

# Projet Programmation Orientée Objet

## 2015 - 2016

### Informations générales

Ce projet est à réaliser par groupe de 3 étudiants. Vous serez évalués sur :

- le nombre d'étapes correctement réalisées et testées.
- l'architecture du programme : bon découpage en paquetages, bonne gestion des exceptions et bonne utilisation de l'héritage et de la bibliothèque standard.
- la lisibilité du code : respect des conventions décrites à la fin de ce document
- le respect des consignes de rendu décrites à la fin de ce document

### Description du sujet

L'objectif de ce projet est de programmer une application de bataille navale permettant d'affronter une intelligence artificielle.

La configuration de la grille de jeu et la position initiale de la flotte du joueur humain sont lus à partir de deux fichiers. La position initiale de la flotte de l'intelligence artificielle est générée aléatoirement pour chaque partie.

Le travail demandé se décompose en plusieurs étapes décrites ci-dessous.

### Étape 1 : mise en place du système de grille

#### Bateaux

Cinq types de bateaux sont disponibles : le porte-avion, le croiseur, le contre-torpilleurs, le sous-marin et le torpilleur. Chaque bateau a un type et une immatriculation. Le tableau ci-dessous liste pour chaque type de bateau le nombre de cases occupées sur la grille.

Bateau	Nombre de cases
battleship	4
patrol-board	3
submarine	3
destroyer	2
aircraft-carrier	5

#### Grille de jeu

Le fichier de configuration de la grille de jeu, nommé `grid.xml`, contient deux types d'information : la taille de la grille de jeu et le nombre de bateaux disponibles pour chaque joueur. Le code ci-dessous en est un exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<grid>
  <dimensions>
    <horizontal>10</horizontal>
    <vertical>10</vertical>
  </dimensions>
  <ships>
    <battleship>1</battleship>
    <submarine>1</submarine>
    <destroyer>1</destroyer>
    <patrol-boat>1</patrol-boat>
    <aircraft-carrier>1</aircraft-carrier>
  </ships>
</grid>
```

Par défaut, la grille de jeu est numérotée de 1 à 10 horizontalement et verticalement. Néanmoins, votre application doit pouvoir gérer une grille de jeu de taille variable, en assurant d'une taille minimale de 10 x 10. Si une taille inférieure est spécifiée, vous devez lever une erreur de type `InvalidGridException`.

Vous devez également vérifier que le nombre total des cases occupées soit inférieur ou égal à 20% du nombre total de cases. Dans le cas contraire, vous devez également lever une exception de type `InvalidGridException`.

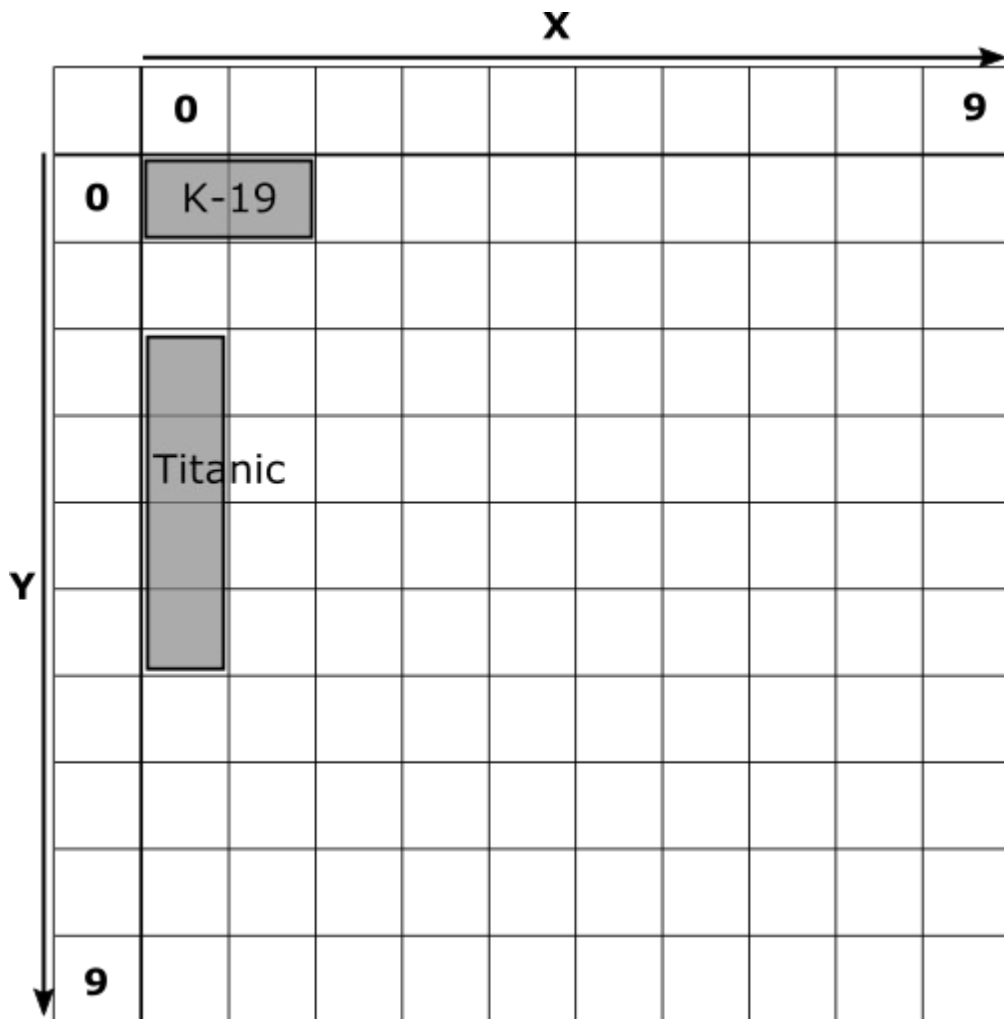
### Position initiale des bateaux

La position initiale des bateaux du joueur humain est spécifiée dans un fichier nommé `ships.xml`. Le code ci-dessous en est un exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<ships>
  <ship type="battleship" id="K-19">
    <position x="0" y="0" orientation="horizontal"/>
  </ship>
  <ship type="destroyer" id="Titanic">
    <position x="0" y="2" orientation="vertical"/>
  </ship>
</ships>
```

