



Université de Caen

U.F.R. de Sciences

Année 2015 – 2016

L3 Informatique

Génie logiciel

Devoir maison

Fils Rouge

Rapport

Etudiants:

**Kapema Omba Serge
Kapema Shako Cedrick
Eklou Manasse Serge
Vatel Nicolas
Rekik Manel**

Manager :

**-Yann Mathet
- Bruno Zanuttini
-Gregory Bonnet**

Introduction

Le but de ce TP est la réalisation d'une application permettant des affrontements entre robots sur une grille. Le jeu sera un jeu de stratégie au tour par tour sans l'intervention d'un utilisateur sauf quand il s'agit de déterminer le nombre de robot et la taille de la grille, c'est à dire des robots jouant automatiquement.

L'application devra présenter une interface graphique intuitive, et permettant le déplacement de robot et de leur attaque entre eux.

L'application sera codée en langage JAVA SE (Standard Edition) et à la fin du TP, les comportements des robots devront être vérifiés par des tests unitaires.

1- Découverte des règles et but du jeu

Le jeu consiste à demander au joueur de saisir la longueur et la largeur de la grille avec un nombre de robot(s). Les robots s'entretuent mais chaque action du robot et dépend de beaucoup de paramètres qui amène le robot à parcourir un arbre de décision définit au préalable, un robot qui n'est pas protégé est susceptible de mourir face une missile ou un laser.

Un robot lance un missile s'il voit l'ennemi plus loin devant lui et ce missile avance case par case jusqu'à atteindre sa cible. Qui elle est toujours là ou sort de la grille. Le laser peut s'envoyer dans le cas où il y a un robot devant à la case voisine. Si l'ennemi n'est pas protégé il peut mourir et disparaître de la grille.

Chaque robot a une énergie au lancement et celle-ci se décrémente à chaque action jusqu'à passer à zéro; et le robot arrête tout mouvement en récupérant de l'énergie juste en faisant un pause puis recommence. Le jeu se termine lorsqu'il reste un seul robot sur la grille alors le robot restant est déclaré gagnant.

L'utilisateur n'agit pas lors de l'exécution du jeu car tous les robots sont indépendants. Un robot protégé par un bouclier résiste à un missile et au laser. Un robot en repos peut mourir car il ne se déplace pas et il est à la merci des autres robots. Un robot peut avancer ou reculer ou tourner dans le cas où il est coincé. Dans le cas où il y'a un missile qui arrive tout droit devant lui ce dernier fait une action hasardeuse qui peut lui être fatale ou chanceuse.

2- Maquette

L'interface finale du jeu ressemble aux maquettes présentées ci-dessous:

