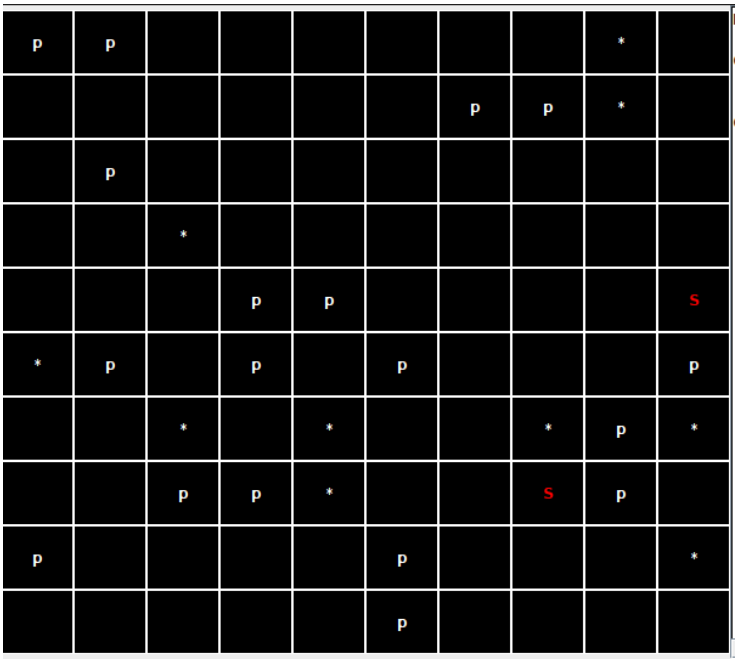
- KERMEZIAN AXEL : 22008897
- KRIMI IBRAHIM : 22011592
- IHANNOUBA NIHAL : 22009099

UNIVERSITÉ CAEN NORMANDIE

Méthodes de conception

Jeu de combat



Sommaire

1.Introduction

Le module de méthodes de conception permet de mettre en pratique et de perfectionner les connaissances acquises en programmation objets et l'utilisation de différents pattern avec le langage Java.
Chaque groupe composé au maximum de quatre personnes, doit réaliser une application MVC qui doit être adaptable à différents supports d'utilisation.

Le jeu choisi est un jeu de combat opposant différents combattants au sein d'un plateau.
Ces combattants possèdent également des armes, bombes, mines …
Les combattants présents sur le plateau doivent alors s'affronter dans le but d'être le dernier présent sur le plateau.

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 et ses règles seront implémentées.

La concertation étant terminée, le groupe a tout d'abord opté pour la représentation d'un plateau sous forme de deux Map :

- Une map contenant les combattants ainsi que leur coordonnée pour clef.
- Une map contenant les éléments du plateau (Bombe,Mine,Vide,Pastille) ainsi que leur coordonnée pour clef.

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 étant propre aux éléments et combattants.
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 réuni afin d'implémenter les différents patterns et la mise en place de l'interface graphique.

3.Fonctionnalités

Détail de la composition du plateau de jeu

Le plateau de jeu est une grille paramétrable en modifiant les constantes au sein de la classe Constante. Le plateau doit avoir une taille supérieure à deux strictement ainsi que posséder le nombre d'emplacement nécessaire pour accueillir les combattants (deux au minimum).
Ce plateau est composé de deux map :

1. Une map contenant les différents combattants ayant pour clef leur coordonné.
2. Une map contenant les différents éléments du plateau ayant pour clef leur coordonné.

Descriptif des combattants :

Nom	**Représentation plateau (toString)**	**Caractéristiques**
Tank	T	Le Tank inflige des dégâts lourds cependant son arme est de courte portée et ses déplacements ont un cout significatif. L'activation du bouclier est privilégiée car son cout est relativement faible
Elite	E	Le joueur élite à des déplacements léger à faible cout son arme a une portée très avancée et inflige beaucoup de dégât, cependant l'utilisation du bouclier à un cout conséquent.
Soldat	S	Le soldat est un juste milieu entre le tank et le joueur d'élite.
RandomCombattant	R	Le joueur Random peut se voir attribué des valeurs aléatoires à toute ses caractéristiques.