Chaque bateau est configuré par une balise `ship`. Dans cette balise, le type du bateau ainsi que son immatriculation sont donnés en attributs. Enfin chaque balise `ship` possède une balise fille `position` qui donne la position de départ du bateau (attributs `x` et `y`) ainsi que l'orientation.

Ci-dessous la grille correspondant au fichier de configuration d'exemple :



Au chargement d'une grille, il faut vérifier que il n'y a pas de chevauchement entre les positions de tous les bateaux (il doit y avoir au plus un bateau sur une case donnée). Si ce n'est pas le cas, il est nécessaire de lever une exception de type `ShipOverlapException`. Il faut aussi vérifier qu'aucun bateau ne dépasse de la grille. Si ce n'est pas le cas, il faut lever une exception de type `ShipOutOfBoundsException`. Enfin, il faut vérifier pour chaque type de bateau que le nombre demandé par le joueur est valide vis à vis du fichier de configuration `grid.xml`, sinon il faut lever une exception de type `ShipsConfigurationException`.

La position initiale des bateaux de l'intelligence artificielle (IA) est calculée aléatoirement. Aucune stratégie particulière n'est requise ici. Le placement des bateaux de l'IA doit respecter les contraintes énoncées ci-dessus.

### Ligne de commande

À la fin de la première étape, votre application doit pouvoir prendre en entrée deux fichiers (`grid.xml` et `ships.xml`) et générer en sortie un fichier svg (`debug.svg`). Le fichier svg doit contenir les grilles du joueur humain et de l'IA. Les grilles doivent ressembler à la grille présentée ci-dessus.

La ligne de code ci-dessous doit être utilisée pour générer le fichier `debug.svg` :

```
$ java fr.enseirb.battleship.App debug grid.xml ships.xml
```

## Étape 2 : mise en place du système de jeu

La ligne de code ci-dessous doit être utilisée pour commencer une nouvelle partie :

```
$ java fr.enseirb.battleship.App play grid.xml ships.xml
```

Le déroulement d'une partie suit le format suivant et s'effectue via le terminal :

- 1) Le système détermine quel est le premier joueur.
- 2) À son tour le joueur humain peut exécuter l'une des commandes suivantes :
  - `view` : sauvegarde dans un fichier svg le placement des bateaux du joueur ainsi qu'une vue partielle de la grille de l'adversaire avec la position des précédents tirs avec leur résultat (manqué ou touché).
  - `debug` : sauvegarde l'intégralité des grilles, de manière similaire à l'étape précédente, avec en plus la position des tirs
  - `fire x y` : tire sur la case choisie. Si la case choisie correspond à une case du bateau de l'IA, le message "Touched boat IMMATRICULATION at x y" ou "Sunk boat IMMATRICULATION at x y" s'affiche suivant si le bateau est touché ou coulé. Sinon le message "Missed at x y" s'affiche.
  - pour tout autre commande, une exception `CommandException` doit être levée et le programme doit demander à l'utilisateur de rentrer une nouvelle commande.
- 3) À son tour, l'IA joue automatiquement en tirant sur une case hasard mais jamais deux fois sur la même case. Un message correspondant à son action doit être affiché : "IA touched boat IMMATRICULATION at x y" si elle touche un bateau, "IA sunk boat IMMATRICULATION at x y" si l'IA coule un bateau et enfin "IA fired at x y and missed" si elle tire dans l'eau.
- 4) La partie se termine au premier joueur qui a coulé tous les bateaux de l'autre. Ce joueur est déclaré vainqueur. Si le joueur humain est déclaré vainqueur, le message "You win!" s'affiche, sinon "Yoo loose!" s'affiche.

