Bataille navale mobile

Objectif de ce mini-projet

L'objectif de votre projet est de réaliser un jeu de bataille navale mobile dans le langage orienté objet Java, en mode console, puis éventuellement graphique.
Vous jeu devra permettre à un joueur humain d'affronter votre ordinateur (intelligence artificielle).

Pour passer plus facilement du mode console au mode graphique, vous devrez définir une organisation modulaire multi-fichiers respectant l'approche **Modèle-Vue-Contrôleur :** Modèle Vue Contrôleur (wikipedia) et Adopter une architecture MVC (openclassrooms). Votre diagramme de classes devra montrer ce découpage modulaire : encadrer avec 3 couleurs différentes les classes faisant partie du Modèle (couleur 1), de la Vue (couleur 2) ou du Contrôleur (couleur 3)

Cahier des charges du déroulement et des règles du jeu

1) Jeu en mode console (16 points)

Le menu d'accueil

Une fois lancé, le jeu proposera un menu classique permettant de réaliser les actions suivantes :

1. Jouer une partie
2. Charger une partie
3. Aide (doit expliquer clairement les règles du jeu et les touches du clavier à utiliser lors d'une partie)
4. Quitter

Caractéristiques du jeu

Le joueur et l'ordinateur disposent chacun de deux grilles de 15*15 cases :

- Une grille n°1 pour positionner et visualiser ses navires
- Une grille n°2 pour visualiser les dégâts causés à l'adversaire

Chaque joueur possède une flotte de 10 navires : 1 cuirassé, 2 croiseurs, 3 destroyers et 4 sous-marins

Caractéristique des navires

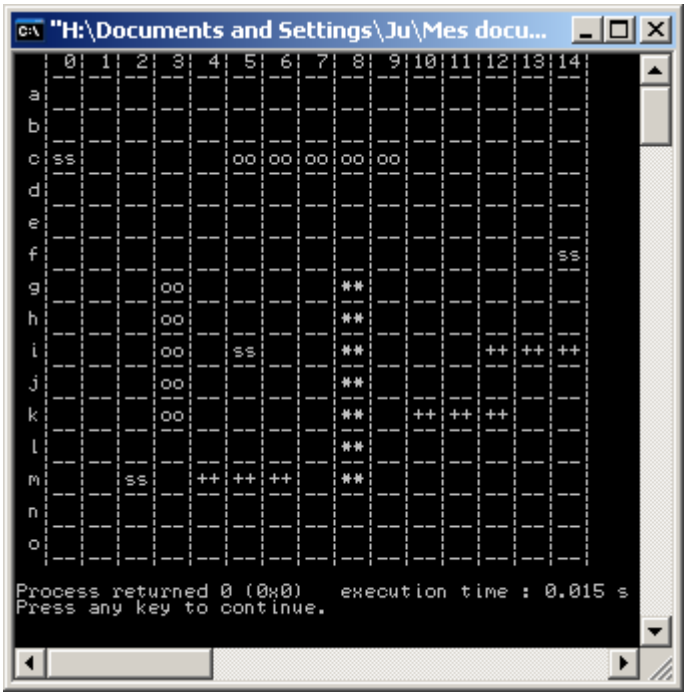
- Chaque type de navire est de taille différente :

type	taille
cuirassé	7 cases
croiseur	5 cases
destroyer	3 cases
sous-marin	1 case

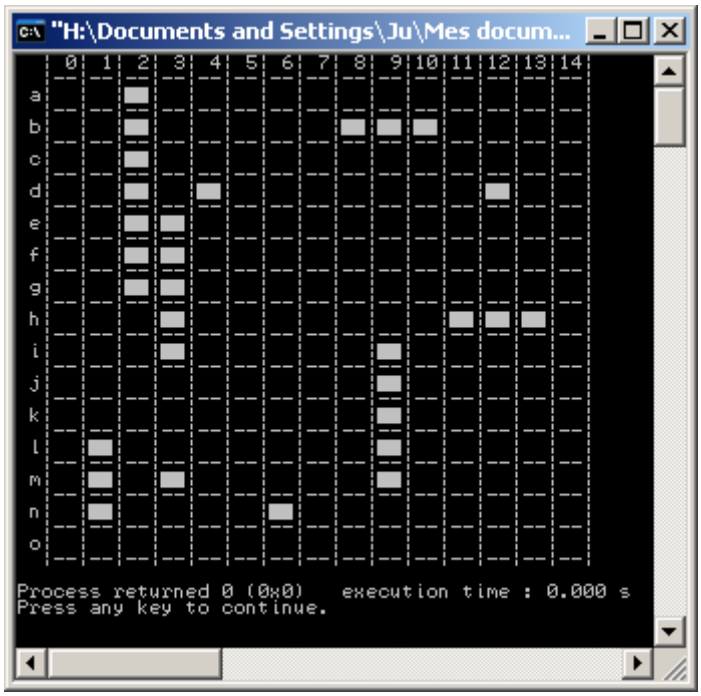
- Le positionnement des navires :

