UNIVERSITÉ DE LORRAINE

Master 1 Informatique

—

DOSSIER D'ANALYSE-CONCEPTION POUR UN JEU DE BATAILLE NAVALE

Equipe 3 :

- MAZROU Abdelghani
- BEN TOUNES Samy
- OUEFIO Innocent Dieu Benit

Lien github : https://github.com/ghaniturismo/BatailleNavale-Equipe3

2017/2018

Sommaire

I) SPÉCIFICATION DU PROJET

L'objectif de ce projet est la réalisation d'un jeu « Bataille navale » sur ordinateur. Il s'agit d'une application mono-utilisateur et le joueur jouera contre le système (l'ordinateur).

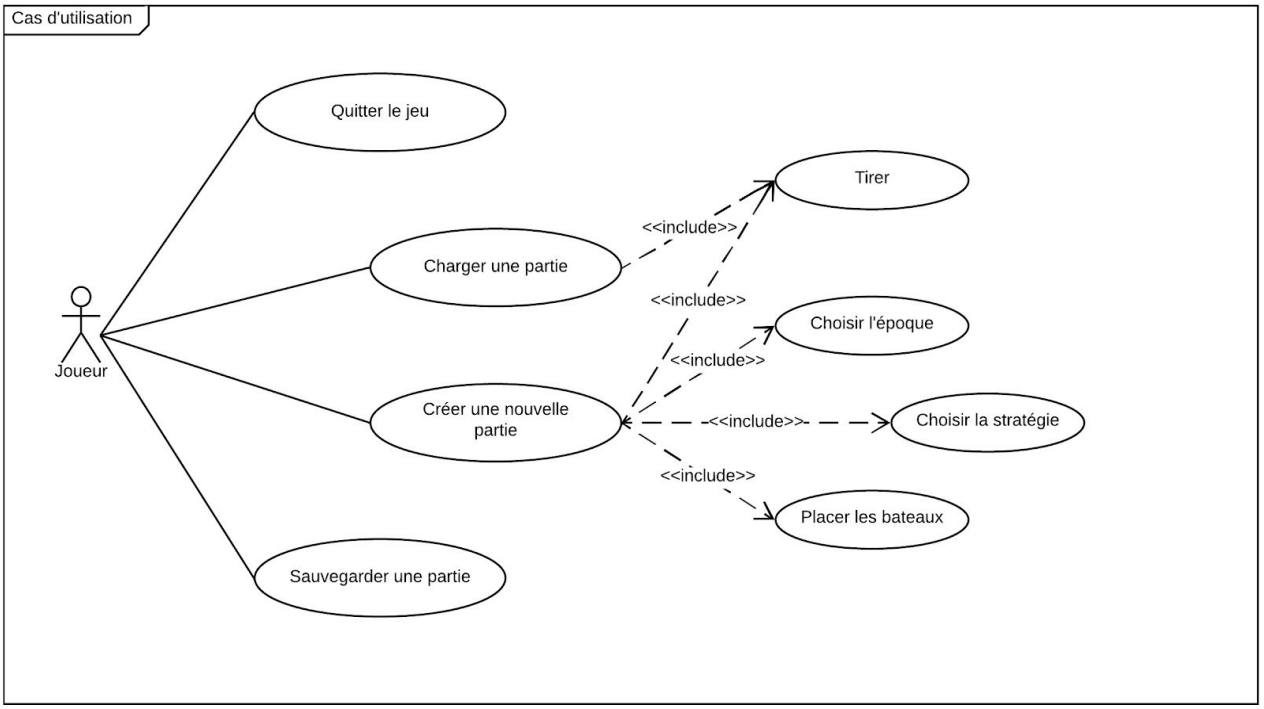
Lorsqu'une nouvelle partie est créée, le joueur doit d'abord choisir l'époque de la bataille navale afin de débuter la partie du jeu, le joueur choisit la stratégie qui sera utilisée par le système, ainsi que le comportement des bateaux de la flotte.

