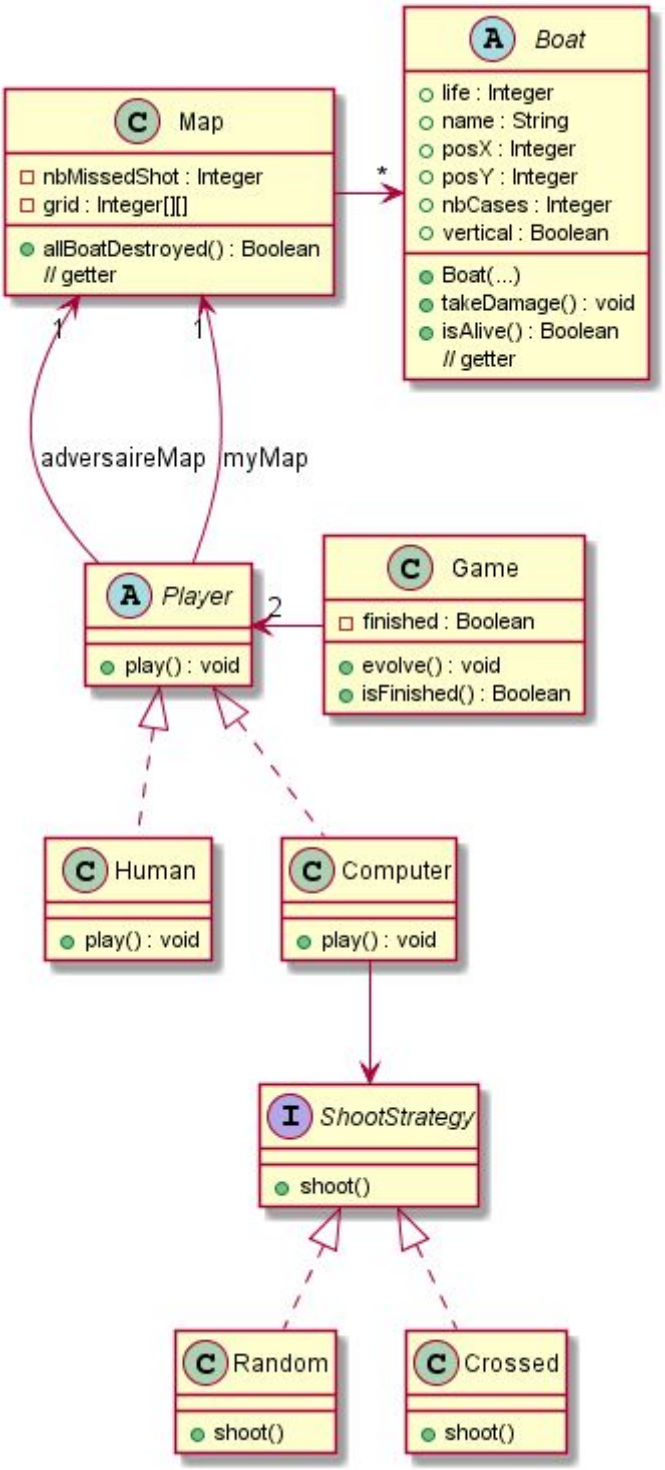
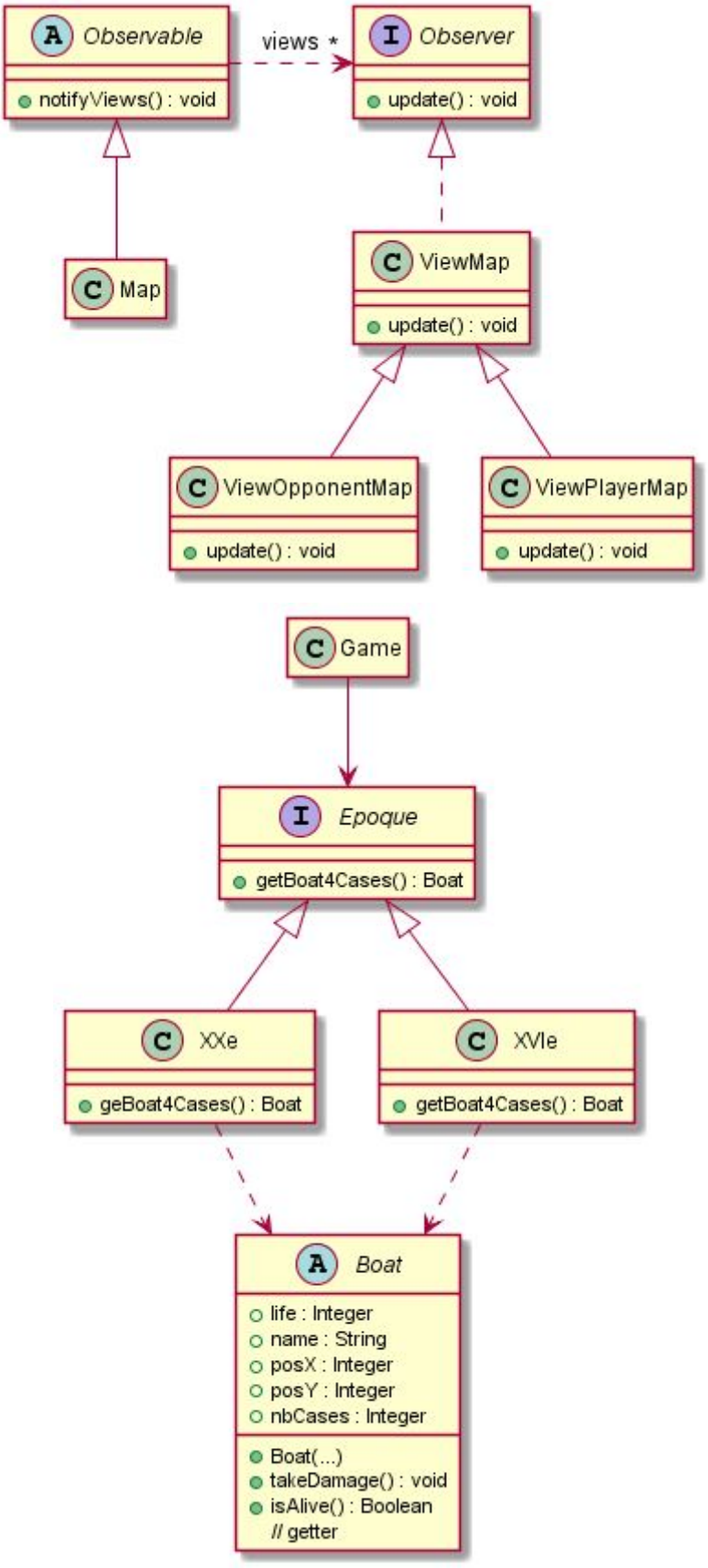
