UNIVERSITÉ
CAEN
NORMANDIE

RAPPORT
COMPLEMENT POO

Projet :
BATAILLE NAVALE

Réalisé par :
Adrien Knell
Daniel Murray
Pierre Royer
Elodie Ratovoherinjanahary

Licence 2 Informatique

Année 2020-2021

Dans le cadre du projet de Complément à la POO il nous a été demandé de réaliser et de programmer, par groupe de 4, un jeu respectant le modèle MVC à l'aide du langage JAVA. Le sujet imposé était de réaliser un jeu de bataille naval dans lequel l'utilisateur joue contre un random. L'objectif principal de ce projet était de mettre en application nos connaissances en programmation orientée objet acquises au premier semestre, d'apprendre à s'organiser en groupe tout en réalisant sa part de travaille en autonomie, mais aussi de découvrir et de maîtriser de nouvelles notions, en particulier le modèle MVC. Nous avions environ 1 mois et demi, avec chaque semaine 1h30 de TP encadrés, pour la réalisation de ce projet.

Table des matières

1 Présentation et Organisation **3**
1.1 Description du projet 3
1.2 Présentation de notre jeu et fonctionnalités implémentées. 3

2 Architecture du projet **5**
2.1 Arborescence du projet : MVC 5
2.2 Chaîne de traitement 9

3 Elements techniques **9**
3.1 La programmation orientée objet 9
3.2 Le modèle MVC 9
3.3 Bibliothèque graphique : Java SWING 10

4 Conclusion **11**

5 Annexe **12**
5.1 Contacts 12
5.2 Sources 12

1 Présentation et Organisation

La première étape du projet était de former un groupe de 3 ou 4 élèves. Dans notre cas, notre groupe s'est rapidement formé et est composé de Élodie RATOVOHERINJANAHARY, Pierre ROYER, Daniel MURRAY et Adrien KNELL. Le jeu que nous devions effectuer était donc un jeu de bataille navale.

1.1 Description du projet

La bataille navale est, à la base, un jeu de société crée en France durant la Première Guerre mondiale et qui est apparu pour la première fois sous forme électronique en 1977. Le but du jeu est simple : deux joueurs possèdent chacun une grille tenue secrète sur lesquelles des bateaux sont placés. Chacun leur tour, les deux joueurs tirent sur une case de la grille adverse afin de toucher et de faire couler les bateaux de l'adversaire. Un bateau est dit `coulé` lorsque chacune des cases du bateau ont été touché. Le gagnant est celui qui parvient à couler tous les navires de l'adversaire avant que tous les siens ne le soient. L'énoncé du sujet nous demandait de créer un jeu de type `Bataille navale` en respectant toutes les règles citées précédemment, faisant jouer l'utilisateur contre un générateur aléatoire, avec une interface graphique effectuée en Java SWING et en utilisant un modèle de structure MVC.

1.2 Présentation de notre jeu et fonctionnalités implémentées.

Notre programme est un jeu de type Bataille Navale respectant toutes les règles d'un jeu Bataille Navale classique et comportant un style graphique assez basique mais intuitif. L'utilisateur joue face à un générateur aléatoire qui tir aléatoirement une case de la grille.

Voici la présentation étape par étape de notre jeu :

— Au lancement du programme, l'utilisateur est invité à inscrire son nom

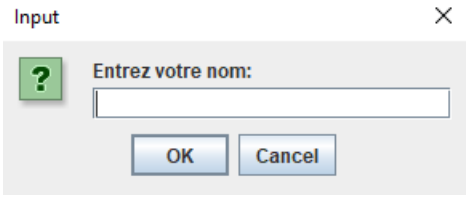

FIGURE 1 – Entrez le nom

— Comme demandé par notre professeur, si vous voulez voir un exemple de coups en ligne de commande, entrez 0 dans la console.

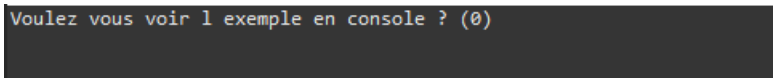

FIGURE 2 – Choix Console

— Sinon, tout de suite après, la partie se lance et les bateaux de l'utilisateur ainsi que ceux de son adversaire sont placés de manière aléatoire. L'utilisateur peut voir l'emplacement de ses propres bateaux, mais bien sûr il n'a aucune information sur les bateaux de son adversaire.