En début de jeu les navires sont positionnés aléatoirement et bien répartis dans la grille N° 1 de chaque joueur (voir ci-dessous des exemples). **C'est le joueur humain qui commence la partie**. Les navires peuvent être positionnés verticalement ou horizontalement sur la grille. Mais attention, deux navires ne peuvent occuper la même case.

Positionnement des navires sur la grille n°1 du joueur humain



Positionnement des navires sur la grille n°1 de l'ordinateur



- La puissance de tir (nombre de cases touchées autour du point d'impact) dépend du type de navires :

type	Puissance de tir
cuirassé	9 cases
croiseur	4 cases
destroyer	1 case
sous-marin	1 case

Les actions d'un navire par tour de jeu et par joueur

Chaque joueur (humain et ordinateur) joue à tour de rôle. Bien entendu, un joueur ne doit pas voir les grilles de son adversaire : seules les grilles du joueur en cours sont affichées.
A chaque tour de jeu, les joueurs peuvent choisir l'une des 2 actions suivantes sur un seul navire de leur choix de la grille n° 1, en choisissant les coordonnées de l'une des cases du navire :

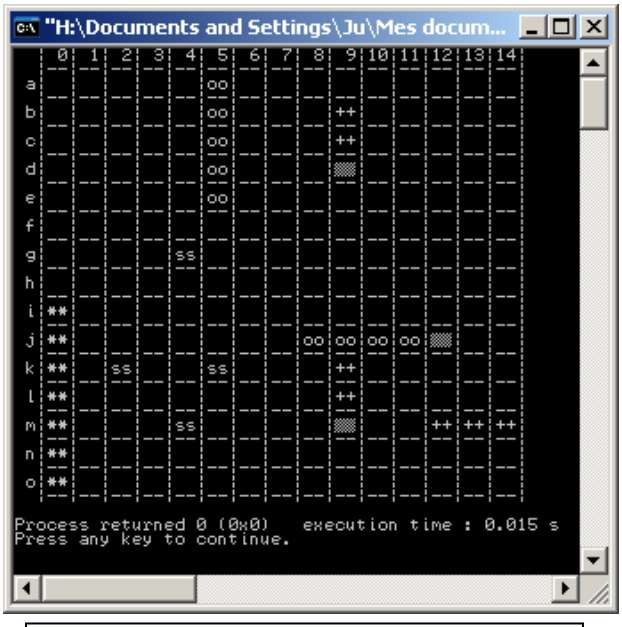
1) Tirer. Le joueur tire en choisissant les coordonnées d'une case valide de l'adversaire (grille n° 2). Chaque destroyer n'est muni que d'une seule fusée éclairante. Le premier tir d'un destroyer dévoile un carré de 4*4 cases dans la grille adverse à partir du coin haut et gauche. Mais attention, les navires adverses de ce carré ne seront visibles que lors du tour du jeu (quelques secondes). Pour couler un navire, il faut avoir touché toutes les cases qu'il occupe : un cuirassé est plus résistant qu'un simple destroyer, les sous-marins ne peuvent être coulés que par d'autres sous-marins. Dans ce cas, le navire adverse coulé disparaît et la case touchée s'affiche sur la grille n° 2.

2) Déplacer le navire d'une seule case sauf s'il est touché. Un navire ne peut pas se déplacer en diagonale. Il ne peut se déplacer que dans sa direction horizontale ou verticale, et d'une seule case à la fois, et le point cardinal de déplacement (est ou ouest s'il est horizontal, nord ou sud s'il est vertical). Evidemment, en cas d'obstacle (case de déplacement occupée par un autre navire ou le bord de la grille), un navire ne pourra pas effectuer son déplacement et devra tenter une nouvelle action.