La partie devra se terminer lorsque le joueur à court de munitions et que le joueur système à couler tous les bateaux adverses, ou bien lorsque toute sa flotte est coulée par le système.

Le joueur visualise sa grille de jeu et les bateaux placés ainsi que la grille de l'adversaire avec ses tirs.

II/ CONCEPTION DU PROJET

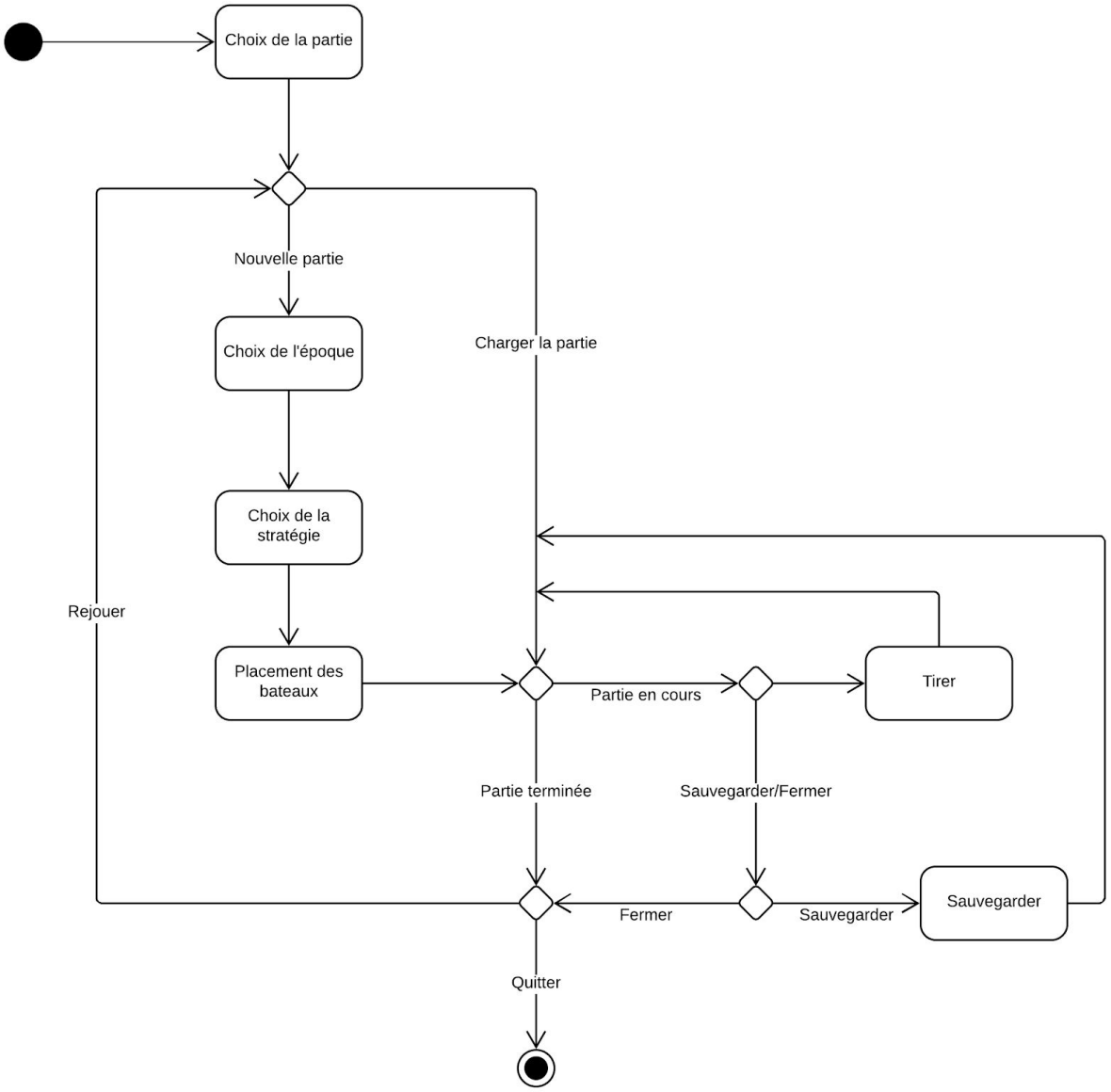
1) Diagramme de cas d'utilisation



La personne jouant au jeu a quatre possibilités. Elle peut créer une nouvelle partie, charger une nouvelle partie, sauvegarder une nouvelle partie ou bien quitter le jeu. Elle peut choisir de sauvegarder une partie et de quitter le jeu quand elle le