Descriptif des éléments :

Nom	**Représentation plateau**
Bombe	B
Mine	M
Mur	*
Vide	
Pastille	p

Détail des différentes stratégies :

Le plateau de jeu et les combattants présent sur le plateau peuvent se voir appliquer une stratégie.

Les combattants

Les combattants ont deux stratégies possibles, celle choisi est modifiable au sein de la classe des constantes. Les combattants peuvent jouer aléatoirement une action grâce à la stratégie aléatoire ou bien utilisée une stratégie plus “réfléchis “ en utilisant la stratégie fine.

Le plateau

Le plateau possède une unique stratégie de tirage : la strategyPlateauAlea. Celle-ci n’est pas totalement aléatoire, une fois que les combattants sont positionnés de façon aléatoire , les éléments sont placés de façon aléatoire également.
Cependant les murs ont une probabilité plus faible de sortir que les pastilles qui elles ont une probabilité plus faible que les cases vide.
Pour résumé les cases vide sont privilégiées pour ne pas avoir un plateau surchargé de mur entrainant l’incapacité au bon déroulement du jeu.

3. Architecture

3.1 Répartition des classes au sein des packages

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

1. Le package game : qui contient les éléments composant un plateau. Celui-ci contient en effet une classe coordonnée qui représente la position d’un objet au sein du plateau. Puis est composé également de sous package : arme, combatant , élément et grille.

2. Le package gui : contient les classes propres à la partie graphique du jeu.

3. Le package observer : contient les classes mettant en place le pattern observer.

4. Le package main : représente la classe Main

5.Le package constante : représente les constantes du jeu.