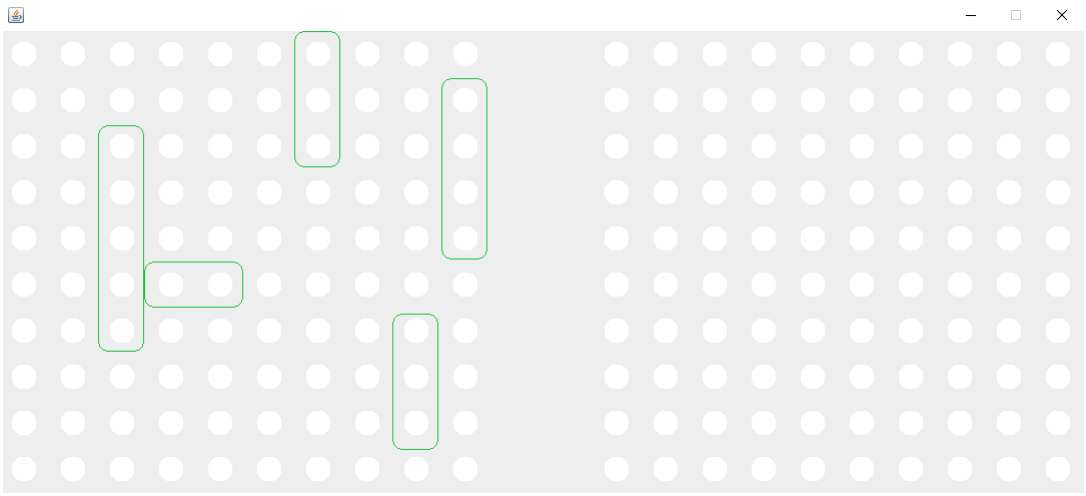

FIGURE 3 – Fenêtre de jeu

— C'est toujours l'utilisateur qui commence par effectuer le premier tir. Pour tirer, l'utilisateur doit simplement cliquer sur l'emplacement de son choix sur la grille de son adversaire. Une fois le tire effectué, la case s'affiche en bleu ou en rouge : bleu pour dire que le tir est arrivé dans l'eau, rouge pour dire qu'un bateau a été touché. Dès que le l'utilisateur tir, le générateur aléatoire tir à son tour et s'affiche également en bleu ou en rouge. Si l'utilisateur tire sur une case qu'il a déjà touché, le tir est pris en compte et lui fait perdre un tour, comme dans les règles officielles de bataille navale.

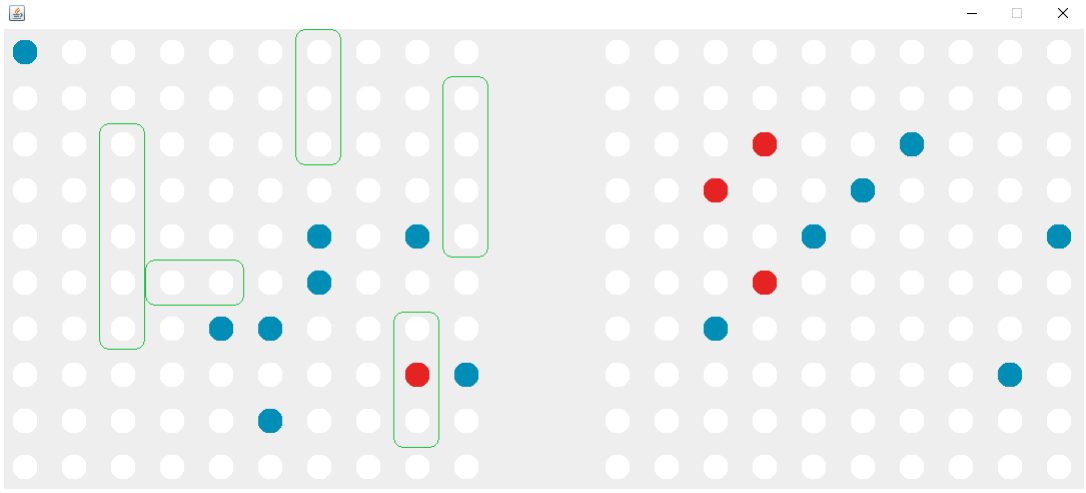

FIGURE 4 – Déroulement d'une partie

— Lorsque l'utilisateur `coule` un bateau de son adversaire, alors automatiquement le bateau s'affiche.