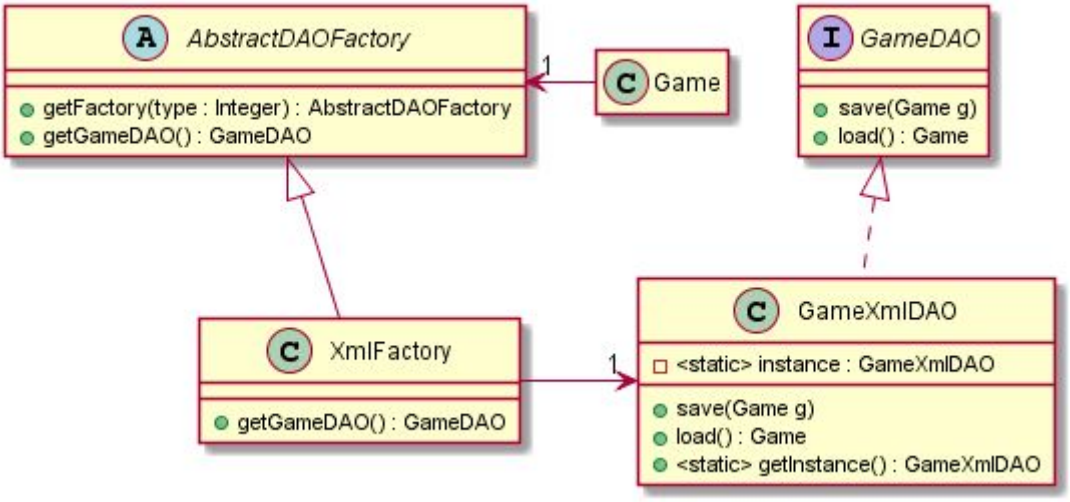
Quentin Thouvenot
Paul Merlin
Léo Martin