souhaite. Par contre, pour pouvoir choisir son époque, choisir sa stratégie ou encore placer les bateaux, elle doit avant cela créer une nouvelle partie (d'où les "<<include>>"). Pour effectuer un tir, elle nécessite d'avoir charger une partie ou bien de créer une nouvelle partie.

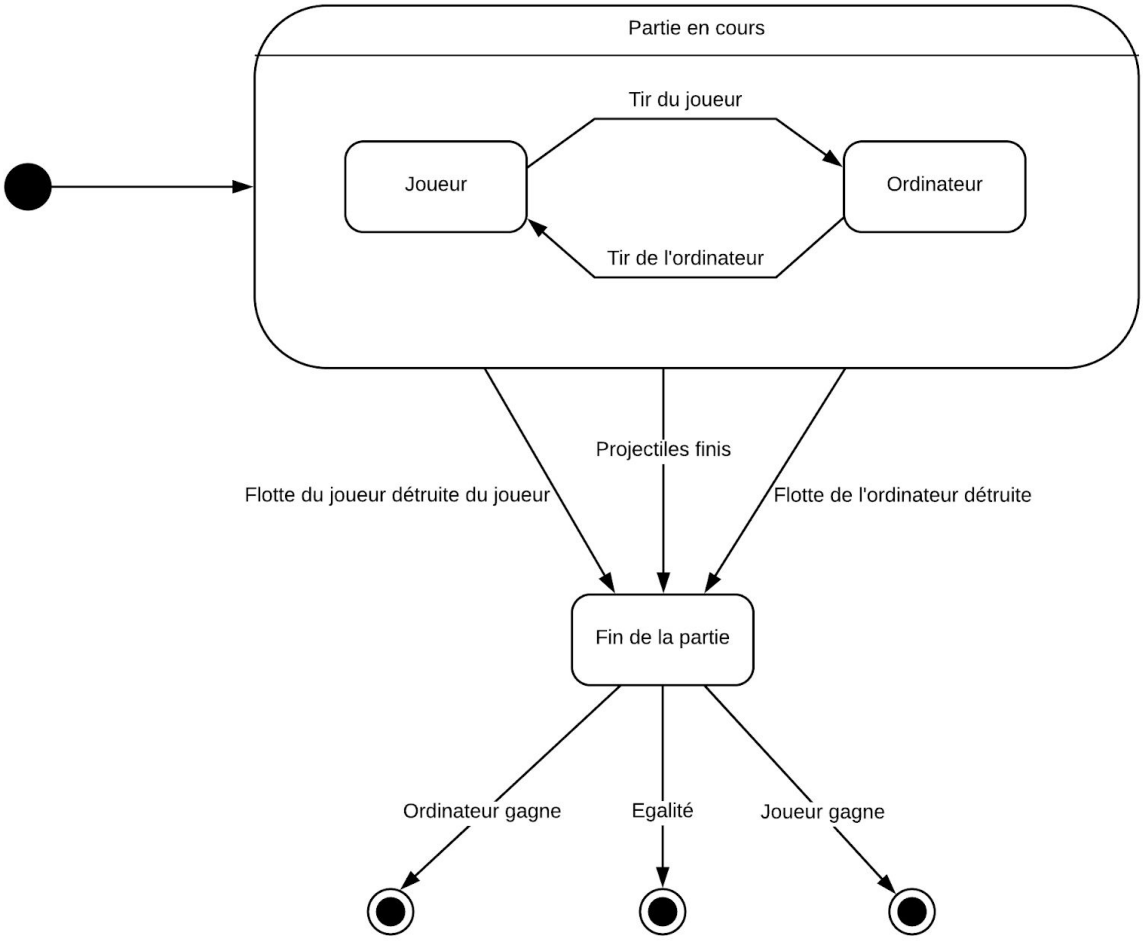
Lors du lancement du jeu, la personne jouant au jeu aura la possibilité de charger son ancienne partie ou bien de créer une nouvelle partie, et y fixer l'époque, la stratégie et le placement de ses bateaux. Au cours de la partie, la personne jouant au jeu aura la possibilité de sauvegarder sa partie ou bien de quitter le jeu.