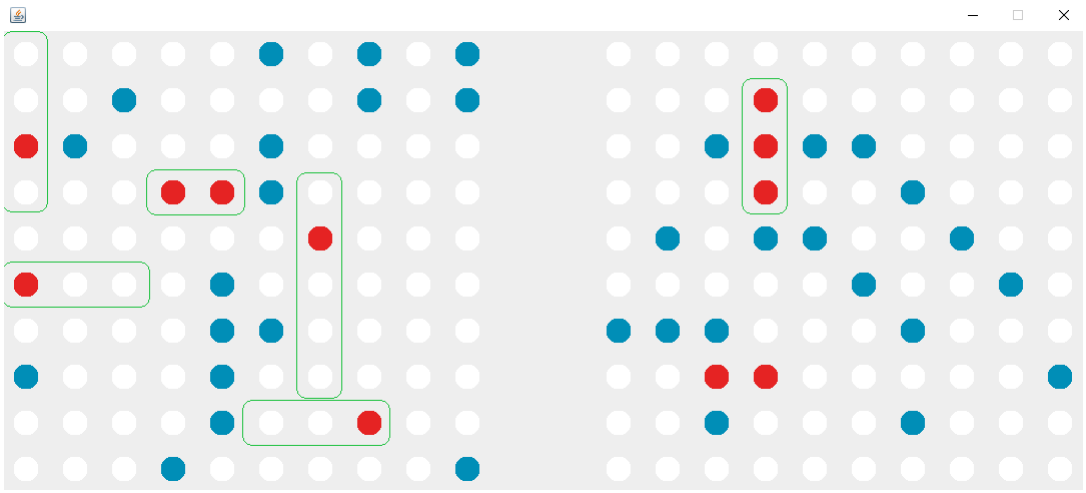

FIGURE 5 – Bateau touché

— La partie se termine lorsque l'un des deux joueurs a coulé tous les bateaux de son adversaire et on affiche le vainqueur.

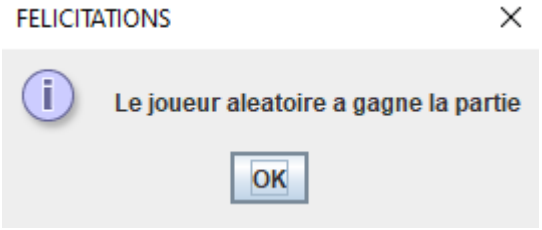

FIGURE 6 – Fin d'une partie

Nous vous avons présenté le sujet de notre projet ainsi que son fonctionnement. Cependant, vous verrez dans la suite les pistes et les chemins empruntés pour concevoir ce projet.

2 Architecture du projet

2.1 Arborescence du projet : MVC

En vue d'une meilleur compréhension de l'arborescence de notre projet, voici un schéma UML qui récapitule l'organisation des packages, des classes, des méthodes et des variables présents dans notre programme.

Tout d'abord, notre projet s'organise en 4 packages distincts :
— Un package `lanceur` contenant le main qui permet le lancement du jeu.