6. Le package vueconsole : représente les différentes vues propres au terminal.

3.2 Diagramme

FIGURE 1 – Diagramme de package

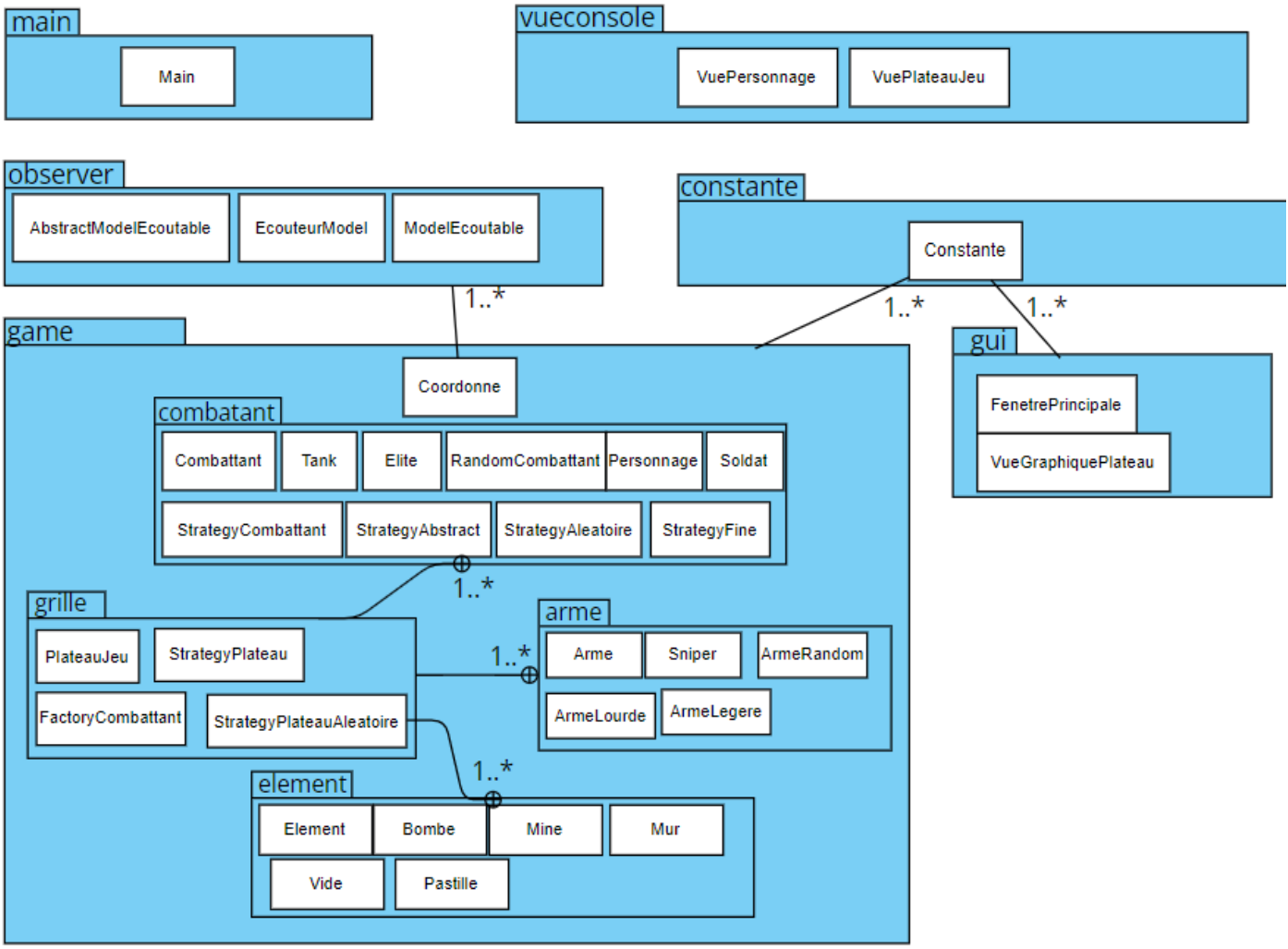
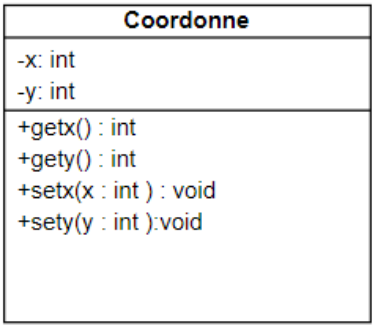
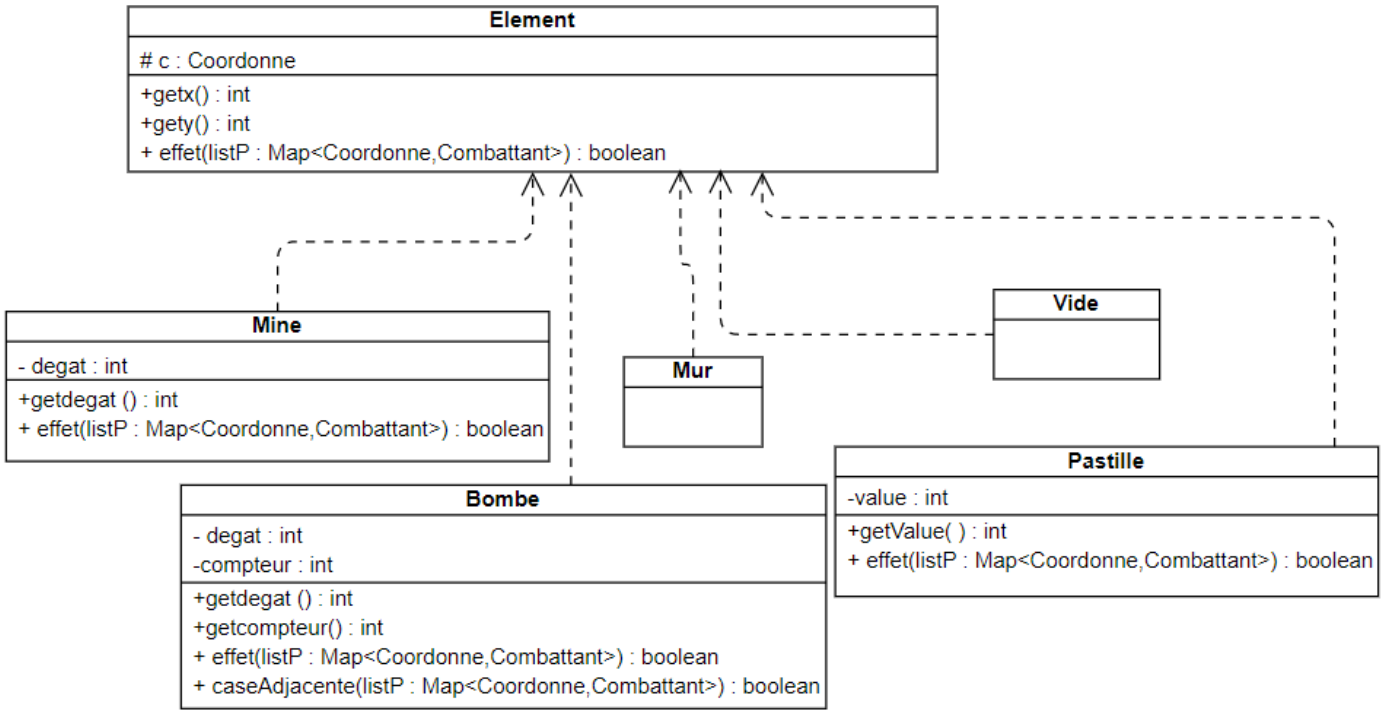