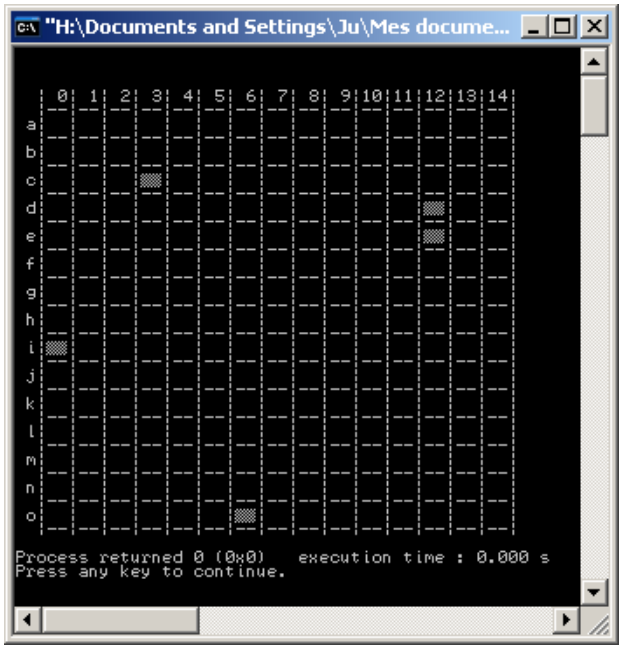
A chaque tour de jeu, il faudra visualiser l'action, les coordonnées et le type du navire choisis par l'ordinateur.

Visualisation des 2 grilles

Les captures d'écran ci-dessus sont données à titre d'exemples. A vous d'imaginer comment visualiser au mieux vos navires et les dégâts subis ou causés à l'adversaire.



Exemple (grille n°1) : En d9 et m9, 2 destroyers ont été touchés. En j12, un croiseur a été touché.



Exemple (grille n°2) : En i0, c3, o6, d12 et e12, des navires adverses ont été touchés.

Victoire, sauvegarde et chargement d'une partie

Le premier qui a coulé toute la flotte de son adversaire a gagné la partie !

A chaque instant, le joueur peut quitter la partie en cours avec une touche du clavier. La partie est alors sauvegardée dans un fichier texte. La sauvegarde comprend la position et le type de tous les navires restant et coulés des grilles.

Pour charger une partie, il faut passer par le menu principal et choisir "Charger une partie".

2) Jeu en mode graphique (4 points)

Dans ce mode graphique, il vous fera reprendre vos méthode d'affichage dans le module de **Vue**. De même dans le module **Contrôleur**, vous devrez faire en sorte que l'utilisateur puisse agir sur un navire (tirer ou déplacer) avec la souris grâce à l'utilisateur de *Listeners* (écouteurs).

Planning et organisation

Période de réalisation du projet : de la semaine du 3 mai 2020 à la semaine du 24 mai 2021 incluse. Soutenance dans la semaine du 31 mai 2021.

Evaluation : La version finale sera présentée à la soutenance du projet qui donnera lieu à la note de projet.

Un diagramme de classes doit être présenté pour montrer la conception objet, y compris en montrant l'approche **Modèle-Vue-Contrôleur** (MVC) comme indiqué dans les Slides de présentation.

Les grandes étapes à respecter

1- Analyse et conception générale du diagramme de classes.

- A partir du cahier des charges (CDC) : extraire les données pertinentes, les regrouper en grandes entités / objets, spécifier et caractériser les attributs et les fonctionnalités de chaque objet, identifier les interactions entre les différents objets ainsi que les différents scénarios possibles, …
- En déduire le **diagramme de classes**, mettant en relation les classes en y intégrant si possible de **l'héritage et du polymorphisme**. Pour chaque classe, définir les attributs (en général *private*, ou *protected* en cas d'héritage), et les méthodes (inutile d'y mentionner, les constructeurs, les getters et les setters).
- IHM : lister les choix à offrir au démarrage, lister les évènements à gérer, déterminer l'organisation et le contenu de l'écran de jeu (maquette), …
- Rédiger une Analyse Chronologique Descendante (ACD) du programme principal.
- Répartir les tâches au sein de l'équipe.