2-1-Accueil

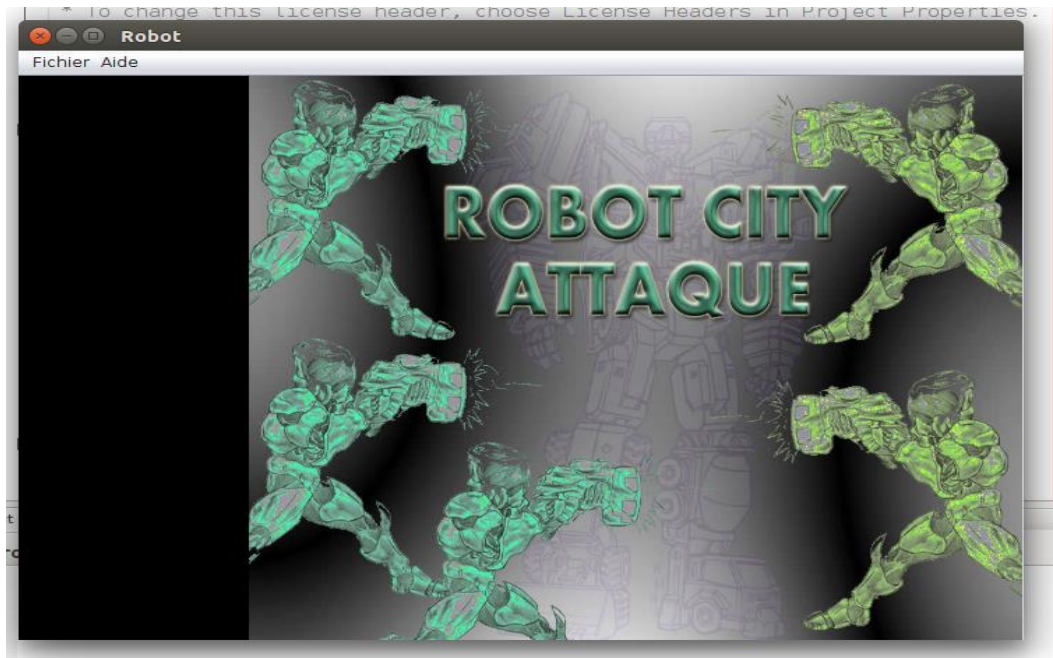


Figure 1: Accueil

2-2- Aide

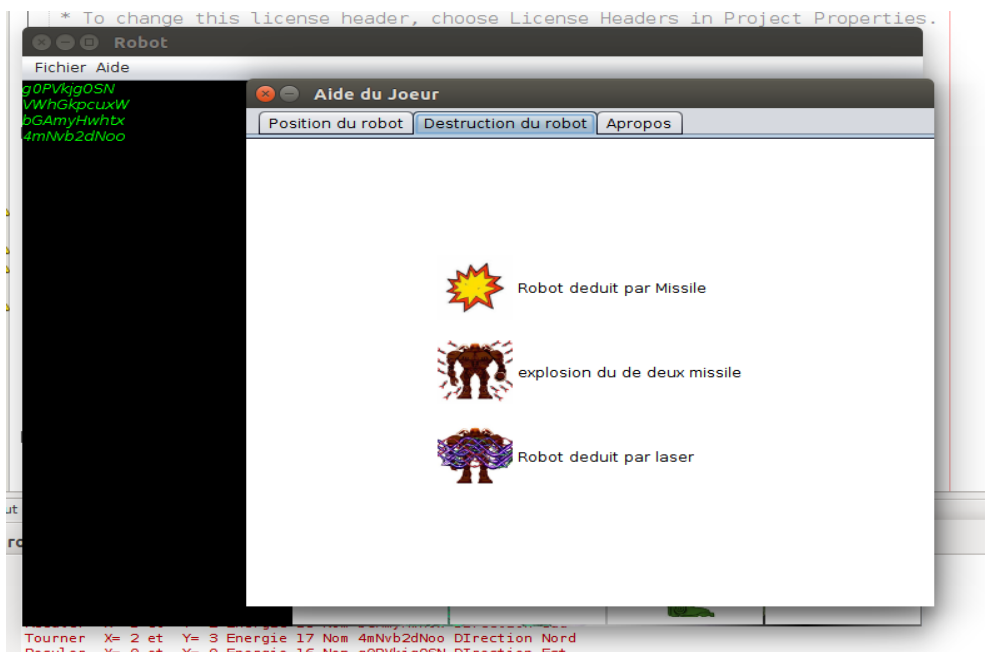


Figure 2: Aide

2-3-Surface de jeu

-Le paramètre du jeu

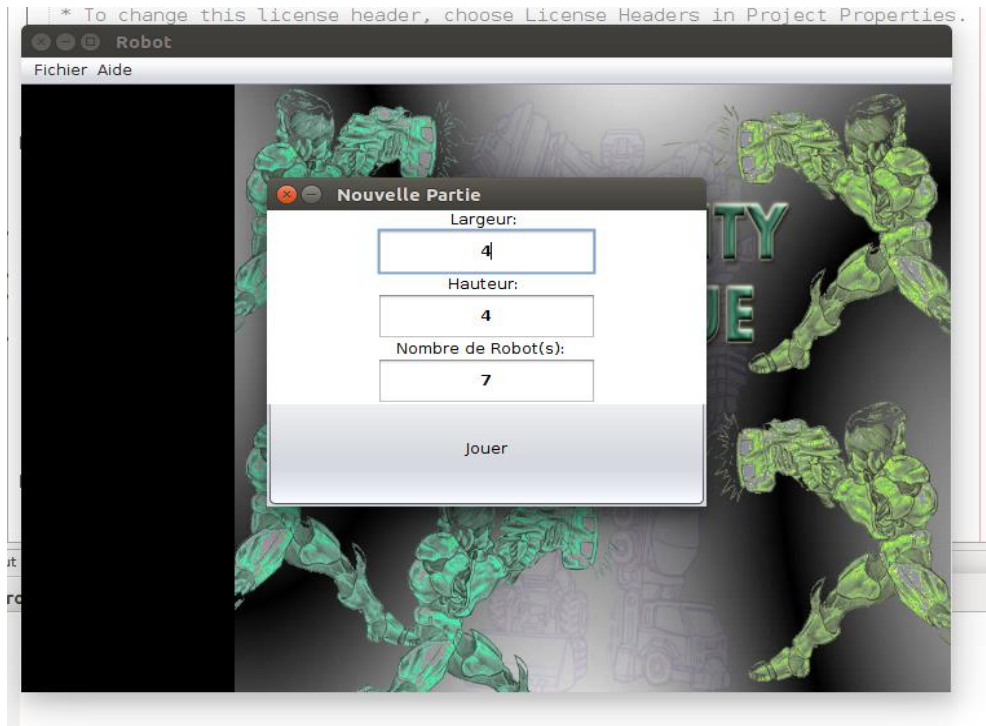