2) Diagramme d’activités



Voici, ci-dessus, le diagramme d’activité de notre système. Au départ, l’utilisateur peut démarrer une nouvelle partie ou bien charger la partie en cours. S’il choisit de démarrer une nouvelle partie, il doit sélectionner les caractéristiques (époque, stratégie, placement des bateaux). Ensuite, la partie peut débuter. Il peut tirer (à tour de rôle avec l’ordinateur) jusqu’à ce que la partie se finisse. S’il choisit de charger la partie en cours, il reprend directement la dernière partie fermée/sauvegardée et il peut continuer la partie. Il peut également sauvegarder et fermer la partie en cours (pour quitter le jeu).

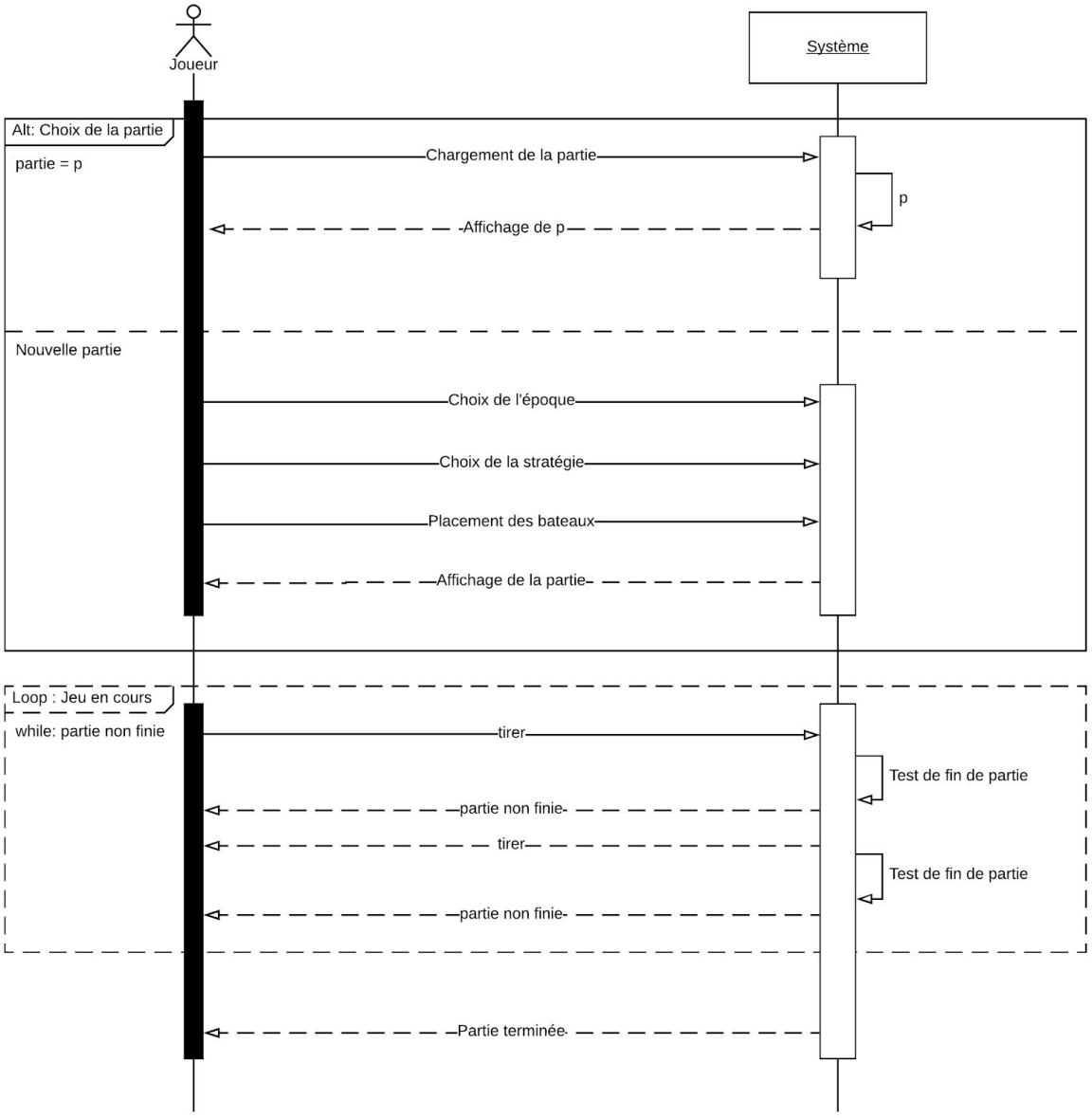
3) Diagramme d’états



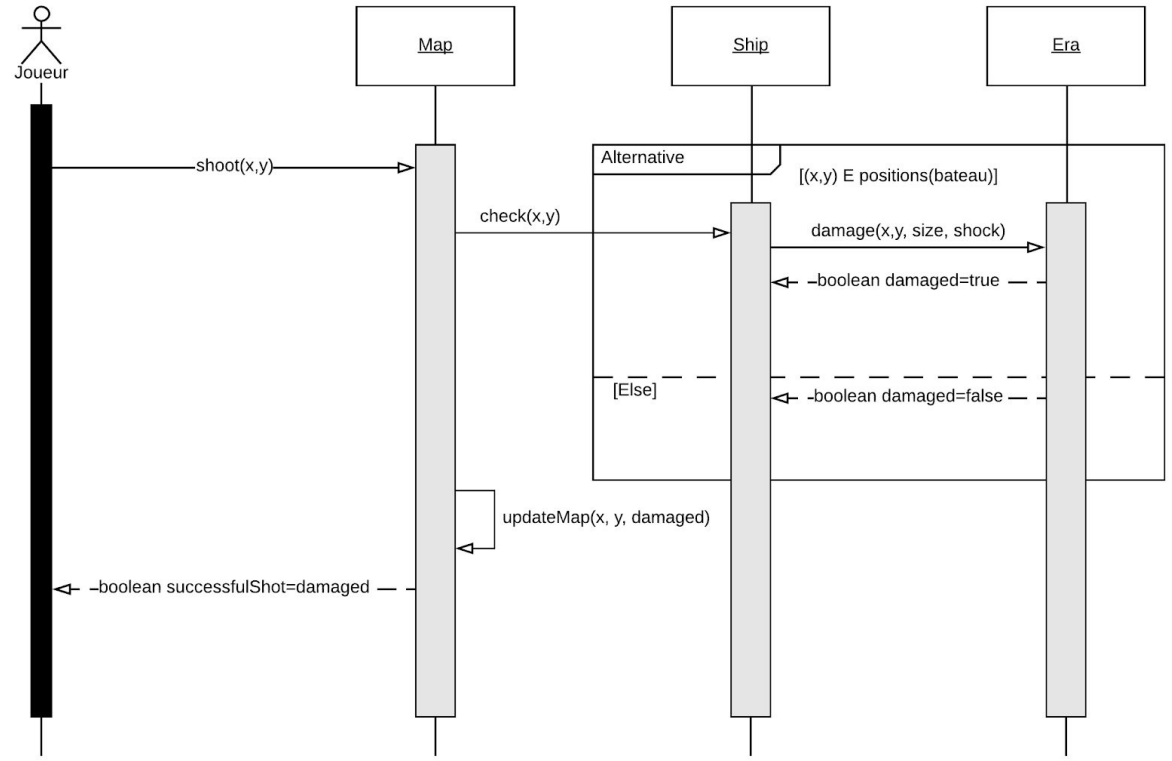
4) Diagramme de séquences

Nous allons expliquer les diagrammes de séquences (voir ci-dessous) pour les cas d'utilisation "Choisir une partie" et "Le joueur tire un projectile".