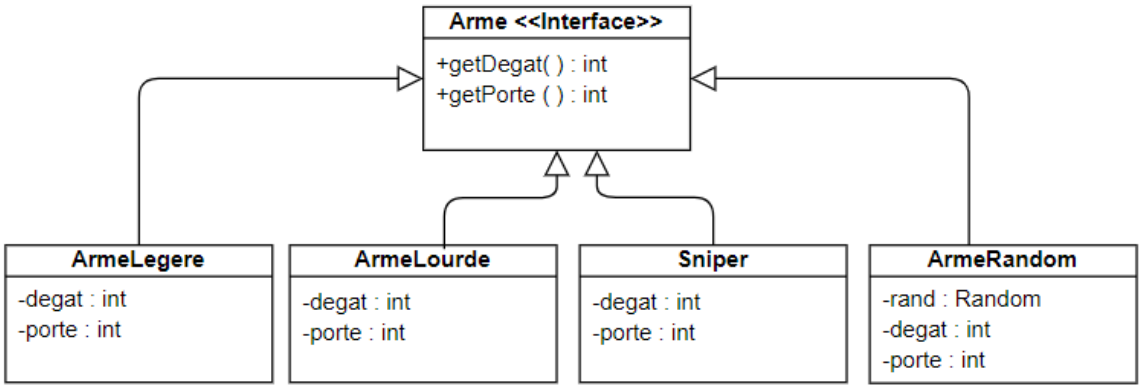
FIGURE 2 – Diagramme du package game



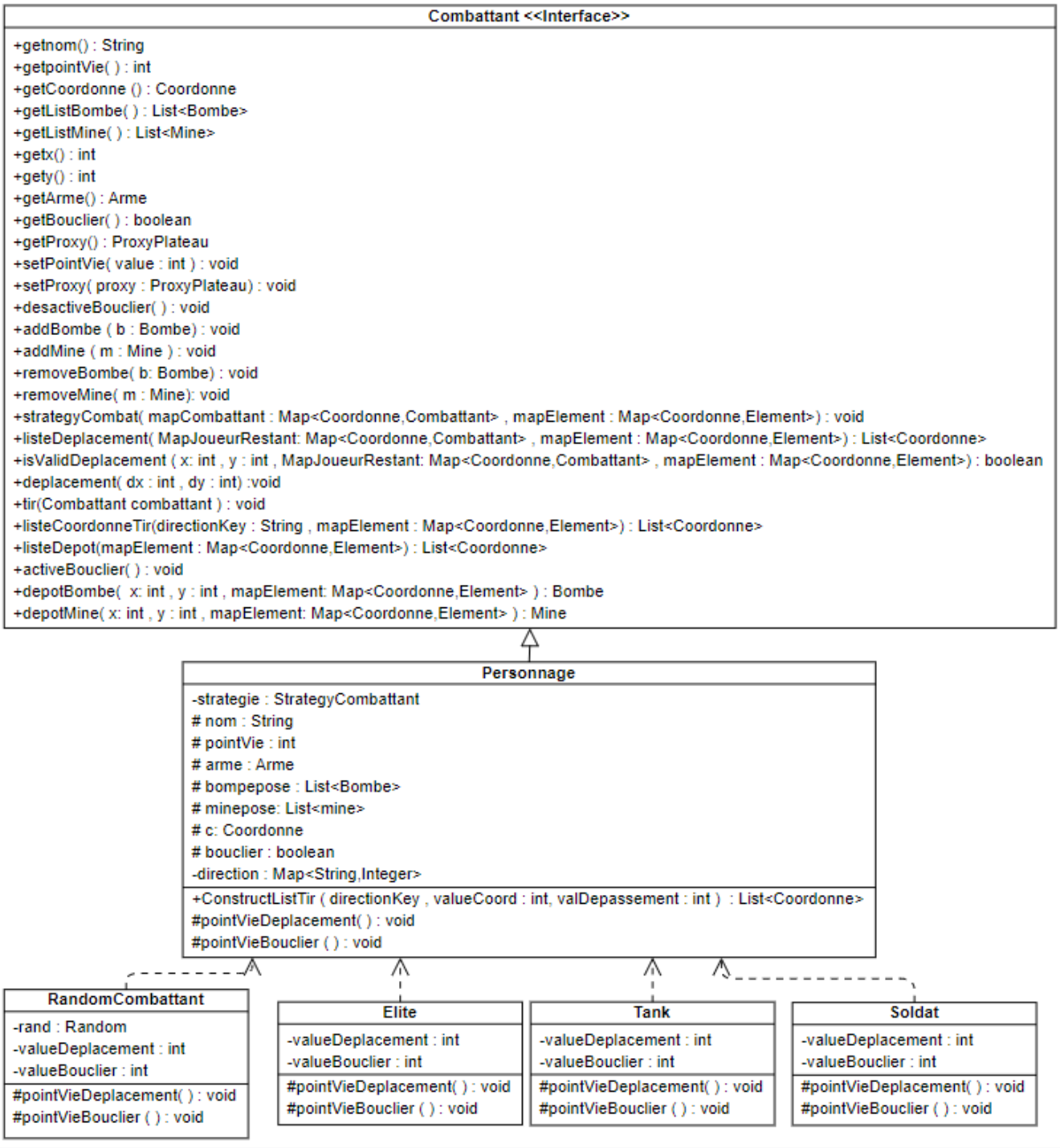
Sous-package element :