Figure 3: le paramètre du jeu

- La surface de jeu

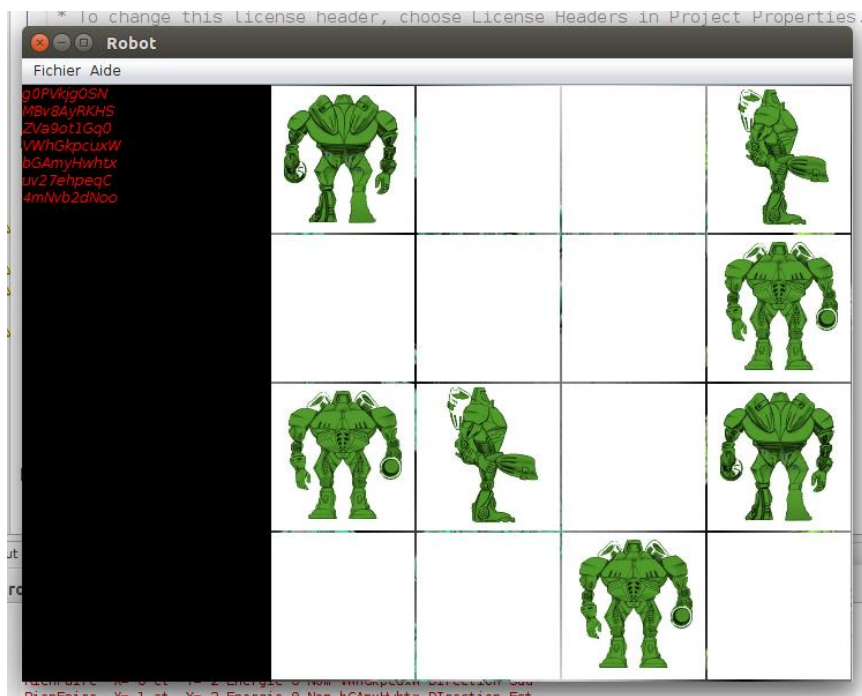


Figure 4: Surface de jeu

3- Diagramme de cas d'utilisations de l'application

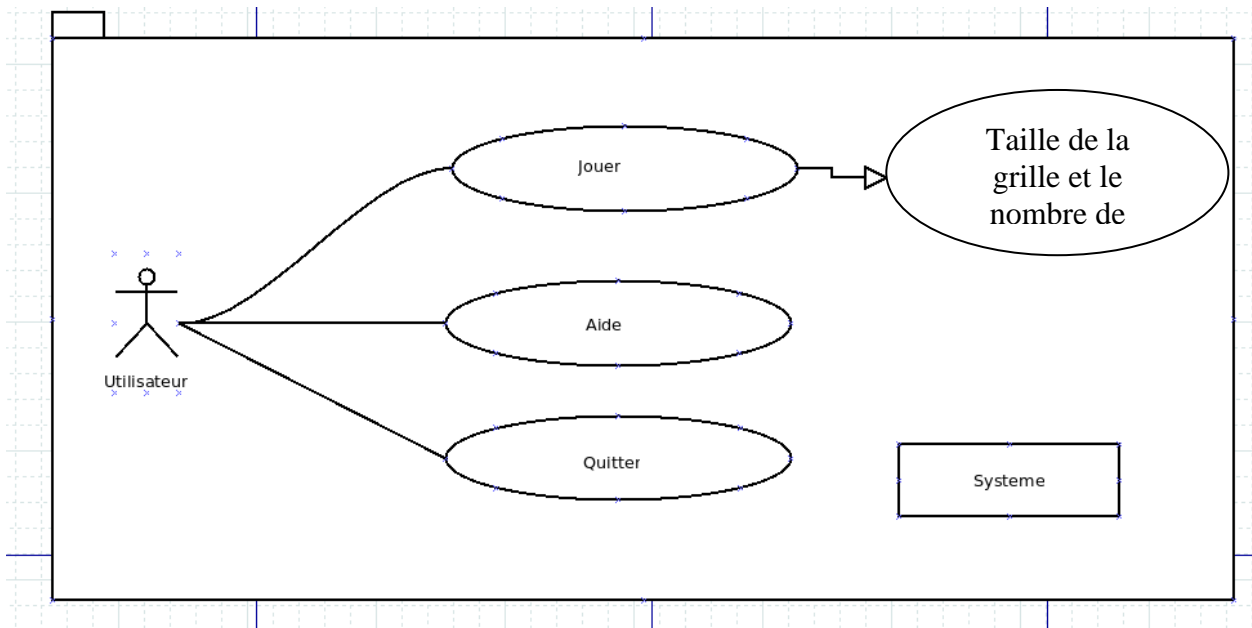


Figure 5: Diagramme de cas d'utilisation