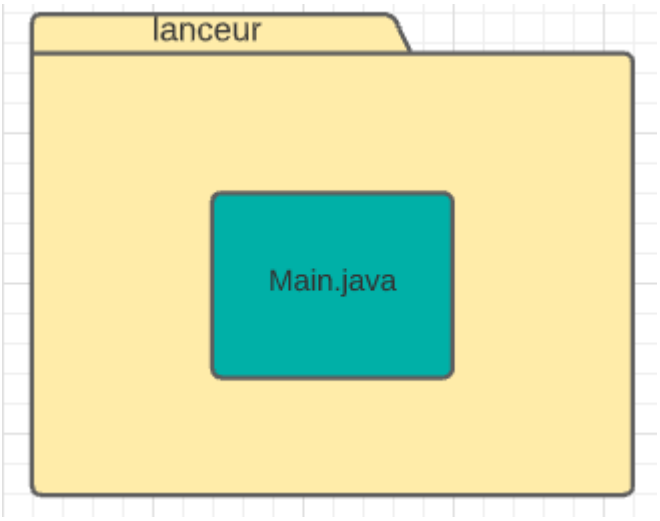

FIGURE 7 – Package Lanceur

— Un package `model` contenant la classe `ModelContent` qui stock toutes les données du jeu, c'est à dire la grille de chaque joueur, l'emplacement des bateaux, les différents tirs effectués, etc ... De plus, dans ce package nous y retrouvons des classes permettant de représenter chaque objet utilisé dans le jeu : la classe `Boat` pour les bateaux, la classe `Bomb` pour les tirs, la classe `BattleField` pour les grilles, la classe `Case` pour les différentes cases des grilles, la classe `Coords` afin de représenter la position des cases des des bateaux et enfin une classe `Player` pour représenter les joueurs. Nous y retrouvons également une classe abstraite `AbstractListenableModel` (fonctionnant avec `ModelListener` de la classe `vue` , voir ci-dessous) permettant de rendre écoutable certains objets afin qu'au moindre changement de l'objet son affichage graphique se mette à jour.

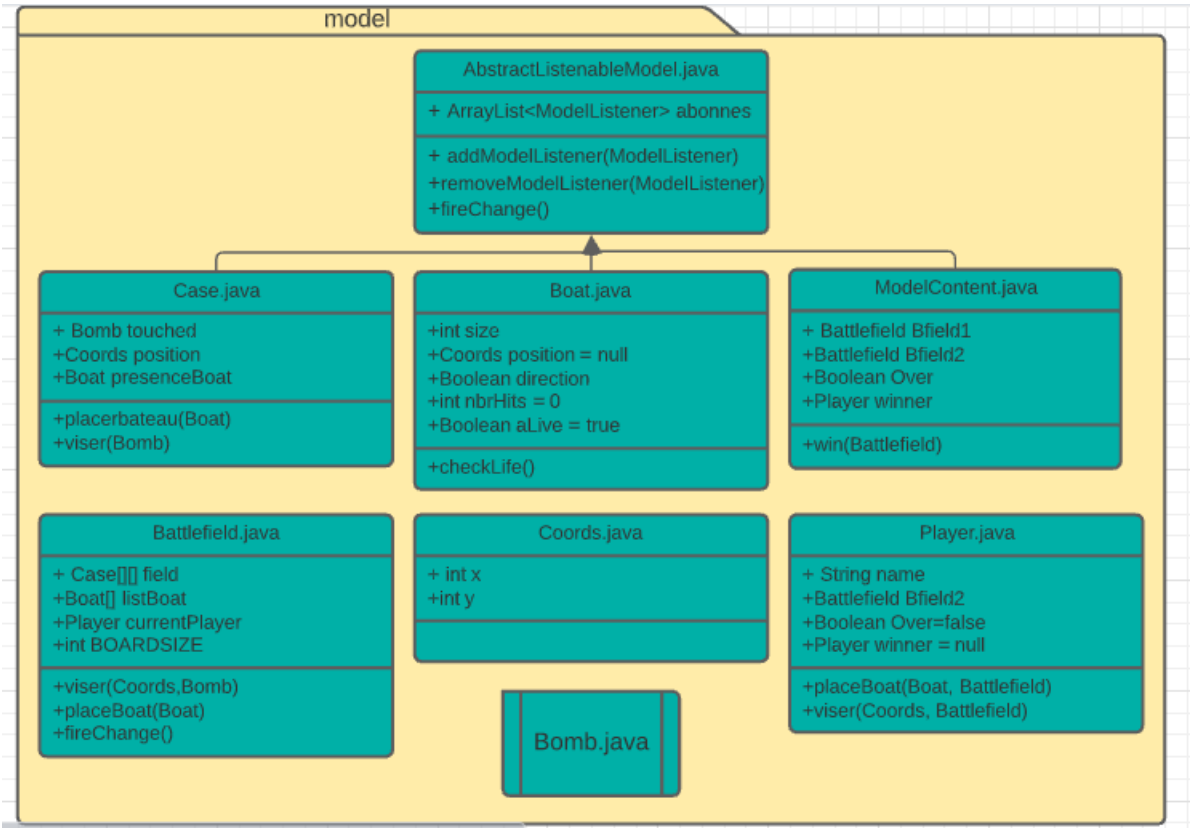

FIGURE 8 – Package Modèle

— Un package `vue` chargé de l'affichage graphique des deux grilles, des tirs et des bateaux. Tout d'abord, ce package contient une classe `Background` qui est un JFrame sur lequel nous affichons les deux grilles des joueurs à l'aide des classes `MyField` et `OpponentField` qui sont des JPanel représentant respectivement la grille de l'utilisateur et la grille du générateur aléatoire. Ensuite, sur ces deux JPanel nous affichons des `CasePanel` qui représentent les cases des champs de batailles. Remarquez que `OpponentCasePanel` hérite de `CasePanel` car l'affichage des cases du générateur aléatoire est différent de celui de l'utilisateur. De plus, nous retrouvons une interface `ModelListener` permettant de mettre à jour des éléments graphiques lorsqu'on lui demande.

