Theo Ponthieu IG3

Java BattleShip

1) La compilation :

Pour compiler l'application à partir de la racine du projet :
javac bataille/actifs/*.java bataille/passifs/*.java exception/*.java fr/battleship/*.java ponthieu/theo/*.java

Pour excecuter (comme précisé dans l'énoncé) :
Jouer dans le mode voulu : java ponthieu.theo.Battleship
Tester les IA : java fr.battleship.TestIA

2) Architecture :

a) Mon architecture est composée de cinqs packages :

- bataille : Contient toute les class liées directement à la bataille navale. Il est composé de deux sous packages (actifs/passifs) qui permettent de séparer les class en deux. Le Main n'instancie directement que les class qui se trouvent dans le package actif.
- exception : Contient toutes les exceptions susceptibles d'être utilisées par la bataille navale
- ponthieu.theo: Contient le Main qui permet de jouer à la bataille navale contre un joueur ou une IA.
- fr.battleship : Contient le main qui test les IA.

Cette architecture permet une prise en main très rapide car elle est triviale. Elle sépare les Exceptions des class java utilisées par la bataille pour plus de lisibilité.
Les Mains sont séparés pour pouvoir les utiliser selon les commandes demandées dans l'énoncé.

Choix pour la conception :

Le Main est l'arbitre du jeu et fait passer les tirs d'un joueur à l'autre (Humain ou IA).
La class Joueur (Humain) et IA mettent en œuvre l'interface Playerable qui permet de les considérer comme des joueurs identiques dans le Main. Seule la manière de les instancier change.

Ces deux class utilisent la même Flotte et les mêmes bateaux.
Ce sont les bateaux qui contiennent leurs positions dans la grille du joueur qui les détient. Ils stockent aussi leur coordonnées touchées.
Chaque joueur gardent en mémoire la grille de positions déjà essayées sur l'adversaire.
L'instanciation des joueurs se fait grâce à leur constructeur qui gère les paramètres de l'IA et de l'humain et demande les informationsà l'utilisateur si nécessaire.

Le niveau de l'IA est un paramètre. Lors du jeu, l'IA choisit les coordonnées sur lesquelles tirer selon son niveau donné lors de l'initialisation.

Les choix et la conception ont été réalisé en collaboration avec Nicolas Guary.

b) Les avantages :

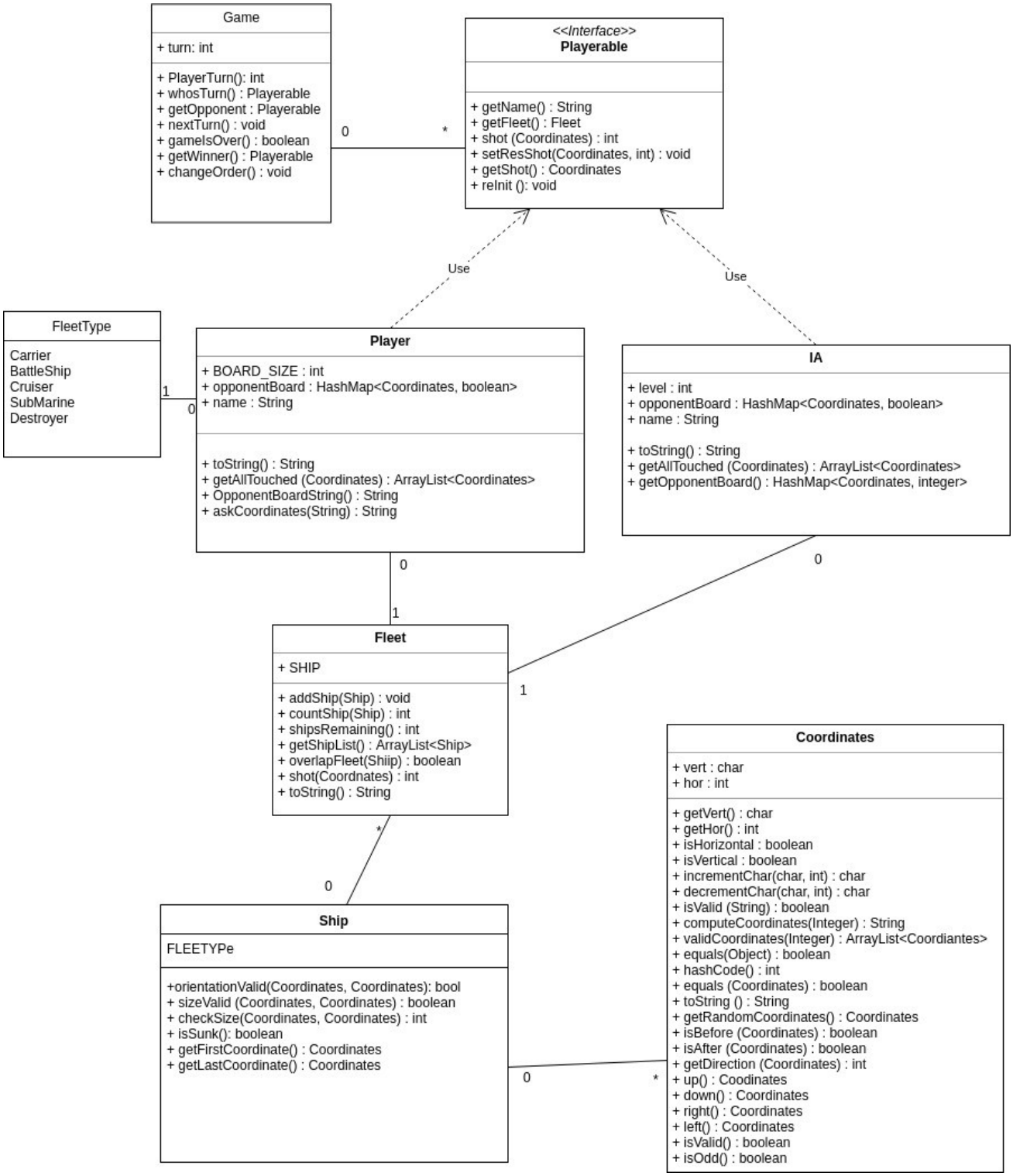
Cette architecture est très simple et facile à comprendre. Elle permet une prise en main rapide et peu de réflexion pour pouvoir être utilisée.
La séparation de la bataille en deux permet de voir rapidement quelles class doivent être directement utilisées et lesquelles font partie uniquement de l'implémentation interne de la bataille navale.
Cette architecture permet d'éviter au developpeurs qui voudraient utiliser le projet pour changer les règles de la bataille d'éviter des erreurs. Il peut simplement, en regardant uniquement l'architecture, deviner quelles class instancier pour réaliser sa propre version de la bataille navale.