A) Choisir une partie



B) Le joueur tire un projectile



Lorsque le joueur voudra tirer un projectile (“shoot(x,y)”) à la position p = (x,y), le système va vérifier, via la map (“Map”), si la position p appartient ou non aux positions d’un bateau du système. Si p appartient aux positions d’un des bateaux du système, le bateau (“Ship”) va subir les dommages en fonction de son époque (“Era”). Cela renverra également un booléen (“damaged”) valant **true** s’il y a eu des dommages. Si p n’appartient pas aux positions d’un des bateaux du système, le booléen renvoyé vaudra **false**.
Après cela, la map se met à jour avec comme paramètres les coordonnées de p et le booléen “damaged” (“updateMap(x,y,damaged)”).
Enfin, la map renvoie au joueur un booléen valant **true** si le tir a été réussi et **false** sinon.

5) Diagramme de classes

A) Diagramme de classes global

Le diagramme de classes du jeu « Bataille navale » présente les différentes classes et interfaces du système, ainsi que les relations entre celles-ci.
Le diagramme de classes se compose globalement des classes suivantes :

BateauFactory : est un singleton servant à la construction des différents bateaux appartenant à différentes époques.

World : elle gère le déroulement d'une partie, elle contient les méthodes suivantes:

- **boolean** : finPartie() : servant à tester si la partie est terminée.
- **void** : savePartie() : permettant de sauvegarder la partie en cours.
- **void** : loadPartie() : charger une partie déjà sauvegardée.
- **void** : addShip(Bateau b) : Ajouter un bateau b la liste des bateaux .

Game: est la classe principale du diagramme. Elle contient deux objets de type World correspondant au monde du joueur et au monde de l'ordinateur.

Epoque : classe abstraite définissant l'époque dans laquelle la partie se déroule, le choix de l'époque se fera par le joueur au début de la partie.

Deux époques disponibles :

- 16ème siècle
- 20ème siècle

XVI_Bateau et **XX_Bateau** correspondent aux packages des différents bateaux de chaque époque.

IA_Strategie est l'interface qui définit les différentes stratégies à mettre en œuvre, en premier lieu on aura deux stratégies à implémenter : la stratégie Aléatoire et Intelligent.

- **AleatoireStrategy** : choisit une case au hasard dans le champ adverse pour effectuer le tir.
- **IntelligentStrategy** : commence par une stratégie aléatoire, dès qu'un bateau adverse est touché, le tir suivant sera effectué sur une case adjacente de chaque côté de la case touchée.