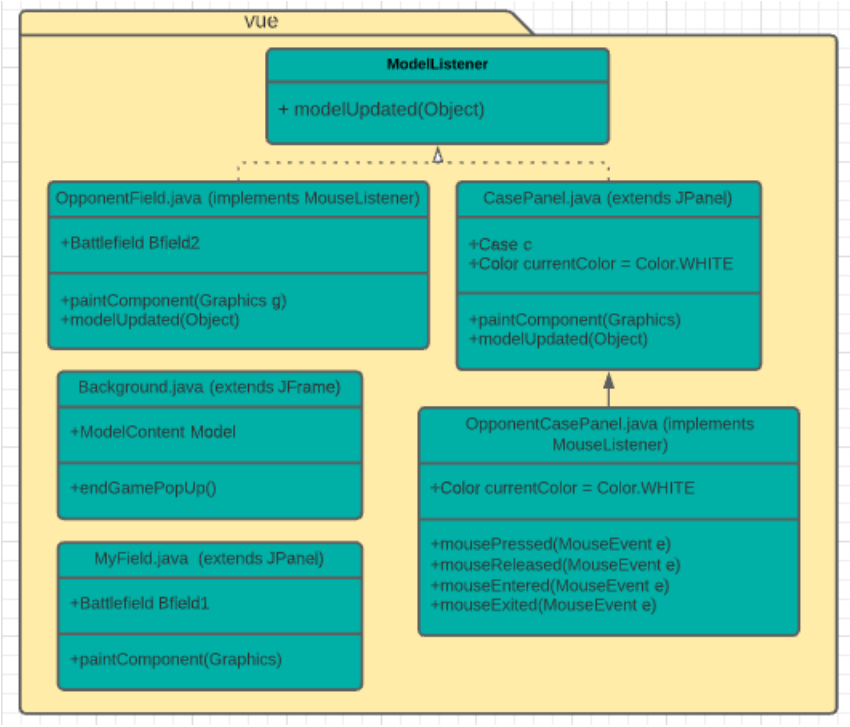

FIGURE 9 – Package Vue

— Un package `controller` permettant en quelque sorte de lier les deux packages `model` et `vue` . En effet, lorsque l'utilisateur clique sur une case afin de tirer sur la grille adverse, le package `vue` capte le clique et son emplacement et avertit la package `controller` qui met à jour la donné dans le `model`.

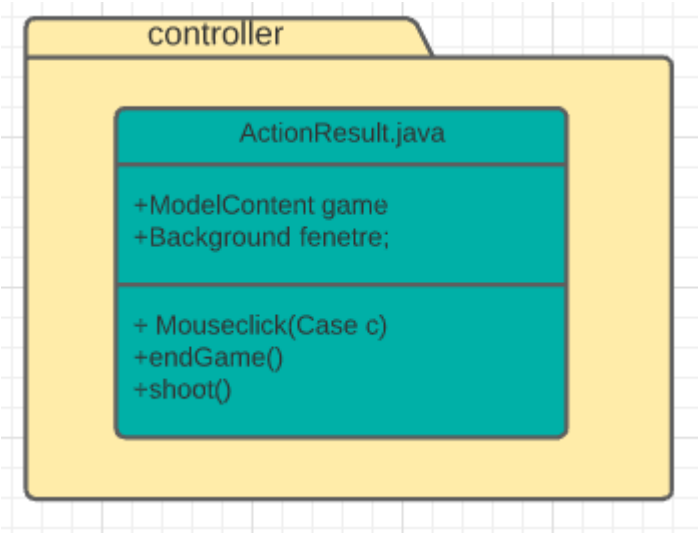

FIGURE 10 – Package Controlleur

2.2 Chaîne de traitement

Nous avons vu précédemment le contenu et l'utilité de chaque classe et package nécessaire au bon fonctionnement de notre jeu. C'est pourquoi, dans cette partie nous allons décrire étape par étape son fonctionnement et plus précisément sa chaîne de traitement.

