bateau est "coulé", la flotte n'est pas parcourus.Les places des cases de ce bateau coulé sont ajoutés directement dans la listeCouler qui permet la modification des valeurs de ces places au sein de la flotte.

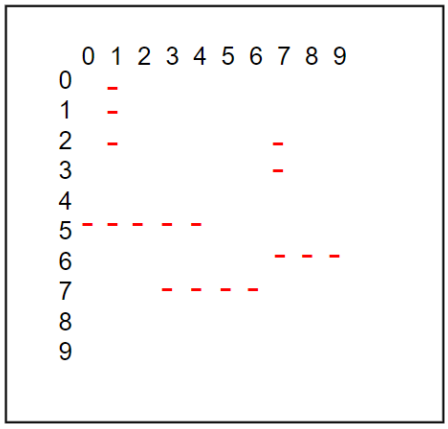
4.1.1 Étape tirage aléatoire d'une flotte et affichage Mer



1. Au commencement, chaque flotte est initialiser avec des Casevide. Les numéros représentent les indices lignes et colonnes.

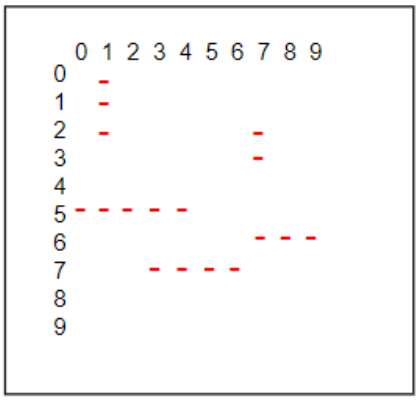


2. Le tirage aléatoire de la direction du bateau est effectué.
Si la direction vaux zéros le bateaux seras horizontal sur le plateau.
Sinon le bateaux seras placé de façon vertical(exemple image).

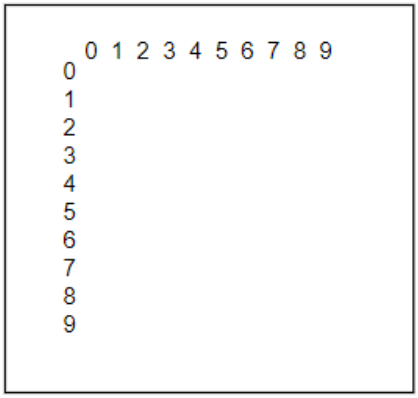


3. Un nombre est alors tiré dans l'intervalle 0 à 99.
Ce nombre formeras la place de départ dans la flotte pour le bateau pris en compte.Selon la direction imposée au préalable , la valeur de l'indice ligne ou colonne de ce nombre seras incrémenté ou décrémenté.
Si sa valeur est inférieur à cinq : l'indice est incrémenté.
Sinon il est décrémenté. Cette opération permet de placé les bateaux de chaque flotte sans collision ni sortie de tableau.

Mer Joueur



Mer Random



4. La Mer subis une redéfinition du toString composée de l'affichage des deux flottes. Les bateaux de la flottes joueur sont visible contrairement aux bateaux du joueur Random.

4.2 Interface Graphique

4.2.1 Introduction

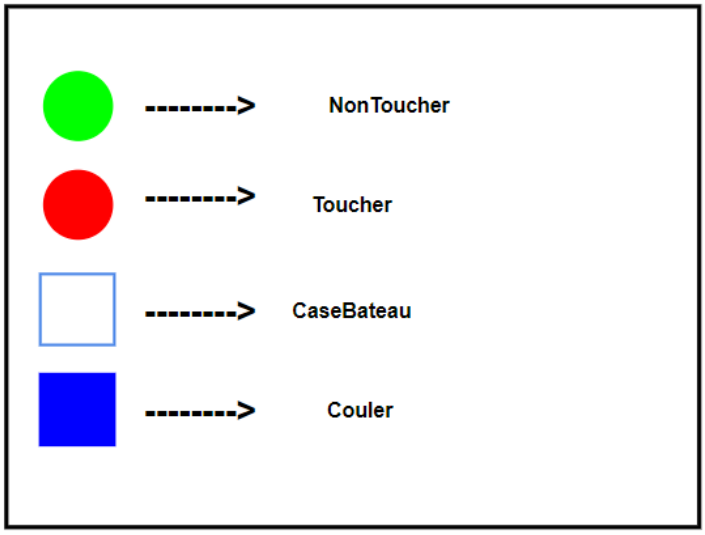
Le but premier du développement de la bataille navale est bien évidement l'implémentation d'une interface graphique en MVC. L'écriture de ces classes a été fortement inspiré des conseils de notre professeur ainsi que des TP effectués.

4.2.2 Les différentes vues

Le programme possède deux vues différentes.Elles implémentent l'interface EcouteurModel contenant la fonction NouvelleAffichage qui est redéfinis pour chaque vue. Cela permet d'appliquer une actualisation des affichages graphiques / terminal.