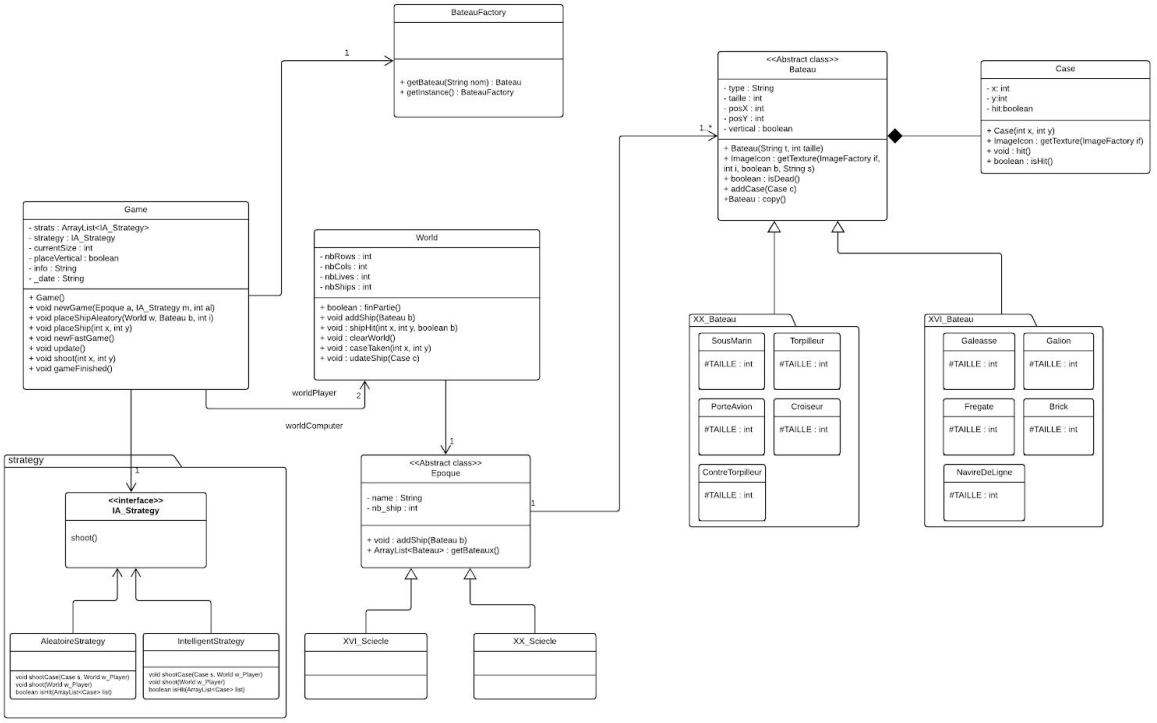
Bateau : classe abstraite qui définit un bateau de manière générale, contient les attributs :

- **type** : le nom du bateau.
- **taille** : le nombre de cases occupées par le bateau.
- **force** : correspond aux dommages infligés par chaque projectile.
- **robustesse** : le nombre de tirs nécessaires pour couler le bateau.
- **nb_projectiles** : le nombre de projectiles à disposition du bateau.
- **isVertical** : un booléen indiquant si le bateau est en position verticale (si faux -> position horizontale).

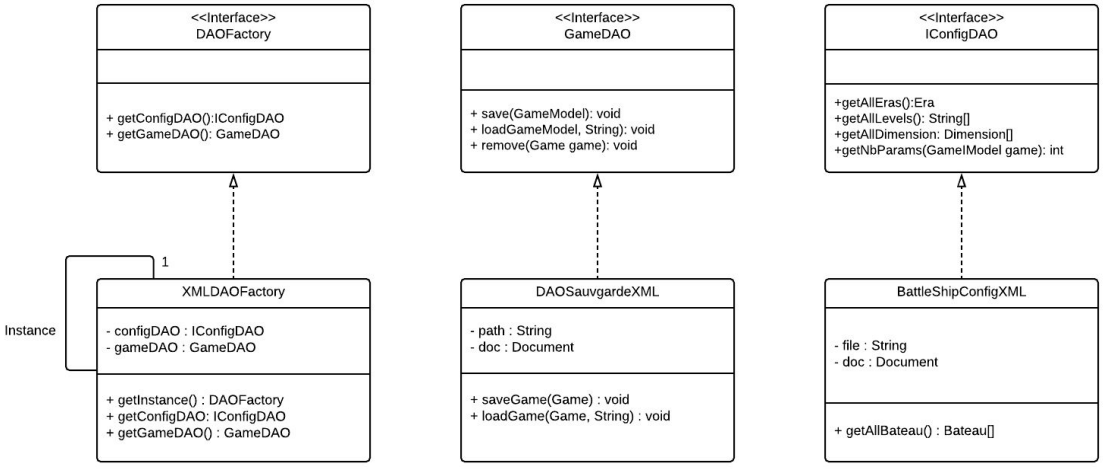
Case : correspond à une case du jeu :

- **x, y** : position de la case sur le champ de bataille.
- **hit** : booléen correspondant à une case touchée d'un bateau (true si case(x,y) a été touchée par un tir).