2- Analyse et conception détaillée

- Pour chaque classe, lister les prototypes de toutes les méthodes requises en précisant ses paramètres d'entrée et de sortie.

- Réaliser progressivement une maquette du jeu en testant au fur et à mesure du codage et en tenant compte des différents scénarios.

- Entre autres critères de qualité, le programme final devra être très facilement adaptable par tout autre développeur (exemples : changement des valeurs d'initialisation, changement des caractères et couleurs d'affichage, …).

3- Développement dans un langage objet (codage, tests).

Dans le langage objet **Java**, implémenter le jeu en respect de votre analyse des 2 étapes précédentes : **diagramme de de classes**, organisation modulaire multi-fichiers selon l'approche **Modèle-Vue-Contrôleur**, avec **héritage et polymorphisme**, commenter les prototypes des méthodes (fonctionnalités) en précisant les paramètres d'entrée/sortie et commenter aussi l'ACD de ces méthodes et du programme principal.

4- Critères de notation.

Vous devez réaliser l'ensemble du cahier des charges de base (expliqué dans la paragraphe précédent). Le langage de programmation doit être le langage objet Java avec héritage... Votre code devra être modulaire, respecter les interfaces en mode graphique et bien commenté !

Un code qui ne compile pas ou qui plante au démarrage ne vaut pas plus de 10/20. Tester donc votre programme avant de le déposer sur Campus...

Votre travail sera jugé sur les critères suivants :

- Le respect rigoureux des règles du jeu énoncé précédemment (CDC)
- La modularité de votre conception et donc de votre code
- La bonne répartition des tâches entre les membres de l'équipe
- L'intérêt, l'originalité, la jouabilité et toutes les caractéristiques que vous prendrez soin de mettre en avant lors de la soutenance.

5- Livrable, deadline du livrable et Soutenance

Le livrable est à transmettre par mail à **jean-pierre.segado@ece.fr** **au plus tard le lundi 31 mai 2021**.

Ce livrable doit contenir les 2 compte-rendu suivants de conception (PowerPoint) et de la version finale du code :

1) Slides de présentation (PowerPoint ou Prezi)

- Page de garde avec titre, noms coéquipiers et groupe de TD (1 slide)
- Sommaire (1 slide)
- **Diagramme de classes** avec l'outil Draw.io ou équivalent, selon le pattern MVC et présentant les attributs (pas d'attribut objet !), les méthodes, les cardinalités, MAIS sans constructeurs ni getters/setters (1 slide ou plus si le diagramme est illisible : par exemple, 1 slide présente l'architecture générale du pattern MVC avec seulement les noms des classes et les relations inter-classes, puis un slide pour chacun des 3 types de modules MVC détaillant le contenu des classes).
- **Design de la maquette de votre interface graphique** principalement composée des 2 éléments suivants :
  - Le *storyboard* : liens entre les pages, symbolisés par une flèche pour naviguer d'une page à une autre (1 à 2 slides)
  - Des *wireframes* de certaines de vos pages : composants graphiques Swing légendés avec les conteneurs encadrés, leur mise en page layout (au plus 3 slides)
- **Bilan** sans blabla (par exemple sous forme de tableau) sur l'état du travail effectué, des compétences acquises et des points d'amélioration. (1 à 2 slides).
- **Sources** : web avec les liens, livres, supports de cours en citant les auteurs. Toute source non citée est considérée de facto comme un plagiat. (1 slide)

2) Version finale du code avec tous les dossiers et fichiers nécessaires

- Dans une archive au format **zip** ou **rar en indiquant bien les noms des coéquipiers** :
  - Tous les dossiers et fichiers du projet développé sur NetBeans (fichiers sources .**java**, exécutable **.jar**, etc.) ;
  - Tout autre fichier jugé nécessaire au bon fonctionnement de votre programme (par exemple, si besoin fichier(s) de sauvegarde, images etc.).
- **Citez vos sources** (exemples : extraits de code récupérés de cette page campus, sites web consultés etc.) en commentaires en début de vos fichiers sources .java

Tout retard de livraison sera pénalisé à raison de -2 points jour de retard. Tout fichier source manquant sera sanctionné par 0. Tout plagiat est rigoureusement sanctionné par 0 et un avertissement.

Le PowerPoint et la démo du code livré seront présentés lors d'une soutenance la semaine du 31 mai 2021.