4- Diagrammes de séquences

4-2- Diagramme de séquence pour demander de l'aide

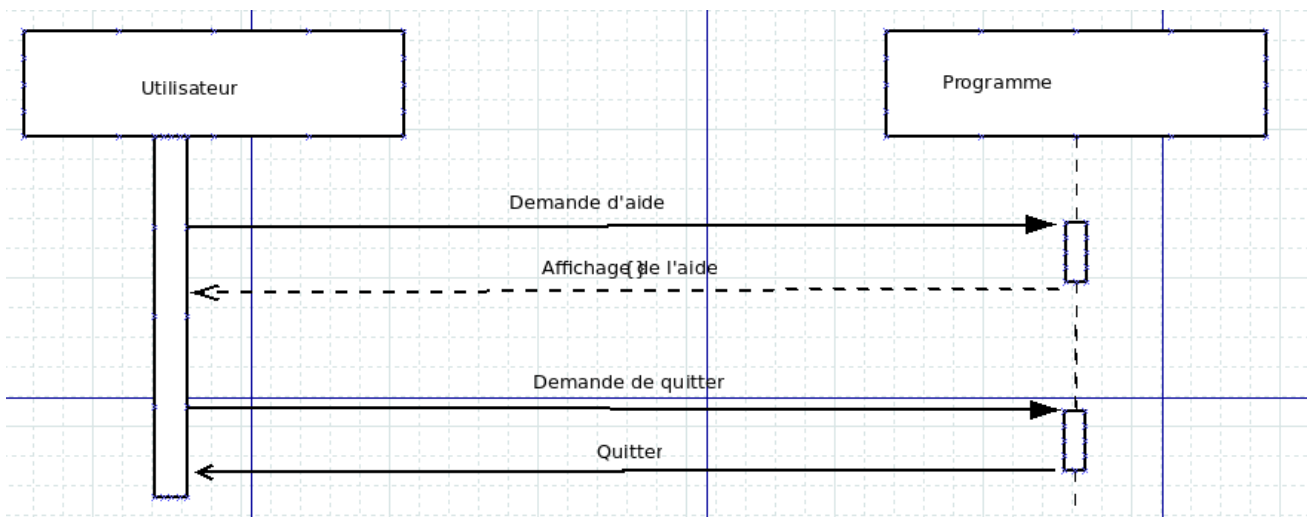


Figure 6: Diagramme de séquence pour demander de l'aide

4-2- Diagramme de séquence pour commencer une partie