Voici ci-dessous le diagramme de classes global.



B) Stockage de la partie (DAO)



Nous allons sauvegarder ou charger une partie sous forme de fichier XML. Tout d'abord, nous allons introduire l'interface "DAOFactory" afin d'obtenir tous les autres DAOs nécessaires. Ensuite, nous aurons la classe "XMLDAOFactory" qui va nous permettre de récupérer les DAOs liés au format XML. Ensuite, pour sauvegarder l'ensemble des éléments du jeu, nous allons découper les donnée en 2 DAO's,

- Un premier se chargeant les éléments nécessaires à instancier à une IConfiguration, nommé "IConfigDAO".
- Un deuxième s'occupant de charger une partie d'un Game, ou de sauvegarder une partie de Gamel, nommé " DAOSauvegarde".

III) IMPLÉMENTATION DU PROJET

1) Lancement du jeu

Pour compiler le programme, il faut utiliser la commande "***ant***".
Pour générer et exécuter le programme (lancer le jeu), il faut utiliser la commande "***ant run***".
Pour lancer les tests unitaires, il faut utiliser la commande "***ant tests***".

Pour supprimer les fichiers binaires et le dossier "users" (voir la partie de la sauvegarde ci-dessous), pour ne garder que les sources du programme, il faut utiliser la commande "***ant clean***".

2) Fonctionnement du jeu

Au démarrage, il y a une image d'un bateau avec un bouton pour jouer. Après avoir cliqué sur ce bouton, il y a deux cartes qui se mettent en place avec un menu. La carte de droite est celle du joueur et la carte de gauche est celle de l'IA. Dans ce menu il y a un bouton "Application" et un bouton "Jeu".
Pour commencer une partie, il faut cliquer sur le bouton "Jeu" du menu. Il y a deux types de parties : une partie personnalisée ou une partie rapide.

Partie personnalisée :

Si le joueur souhaite placer ses bateaux sur sa carte, alors il doit choisir la partie personnalisée (ou la touche ***F1***). Il choisit l'époque, la difficulté de l'IA (Aléatoire ou Intelligente). Il doit ensuite choisir les bateaux disponibles en bas, sous la carte, et les placer en faisant un **clic gauche** avec la souris (ou le pad). Par défaut, les bateaux sont placés horizontalement. Pour placer un bateau verticalement, le joueur doit faire un **clic droit** avant de faire le **clic gauche**.

Partie rapide :

Si le joueur souhaite que ces bateaux soient placés aléatoirement, alors il doit choisir la partie rapide (ou la touche ***F2***). L'époque et la difficulté de l'IA sont définis par défaut.

Une fois les bateaux placés, le joueur doit cliquer sur une case sur la carte de l'IA. Si un bateau est présent sur cette case, alors une image symbolisant un impact sur un bateau remplace l'image de base de la case choisie. S'il n'y a rien, une image symbolisant un raté remplace l'image de la case. L'IA joue directement après le joueur. Le jeu se termine lorsque le joueur ou bien l'IA n'a plus aucune case comportant un élément d'un bateau.

3) Sauvegarde/Chargement

A) Sauvegarde

À n'importe quel moment d'une partie, à condition d'en avoir commencée une, le joueur peut sauvegarder l'état actuel. Pour cela, il doit aller dans le bouton "Application" du menu et choisir "Sauvegarder Partie" (ou bien faire un ***CTRL + S***).
La sauvegarde va, en utilisant le DAO, créer un dossier "**users**" dans lequel sera crée un fichier "profil1.xml". Dans ce dernier, il sera récupéré la date et l'heure de la sauvegarde, la position des bateaux du joueur et de l'IA. Il y a également la liste des cases ciblées par le joueur ("**Joueur1**" dans le fichier xml) et par l'IA [Ordinateur] ("**Joueur2**" dans le fichier xml) avec une précision sur la réussite ou l'échec du tir.

B) Chargement

Nous n'avons pas réussi à finir d'implémenter correctement le chargement d'une partie..