DEVOIR DE CONTROLE CONTINU

MIM4A1

A RENDRE POUR LE VENDREDI 23 AVRIL 2021

UNIVERSITÉ CAEN NORMANDIE

Rappel : le plagiat et l'échange de code sont interdits et passibles de sanctions.

1. Organisation

Ce devoir de Contrôle Continu est à réaliser en groupes de 4 étudiants. En cas de nécessité, des groupes de 3 étudiants pourront être acceptés, mais en aucun cas des groupes de plus de 4.

Une partie du temps de travail de TP (voire l'intégralité des 4 derniers TP) pourra y être consacrée, selon les indications de vos enseignants respectifs, mais du travail hors des heures de TP sera indispensable pour mener à bien ce devoir.

Un dépôt devra être créé sous subversion pour chaque groupe (obligatoirement comme un sous-projet du projet "TP Compléments Objet L2"). Les noms court et long du dépôt devront comporter les noms des quatre membres du groupe, selon l'exemple suivant :

- nom court "durand-dupont-smith-doe"
- nom long "Génie Logiciel Durand, Dupont, Smith, Doe".

D'autre part, Céline Alec, Christophe Charrier, Yohann Jacquier, Olivier Ranaivoson et Yann Mathet devront être ajoutés comme **managers** du dépôt. Si ces points ne sont pas respectés, le devoir ne sera pas corrigé (note de 0).

Le 23 avril au soir (minuit), le code de chaque groupe sera extrait de son dépôt pour correction : un répertoire nommé *livraison* devra être présent à la racine et contenir :

- un répertoire *src* contenant le code commenté
- un répertoire *doc* contenant la Javadoc
- un répertoire *dist* contenant l'exécutable (un fichier .jar, et d'éventuelles ressources nécessaires à l'exécution)
- Et enfin un répertoire *rapport* contenant un rapport au format PDF contenant toute information utile à la compréhension de votre conception logicielle. Par exemple quelques diagrammes de classes précisant votre mise en oeuvre de MVC, le principe de vos algorithmes (mais pas directement le code), etc. Ce rapport pourra faire aux environs de 5 à 7 pages bien illustrées (ou plus si nécessaire).

2. Critères d'évaluation

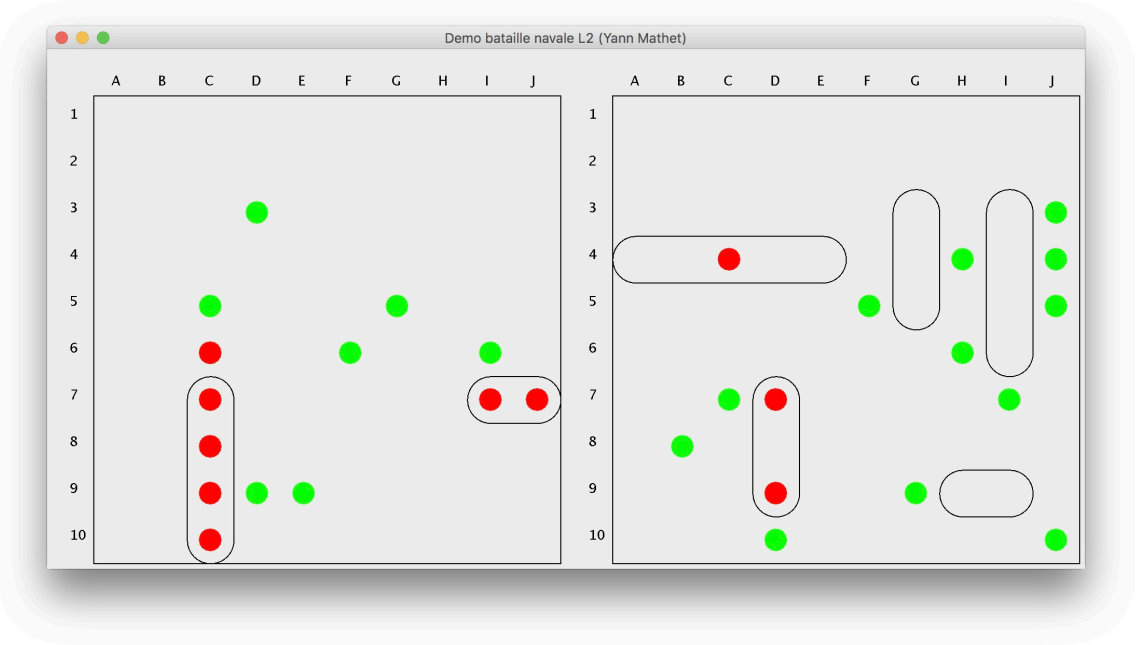
En premier lieu, nous prendrons en compte la qualité de l'architecture logicielle et de votre code. Nous regarderons notamment si votre conception est :