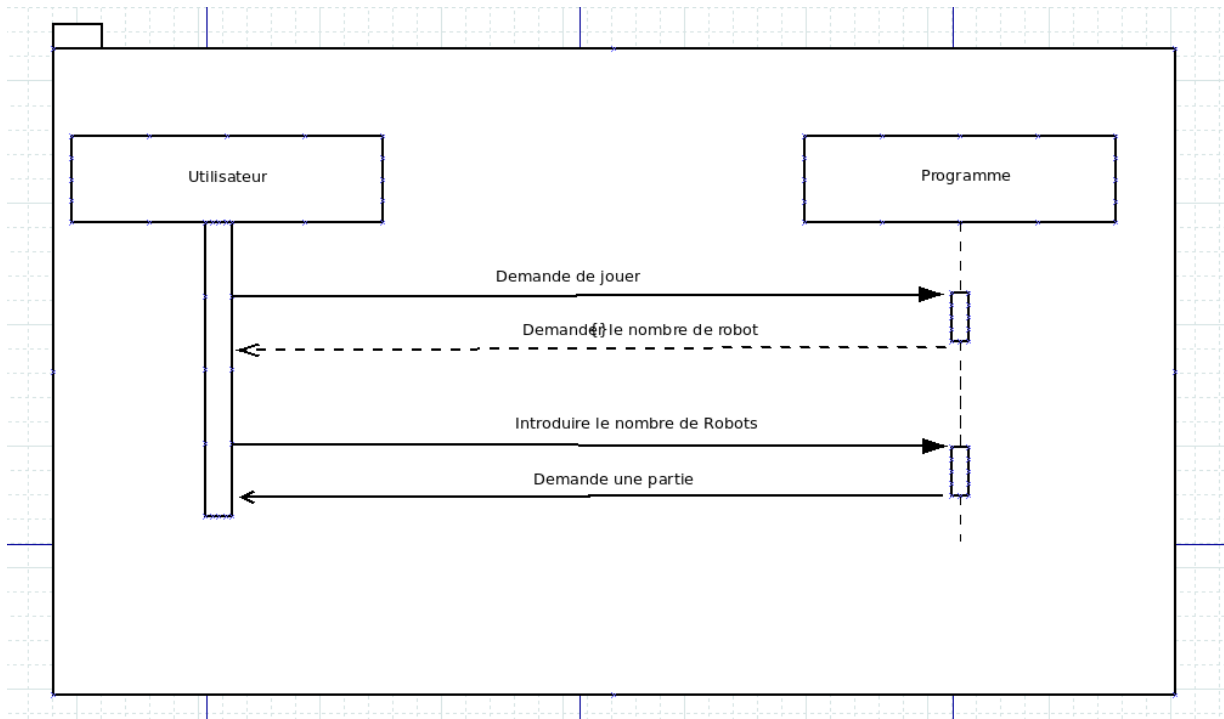


Figure 7: Diagramme de séquence pour débiter une partie

5- Diagrammes de classes

Voici le diagramme de classes qui représente les différentes classes implémentées et l'architecture de l'application.