Bataille navale : Dossier d'analyse et de conception

UML :


Boat
life : Integer
name : String
posX : Integer
posY : Integer
nbCases : Integer
vertical : Boolean
Boat(...)
takeDamage() : void
isAlive() : Boolean
// getter
Map
nbMissedShot : Integer
grid : Integer[][]
allBoatDestroyed() : Boolean
// getter
*
1
1
adversaireMap
myMap
Player
play() : void
2
Game
finished : Boolean
evolve() : void
isFinished() : Boolean
Human
play() : void
Computer
play() : void
ShootStrategy
shoot()
Random
shoot()
Crossed
shoot()


Observable
views *
Observer
notifyViews() : void
update() : void
Map
ViewMap
update() : void
ViewOpponentMap
update() : void
ViewPlayerMap
update() : void
Game
Epoque
getBoat4Cases() : Boat
XXe
geBoat4Cases() : Boat
XVIe
getBoat4Cases() : Boat
Boat
life : Integer
name : String
posX : Integer
posY : Integer
nbCases : Integer
Boat(...)
takeDamage() : void
isAlive() : Boolean
// getter


AbstractDAOFactory
getFactory(type : Integer) : AbstractDAOFactory
getGameDAO() : GameDAO
1
Game
GameDAO
save(Game g)
load() : Game
XmlFactory
getGameDAO() : GameDAO
1
GameXmlDAO
<static> instance : GameXmlDAO
save(Game g)
load() : Game
<static> getInstance() : GameXmlDAO

Explications :

1. Les Époques

Utilisation d'une Abstract Factory pour modéliser les différentes époques que le jeu propose. Une partie possède une unique époque qui est commune pour le joueur et l'ordinateur.
Chaque classe concrète Époque est une factory permettant de créer des bateaux en fonction du nombre de cases qu'ils occupent.

Ainsi pour l'époque XVIème siècle, la création d'un bateau de 4 cases sera caractérisée par :

- un nom : Galion
- un nombre de points de vie : 1

Pour l'époque XXème siècle, la création d'un bateau de 4 cases sera caractérisée par :

- un nom : Croiseur
- un nombre de points de vie : 2

Pour ajouter une époque, il suffit d'ajouter une classe concrète qui implémente Époque en définissant pour chaque type de bateaux (nombre de cases) ses caractéristiques.

2. Déroulement du jeu

Au début d'une partie, les bateaux sont placés aléatoirement sur la carte pour le joueur et l'ordinateur.
A chaque boucle de jeu, le joueur choisit les coordonnées de son attaque en utilisant *adversaireMap* (objet de type *Map*).
La classe *Map* possède la liste des bateaux que la carte correspondante contient, ce qui permet de vérifier si un tir va toucher un des bateaux et auquel cas d'appliquer la méthode *takeDamage()* au bateau concerné.
Map hérite de *Observable* permettant de notifier à chaque modification un objet de type *Observer* chargé de l'affichage.

adversaireMap possède ainsi comme observer un objet de type *ViewOpponentMap* permettant d'afficher au joueur la carte de son adversaire en indiquant uniquement les tirs réussis/ratés (attribut grid dans *Map*).
myMap, correspondant à la carte du joueur, possède comme observer un objet de type *ViewPlayerMap* permettant d'afficher au joueur l'emplacement et l'état de ses bateaux.

Lors du tour de l'ordinateur, la méthode *play()* retourne, en fonction de la stratégie de tir utilisée, les coordonnées de l'attaque. Pour cela, nous avons choisi d'utiliser le patron de conception Strategy.

3. Sauvegarde de l'état du jeu

Pour sauvegarder/charger une partie, nous avons mis en place le design pattern DAO en choisissant pour l'instant comme format de données XML.