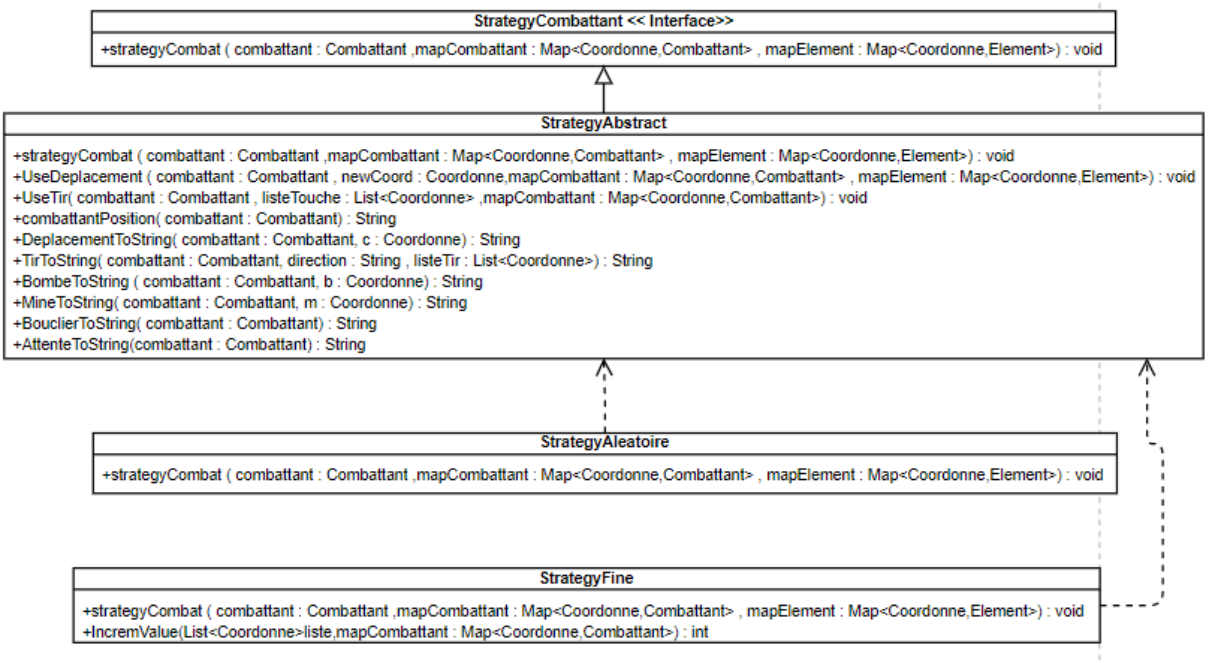


Sous-package arme:



Sous-package combatant:





Sous-package grille:

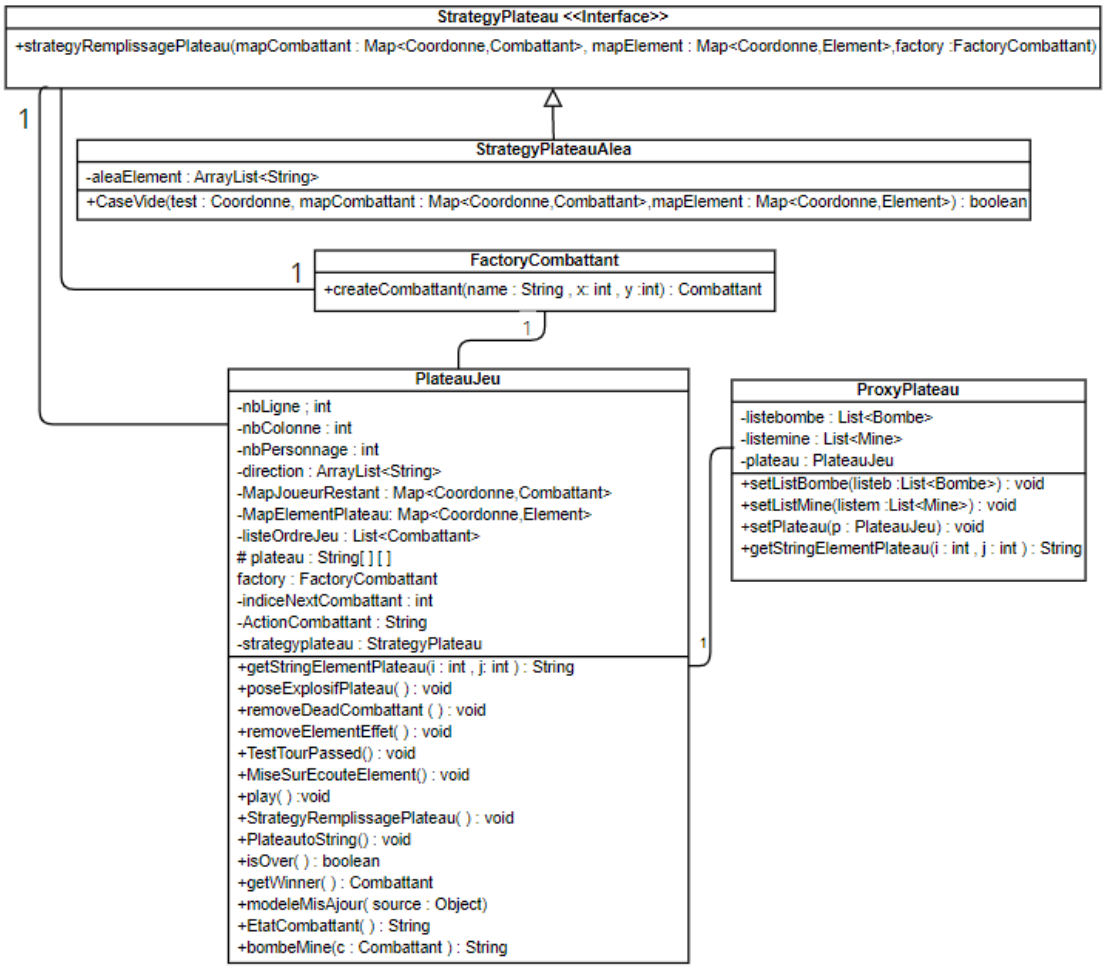


FIGURE 3 – Diagramme du package observer

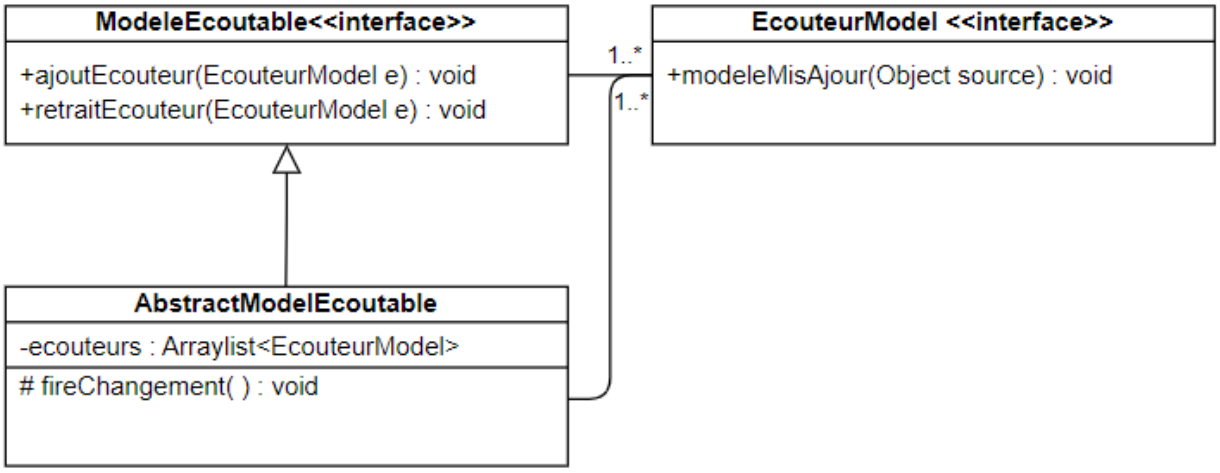


FIGURE 4 – Diagramme du package gui

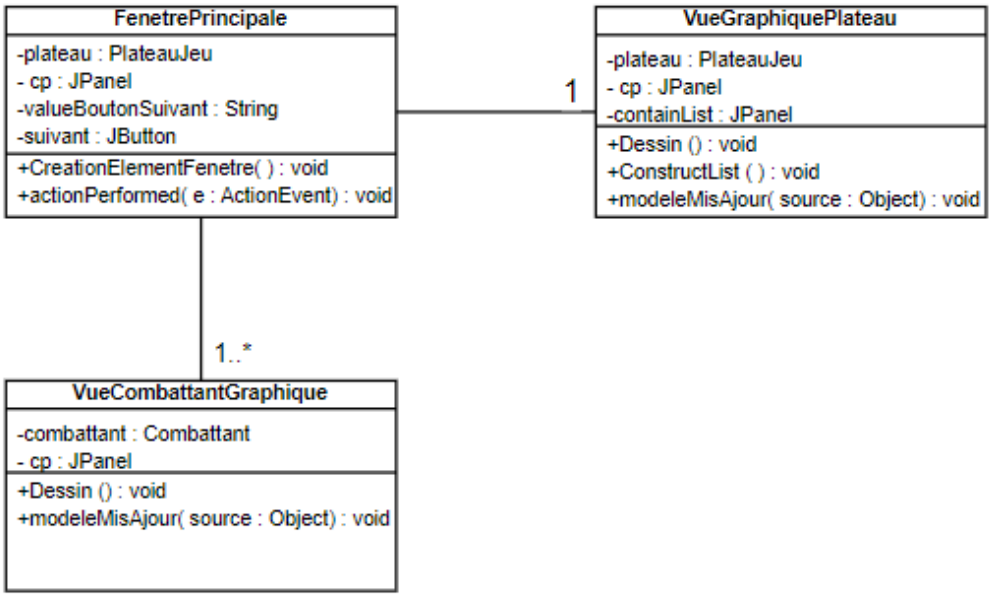


FIGURE 5 – Diagramme du package vueconsole

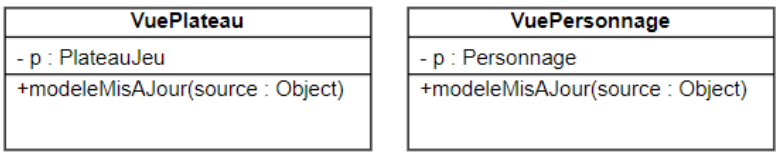
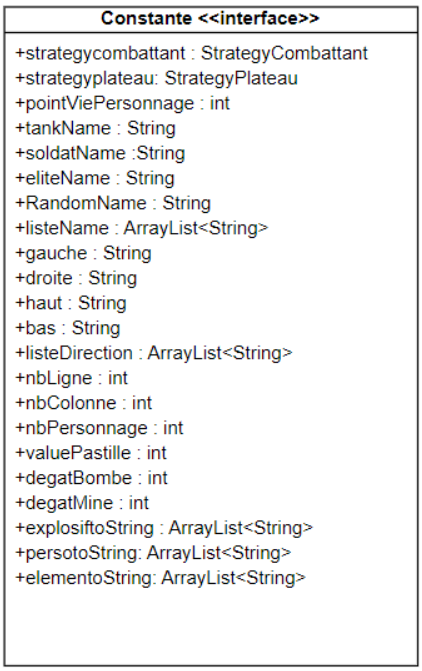


FIGURE 6 – Diagramme du package constante



5 Manuel d'utilisation

Architecture du dossier principale :

- dist : permet de contenir le fichier .jar propre à l'exécution du programme.
- src : contient les packages ainsi que leurs différentes classes.
- Doc : contient la javadoc et ses différents modules nécessaire.
- Rapport : contient le rapport sous format pdf.
- build.xml : Permet la création des répertoires nécessaires à la compilation, la Javadoc et également le .jar pour exécuter celui-ci.
- compilation.sh : Script permettant également la compilation et l'exécution du programme.
- sources.txt : contient les sources utiles au script d'exécution compilation.sh.
- Un fichier Constante au sein du package constante permet la configuration des options jeu.

Compilation du projet :

Première option : Ouvrir un terminal à la racine du projet puis effectuer la commande ant qui exécuteras le build.xml.

Seconde option : Lancer le script de compilation avec la commande : ./compilation.sh

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 des différents pattern pour la mise en place d'un code propre et réutilisable 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 :
Les différents patterns, le développement de l'application en MVC ainsi que l'implémentation des règles du jeu ont été correctement effectuées. Cela dit quelques optimisations reste envisageable.

6.1 Optimisation possible

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

1. Amélioration visuelle de l'interface graphique : inclure un Tilset 2D pour représenter les différents éléments du plateau.

2. L'implémentation d'une Stratégie élaborée pour chaque type de combattant.
Cependant surcharger le code d'avantage n'était pas désiré par la totalité du groupe. L'objectif ciblé était clairement de développer correctement l'application en MVC ainsi que l'utilisation des différents patterns, ce qui a retenu toute notre attention.