A. Modèle de l'Arbre Binaire

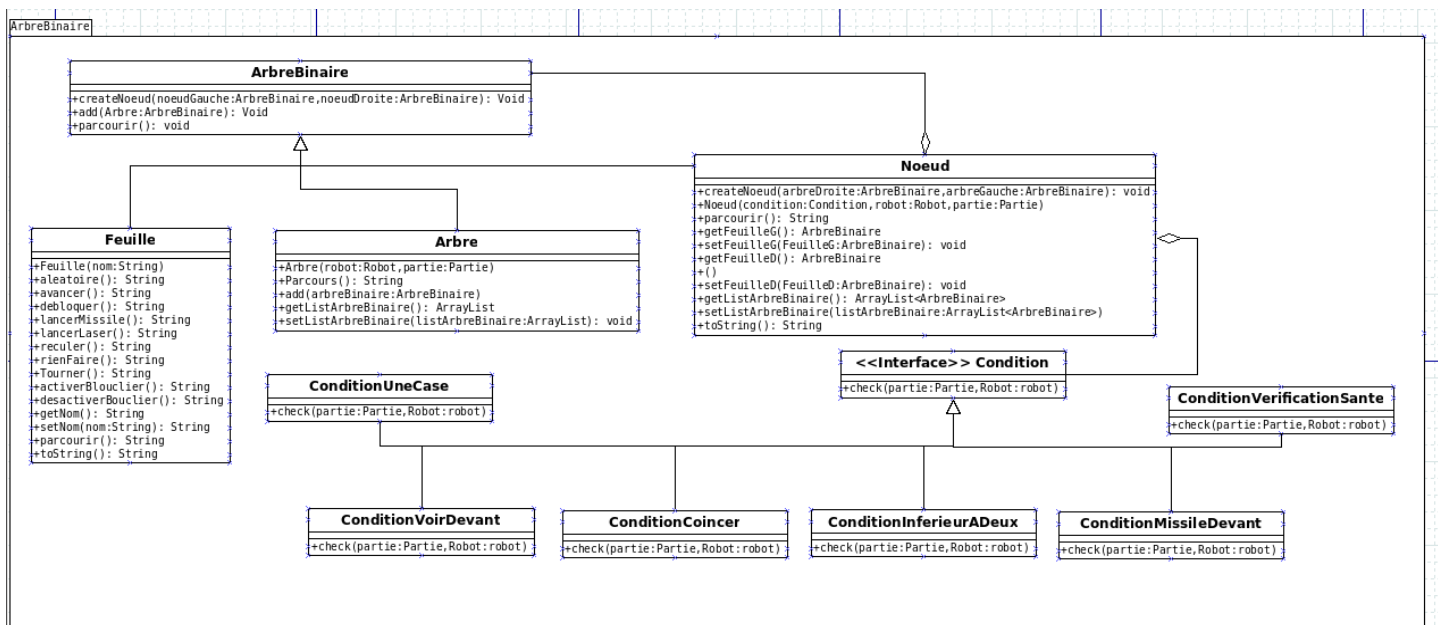


Figure 8: Diagramme de classes modèle de