1. **VueMerLabel** : Permet l'ajout d'un texte stipulant l'état de la partie au sein de l'interface graphique ainsi que dans le terminal.
2. **VueMer** : Effectue la représentation des flottes dans l'interface graphique. A l'aide de graphics2D les cases des flottes sont représentées de la façon suivante :

FIGURE 5 – Représentation graphique



4.2.3 Mise à jour des vues

L'actualisation des vues est primordiale au bon déroulement d'une partie.Elles doivent être mise à jours lorsqu'un joueur effectue un tir.Pour ce faire la classe AbstractModelEcoutable contient les différentes vues dans une liste.A l'aide d'une méthode , les vues de la liste sont extraites et exécute chacune la fonction NouvelleAffichage. Cette méthode est appelée à la suite de chaque tir effectué dans le play au sein de la classe Mer.
L'affichage dans le terminal et de l'interface graphique est donc automatiquement actualisé lors d'un tir.

4.2.4 La Fenêtre de l'interface graphique : MerGui

La classe MerGui est la fenêtre graphique du jeu.Cette classe est une extension de la JFrame , dont toutes les vues codées lui sont ajoutées.
L'étape primordiale est évidement l'implémentation du listener et de son contrôleur. La souris étant le moyen d'interagir exigé ,MerGui a donc implémentée l'interface MouseListener pour redéfinir particulièrement la méthode MouseClicked.
La VueMer est écoutée , lorsqu'un clique de souris est effectué , sa validité est testée en autorisant uniquement un clique sur la flotte du Joueur Random. Une fois le clique effectué impliquant un tir valide, celui-ci est exécuté et notifie le changement de vue.

5 Manuel d'utilisation

Architecture du dossier principale :

dist : permet de contenir les fichiers .class.
src : contient les packages ainsi que les différentes classes.
doc : contient la javadoc "index.html" et ses différents modules nécessaire.
rapport : contient le rapport sous format pdf.
compilation.sh : Script permettant l'exécution du programme.
sources.txt : contient les sources utile au script d'exécution.

Compilation du projet :

Ouvrir un terminal dans le dossier principale contenant src ,dist,doc et compilation.sh ;
Lancer le script de compilation avec la commande : ./compilation.sh

L'exécution va générer les flottes aléatoirement , qui serons affichées sur la console et également dans l'interface graphique en mentionnant le nom de chacune.

Affichage des flottes dans le Terminal :

Bateau(Joueur uniquement) = -
Toucher = X
NonToucher = !
Case vide = (caractère espace).
Couler = C

Effectuer un tir :

Tout au long de la partie un tir peut être effectué via le clavier au sein du terminal ou à l'aide de la souris dans l'interface graphique.

Choix saisie dans le terminal :
Le choix de la saisie de la ligne et de la colonne du tir serons demandés. Si les choix ne sont pas valides , une nouvelle demande de saisie est exécutée.

Choix clique interface graphique :
La possibilité d'effectuer un tir directement sur l'interface graphique est possible uniquement sur la flotte du JoueurRandom.

Le tir étant effectué l'affichage graphique/terminal sont actualisés ainsi que l'état de la partie. Lorsqu'un joueur possède uniquement des bateaux coulés, l'application se ferme automatiquement et renvoi l'affichage du gagnant dans le terminal.

6 Conclusion

Ce projet nous a permis d'utiliser la connaissance apportée par l'université sur le langage Java pour implémenter un jeu et ses règles en autonomie. L'étude et l'écriture d'une interface graphique nous a permis d'aborder pour la première fois ce vaste domaine essentiel et très intéressant.Nous avons acquis de nouvelle connaissance sur un sujet qui nous était totalement inconnu.

Dans l'ensemble le cahier des charges a été respecté :
Le moteur du jeu , le tirage d'une flotte totalement aléatoire ainsi que l'interface graphique en MVC sont implémentés correctement. Cela dis quelques optimisations sont possibles concernant l'interface graphique.

6.1 Optimisation possible

De nombreuses optimisations sont possibles pour le jeu de la bataille Navale.Cela dis, le manque de temps a complètement résilier la possibilité de les implémenter.

1. **Placement des bateaux du Joueur graphiquement :** l'ajout de méthode permettant de placer les bateaux graphiquement à l'aide de la souris aurait été une option agréable pour le joueur exécutant le programme.

2. **L'implémentation d'un joueur Random plus performant :** Le joueur Random ayant des tirs totalement aléatoires, pourrait lors d'un tir sur un bateau ciblé les cases adjacentes.Cependant surcharger le code d'avantage n'était pas désiré par la totalité du groupe. L'objectif ciblé était clairement de développer correctement l'interface graphique en MVC , ce qui a retenu toute notre attention.