## Étape 3 : mise en place d'une intelligence artificielle avancée

Dans l'étape 3 vous allez rajouter des fonctionnalités à l'IA. Vous allez programmer plusieurs stratégies de placement de bateaux et plusieurs stratégies de tir. La stratégie à utiliser dans le jeu sera spécifiée dans le fichier de configuration `grid.xml` de cette manière :

```
<?xml version="1.0" encoding="UTF-8"?>
<grid>
  <dimensions>
    <horizontal>10</horizontal>
    <vertical>10</vertical>
  </dimensions>
  <ships>
    <battleship>1</battleship>
    <submarine>1</submarine>
    <destroyer>1</destroyer>
    <patrol-boat>1</patrol-boat>
```

```
        <aircraft-carrier>1</aircraft-carrier>
    </ships>
    <strategies placement="random" firing="random"/>
</grid>
```

Les stratégies développées dans l'étape 1 et 2 se nomment "random". Vous devez rajouter les stratégies suivantes.

### Placement

- *pack* : dans cette stratégie les bateaux doivent être regroupés dans une zone.
- *far* : dans cette stratégie les bateaux doivent être éloignés les uns des autres.

### Tir

- *pack* : dans cette stratégie les tirs doivent être regroupés dans une zone.
- *far* : dans cette stratégie les tirs doivent être éloignés les uns des autres.

Une fois ces stratégies implémentées, imaginez une nouvelle stratégie de placement et une nouvelle stratégie de tir.

### Bonus

Implémenter une version à deux joueurs humains via le réseau.

## Consignes

Votre projet doit respecter les contraintes énoncées ci-dessous. Une pénalité de 2 points sera appliquée aux projets ne les respectant pas.

### Architecture

Le packaging par défaut de votre application doit être `fr.enseirb.battleship`.

Les sources doivent se trouver dans un répertoire nommé `src`, les tests dans un répertoire `test`, les bibliothèques tierces dans un répertoire `lib` et les binaires résultant de la compilation dans un répertoire `bin`.

### Code

- Votre code est en anglais.
- Votre projet contient plusieurs paquetages.
- Les noms des paquetages sont intégralement en minuscules.
- Les noms des classes et interfaces commencent par une majuscule.
- Les noms des méthodes, des attributs et des variables commencent par une minuscule.
- Le nom des variables a un sens.
- Utilisez le CamelCase (<https://fr.wikipedia.org/wiki/CamelCase>) pour les noms des classes, interfaces, méthodes et attributs.
- Les attributs ont une visibilité `private` ou `protected`. Des accesseurs permettent si nécessaire d'accéder en lecture et/ou écriture aux attributs.
- La visibilité des méthodes dépend de leur utilisation.
- Les constructeurs sont définis uniquement s'ils sont nécessaires.
- Le code des méthodes compliquées de votre projet doit être commenté.

## Livrables

Une première évaluation du travail des étudiants sera réalisée lors de la dernière séance avec l'encadrant. À cette échéance, au moins les deux premières étapes devront avoir été finalisées. Ainsi, chaque groupe devra réaliser une démonstration de son application de bataille navale. De plus, chaque groupe devra présenter en détail l'état d'avancement de son projet et justifier l'ensemble des choix de conception.

Dans un second temps, il sera demandé aux étudiants de fournir une archive contenant le code source du programme. À la racine de l'archive, il doit y avoir un fichier **README** spécifiant les étapes qui ont été complétées et celles qui ne l'ont pas été.

Le schéma de nommage de l'archive doit être le suivant (pénalité de 3 points en cas de non respect) : **pg220-2015-LOGINS.zip**, où LOGINS est la liste des logins des membres du groupe, classée par ordre alphabétique et séparés par le symbole\_.

L'archive est à déposer, avant le **4 janvier 2016** à 23h, via la plateforme <http://moodle.ipb.fr>.