l'arbre binaire

B. modèle

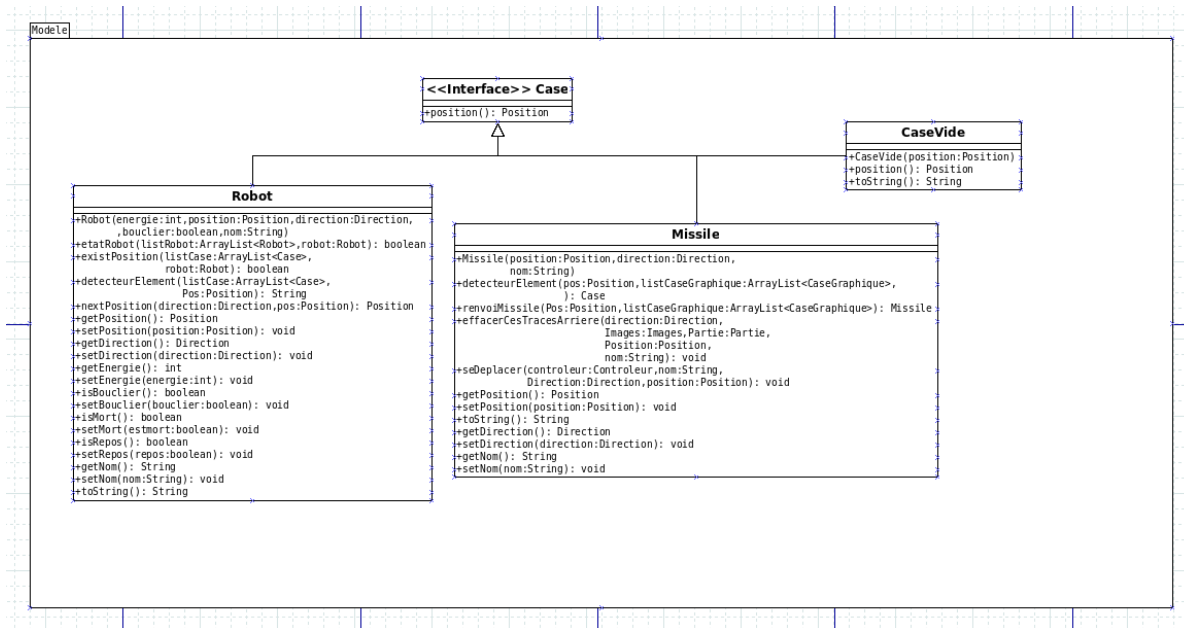
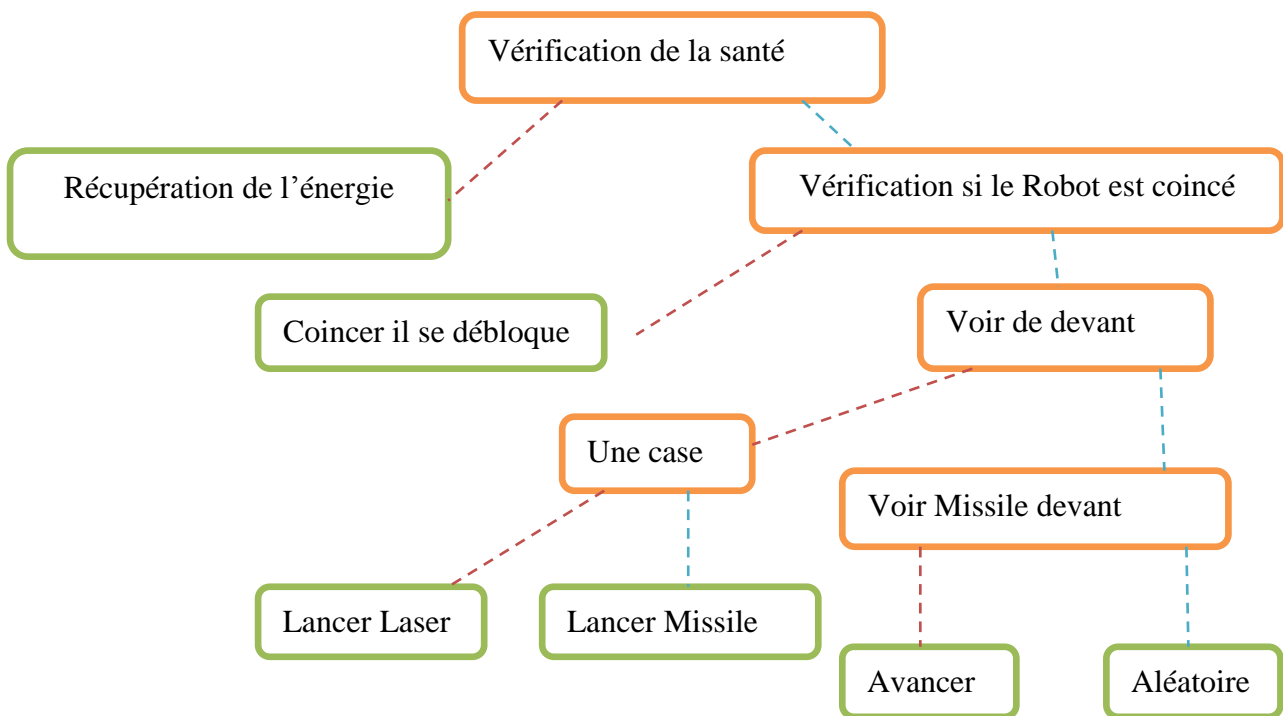