Theo Ponthieu IG3

Les inconvénients :

Cette architecture est moins adaptée si l'implémentation de la bataille navale doit changer. Il faudra déplacer des class qui seront instanciées directement dans le main car elles seront utilisées differemment.

c) Une figure UML qui décrit l'application :



Theo Ponthieu IG3

3) AIs :

Level Beginner :

- Cette IA tire de façon aléatoire sans mémoire.

Level Medium :

- Cette IA tire de facon aléatoire, mais avec mémoire. Cela permet de diminuer par 4 le nombre de tours nécessaires pour finir une partie classique par rapport à l'IA beginner.

Un augmentation par 4 de la difficulté permet d'assurer à plus de 99 % la victoire de l'IA Medium sur l'IA beginner.

Level Hard :

- Cette IA est plus complexe que les précédentes. Il y a plusieurs améliorations qui font qu'elle est meilleure. Les tirs sont fait de manière aléatoire uniquement sur les cases impaires (A1, A3, B2…). Le plus petit bateau étant de 2 cases, cela permet de le repérer sans à avoir à toucher toutes les cases.

Cette IA ne tire pas uniquement de façon aléatoire. Lorsque qu'elle touche un bateau, elle va commencer par chercher son orientation en tirant sur les cases valides autour de la case touchée. Un fois qu'elle a trouvé la direction du bateau (si le bateau n'est pas déjà coulé) elle va tirer sur une case valide dans le prolongement des deux cases précédemment touchées.

Quelquefois le placement des bateaux peut induire en erreur, dans ce cas pas de cases valides dans le prolongement du faux bateau. Elle tire alors au hasard autour de ces deux cases pour se débloquer.

Cette stratégie permet de diminuer d'environ par deux le nombre de tour nécessaires pour finir une partie par rapport à l'IA Medium. Cela permet d'avoir environ 98 % de chance de gagner.

4) Post-mortem :

Ce projet n'est pas mon premier projet réalisé en java, j'ai déjà eu l'occasion d'acquérir de l'expérience pour la mise en place de la structure et du déroulement de l'implémentation des fonctionnalités. Les précédents Post-Mortem m'ont appris que la reflexion avant la phase de code permet de gagner beaucoup de temps. Néanmoins, ce projet a été très formateur.

a) Ce qui s'est bien déroulé.

L'implémentation du jeu de base entre deux joueurs s'est bien déroulé. J'ai dû faire très peu de rollback dans le code car j'avais passé beaucoup de temps sur papier avant de commencer à coder. Quelques cas de figure n'ont pas été pris en compte pendant la phase de réflexion mais rien de particulièrement handicapant qui aurait nécessité de recommencer l'implémentation.

Lors de la création des IAs, la mise en place et la stratégie ont elles aussi données de bons résultats. Je pense notamment à la mise en place de la dernière IA qui a donné instantanément des résultats très satisfaisants lors de la phase de tests.

b) Ce qui s'est mal déroulé.

Lorsque que la bataille navale est devenue oppérationnelle pour deux joueurs, il a fallut implémenter les IAs. La difficulté fut de trouver la bonne implémentation pour permettre l'utilisation de la bataille navale de plusieurs façons. Les joueurs peuvent être une IA ou un humain et les IAs peuvent avoir des niveaux différents. Une fois l'implémentation de la première IA le Main doit pouvoir en étant unique, permettre de faire tourner la bataille navale selon n'importe quels paramètres de configuration. J'ai dû changer l'implémentation pour factoriser du code. Je me suis perdu dans l'implémentation car je m'étais trop éloigné de la réalité et j'ai dû m'y reprendre à deux fois avant d'avoir quelque chose de convainquant.

Ce changement d'implémentation m'a fait perdre un peu de temps.

Theo Ponthieu IG3

c) Ce qui j’ai appris.

Avant de commencer ce projet je savais qu’il fallait s’attarder partciulièrement sur la phase de conception pour éviter de faire des erreurs et se retouver dans une impasse.
Ce que je n’avais pas pris en compte est que mon implémentation devait permettre l’extension de nouvelles fonctionnalités comme les IAs. J’ai réalisé le joueur sans avoir de réflexion sur la suite. Mon erreur fut de penser que les humains et IAs n’avaient rien en commun. J’ai appris qu’il faut chercher, une fois la phase de conception finie, a éprouver les premières idées pour trouver des manières de l’améliorer même si cette dernière se suffit à réaliser le projet en soi.
Plus j’acquière de l’expérience et plus la réflexion avant l’implémentation me semble importante.