* Tout d'abord, la première étape du programme consiste à créer une instance de `ActionResults` qui va créer une instance de `ModelContent` et la fenêtre du jeu.
* A partir de là, le jeu est totalement indépendant. En effet, à chaque clique de l'utilisateur sur la grille de son adversaire, le package `vue` le capte, prévient le `controller` qui modifie le `model` . Ensuite, les classes `Case` et `Boat` héritants de `AbstractListenableModel` préviennent la `vue` pour qu'elle se mette à jour et affiche les nouvelles informations.
* A chaque clique de souris, nous vérifions s'il reste encore des bateaux en vie sur les deux plateaux. Si un des deux joueurs n'a plus de bateau, nous affichons le pop-up de fin de partie

3 Elements techniques

La conception de ce projet nous a permis de revoir certaines notions techniques déjà acquises mais aussi et surtout de mettre en application de nouvelles notions vues en cours. En effet, pour réaliser ce projet nous avons utilisé les principes de bases de la programmation orientée objet, mis en place une structure MVC et utilisé une bibliothèque graphique nouvelle.

3.1 La programmation orientée objet

Durant notre année de L1 informatique nous étions habitué à de la programmation basique sans POO avec du HTML, JavaScript, Langage C, Python,... Or, ce projet nous a montré et convaincu que la programmation orientée objet était un système très puissant et bien adapté à la conception de jeu mais aussi de diverses logiciels. En effet, l'utilisation de classe, d'interface et de package permet une organisation et une arborescence claire des projets.

3.2 Le modèle MVC

L'objectif et la notion technique la plus importante de ce projet était l'élaboration d'un modèle MVC organisé et fonctionnel. En effet, ce modèle est très adapté à la conception de jeu car il permet de séparer trois éléments majeurs d'un programme : Le stockage des données, l'affichage graphique des données et les actions à effectuer lorsque l'utilisateur interagit avec le programme. Cela permet, en plus d'avoir une arborescence claire, d'avoir un code facilement modulable, modifiable et compréhensible. Pour notre groupe, la mise en place du modèle MVC a été assez naturel car nous avions tous bien compris son fonctionnement grâce aux cours et aux travaux pratiques de `Complément à la POO` .

3.3 Bibliothèque graphique : Java SWING

Pour concevoir et coder l'interface graphique de notre programme nous avons fait appel à la bibliothèque graphique Java Swing.

Cette bibliothèque, facile à prendre en main, nous a permis d'afficher nos différents objets, bateaux, grilles, bombes, cases, ... Pour cela, Java Swing propose un système de JFrame, JPanel ou encore JLabel afin d'afficher nos différents éléments de manière hiérarchique. De plus, Java Swing permet de capter les événements de l'utilisateur, notamment le clique, ce qui lui permet de tirer sur la case qu'il souhaite. De plus, cette bibliothèque se marie parfaitement avec l'utilisation de `AbstractListenableModel` et `ModelListener` qui assure la mise à jour de l'interface graphique.

4 Conclusion

Ce projet nous a permis de mettre en application nos connaissances en java acquises au cours du premier et deuxième semestre. Nous avons découvert et pu approfondir la maîtrise de plusieurs notions : le modèle MVC, la bibliothèques JavaSwing. De plus, nous avons acquis des méthodes de travail de groupe essentielles à l'avancement d'un projet.

Cependant, notre projet n'est pas totalement fini, faute de temps. En effet, nous aimerions implémenter des éléments audio et visuels afin d'améliorer l'experience de l'utilisateur : ajouter une musique de fond ainsi que des bruitages lorsque l'utilisateur tir ou encore implémenter des graphismes plus réalistes.

5 Annexe

5.1 Contacts

- ◇ KNELL Adrien : 06 95 00 26 43
- ◇ MURRAY Daniel : 07 52 05 17 13
- ◇ ROYER Pierre : 06 14 03 57 58
- ◇ RATOVOHERINJANAHARY Elodie : 06 24 34 99 08

5.2 Sources

- — Bataille Navale - Wikipédia
- — Java Swing - Wikipedia
- — Modèle MVC - Wikipedia
- — Site developpez
- — Openclassroom