Figure 9: Diagramme de classes modèle

6-harchitecture de l'Arbre Binaire



	Feuille
	Nœud
	Faux
	vraie

7- Diagrammes de Packages

Le diagramme de séquence suivant illustre l'implémentation du modèle MVC comme architecture de conception de l'application :

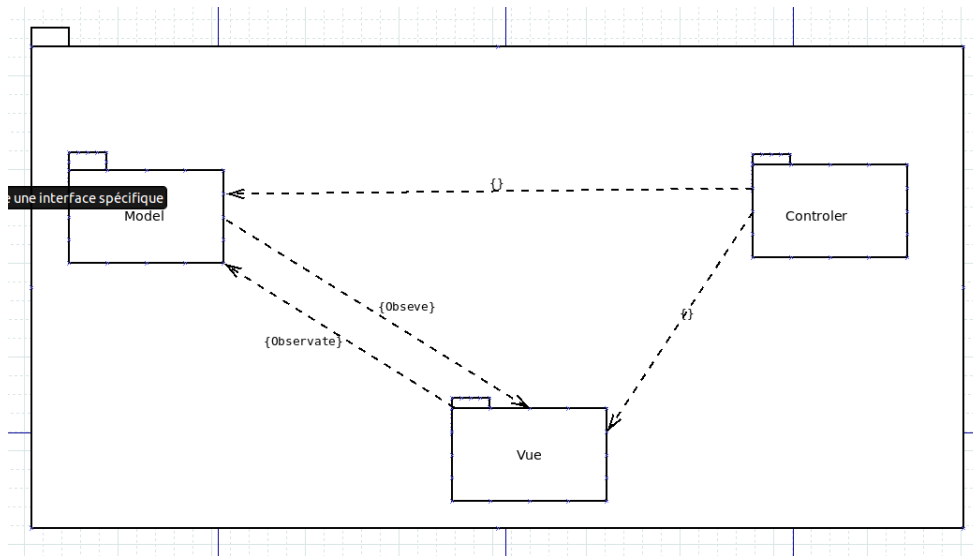


Figure 10: Diagramme de Packages (MVC)

7- Réalisation

- L'application est codée en Java.
- Le modèle qui est isolé des différentes parties est implémenté en premier.
- Suivant une architecture MVC, la vue est aussi séparée dans un package à part.
- Le contrôleur fait la liaison entre la vue et le modèle.
- Le design pattern Observer est implémenté pour rendre le modèle observable et la vue observatrice du modèle.
- Le design pattern Stratégie permet de faire jouer des joueurs robots avec différentes stratégies.
- Des fonctionnalités avancées sont en cours de développement.
- Des tests unitaires présentés.