- facile à comprendre (répartition en packages et en classes cohérente, noms de classes et de méthodes éloquents, commentaires dans le code lorsque c'est utile)
- facile à maintenir et à faire évoluer (pas de code spaghetti, pas de code redondant, mise en oeuvre de MVC)
- robuste (tests effectués)

Bien sûr, une bonne ergonomie, un design agréable, des options supplémentaires seront appréciés mais ne constitueront pas l'essentiel de la note. En particulier, une application qui ferait ce qui est demandé dans le sujet mais qui ne répondrait pas aux critères d'évaluation exposés ci-dessus n'obtiendrait assurément pas la moyenne.

3. Sujet : Jeu de Bataille Navale

Le but de ce devoir est de réaliser une application de jeu, dotée d'une interface graphique, (mais pouvant être utilisée sans l'interface graphique) qui consiste en une bataille navale, comme illustré par exemple ci-dessous (à gauche, les tirs du joueur humain, qui ne voit les bateaux adverses apparaitre que lorsqu'il a réussi à les couler, ici 2 sur les 5) :



Chacun des deux joueurs dispose d'une flotte (ensemble de navires) positionnée sur une grille (représentant une portion de mer en 2 dimensions). Les joueurs tirent chacun à leur tour sur une position du camp adverse. Si un bateau adverse est impacté, le tir apparaît d'une façon spécifique (rouge dans l'illustration) par rapport à un tir raté (vert dans l'illustration). Le gagnant est le premier joueur parvenant à couler l'ensemble de la flotte adverse.

Nous souhaitons une réalisation intégralement MVC, avec un modèle totalement indépendant de l'interface graphique. En particulier, le jeu devra être jouable en ligne de commande (par affichage sous forme de System.out.println(...) et contrôle au clavier). Voici par exemple un affichage sur la sortie standard correspondant à la partie gauche de l'illustration précédente :

```
  !

!   !
X  !  !
X     XX
X
X!!
X
```

La partie interface ne constituera donc, comme toujours en MVC, qu'une partie Vue-Contrôleur supplémentaire venant se greffer sur le modèle.

L'évaluation prendra en compte, outre les aspects MVC, les efforts qui auront été faits pour diminuer la complexité algorithmique. Par exemple, lors d'un tir, si l'on fait un parcours de tous les bateaux de la flotte pour voir s'ils sont impactés, la complexité est relativement importante (d'autant plus importante que les bateaux sont nombreux), alors que l'utilisation d'un tableau à deux dimensions peut permettre de trouver l'impact éventuel en une seule opération.

On proposera de faire jouer l'humain contre l'ordinateur. Ce dernier pourra jouer de façon aléatoire, mais là aussi on essaiera de diminuer la complexité (éviter de tirer plusieurs fois au même endroit).

Ces aspects algorithmiques pourront être mis en avant et détaillés dans le rapport.

Concernant l'interface graphique, on proposera de faire jouer le joueur humain à la souris, en cliquant directement sur la position de son choix. Chaque tir touchant un bateau apparait en rouge (le fameux « touché »), et les tirs ratés apparaissent en vert. Lorsqu'un bateau adverse est coulé, il apparait explicitement de sorte à ce que le joueur en soit averti (le fameux « coulé » que l'on annonce à l'adversaire).
L'illustration ci-dessus est donnée à titre indicatif, la réalisation graphique est laissée libre.