Projet Bataille navale

Présentation

Le projet consiste à programmer un jeu de bataille navale jouable en mode console avec divers contraintes : taille de grille variable, mode mono-joueur, listes triées de bateau, sauvegarde et reprise de partie, mise en œuvre des concepts du cours (PO, Listes, Exceptions, Entrées/sorties).

Ce projet est très différent des projets que vous avez l'habitude de faire dans le sens ou vous serez en autonomie complète. Nous ne vous donnerons pas une suite d'instructions pour construire le projet. Les consignes à respecter sont celles données dans ce document et éventuellement les mises à jour faites suite à vos remarques.

Contraintes

Taille de grille variable :

La grille de jeu est un rectangle de taille quelconque (entiers), choisi au démarrage de la partie. En option, certaines cases pourront ne pas exister (voir *options et extensions*)..

Mono-joueur :

Une bataille doit pouvoir être jouée contre soi-même. Ce mode évite la difficulté de créer un adversaire virtuel (intelligence artificielle) contre lequel jouer. Le joueur dispose de 2 grilles : une pour placer ses bateaux, l'autre pour tirer. Cependant, dans ce mode simplifié et dégradé le joueur tire en fait sur ses propres bateaux. Le mode multi-joueur ou avec adversaire virtuel sera traité en option. Bien que moins intéressant à jouer, ce mode nécessite de mettre en œuvre tous les principes et concepts du jeu de la bataille navale nécessaires pour la suite (*options ou extensions*).

Listes triées de bateau :

On doit pouvoir afficher la liste des bateaux triée par taille et aussi par pourcentage d'impact (2 tris distincts). Il s'agit ici d'une contrainte technique dont le but est le tri d'ArrayList. Mais d'autres tris sont aussi possibles et si certains vous semblent pertinents, ils peuvent s'ajouter ou remplacer ceux proposés ici.

Sauvegarde et reprise de partie :

Une partie en cours doit pouvoir être sauvegardée dans un fichier afin d'être poursuivie plus tard. La sauvegarde se base sur l'écriture d'objets dans un fichier binaire.

Mise en œuvre des concepts du cours :

La réalisation se base sur la programmation objet (cases et bateaux sont des objets ainsi que jeu et grille). Elle doit mettre en œuvre les concepts du cours : Il doit y avoir des listes (List, ArrayList avec tri), des exceptions (exceptions personnalisées remontées et attrapées) et des entrées/sorties (sauvegarde et reprise de partie en cours, usage de Scanner et affichage formatés dans la console)

Options et extensions :

Cases vides :

En option, certaines cases de la grille peuvent être vide (nulle, donc non utilisable, ni bateau ni tir possible, telles des terres, îles ou continents), permettant des grilles de formes « quelconques »

(carte ou autre pixel-art, à choisir au démarrage ou à concevoir soi-même = éditeur de carte).

Multi-joueur et adversaire virtuel :

En mode multi-joueurs, il faudrait plusieurs écrans pour cacher ses bateaux ou bien n'afficher que les grilles de tir après que chaque joueur ait choisi secrètement la position de ses bateaux.
On préférera donc au mode multi-joueurs le mode adversaire virtuel. L'adversaire virtuel est une intelligence artificielle capable de placer ses bateaux puis de tirer pour gagner. Plusieurs stratégies sont possibles et on peut aussi imaginer faire jouer entre elles plusieurs intelligences artificielles pour sélectionner la meilleure...

Affichage graphique :

En plus de la sortie console, il est possible d'ajouter au projet un affichage graphique mais ceci n'est pas prioritaire et ne doit pas être abordé tant que toutes les contraintes n'ont pas été réalisées.

Barème :

Le barème est donné à titre indicatif et peut changer :

- Code fonctionnel (5 points)
- Respect des règles du jeu (4 points)
- Diagramme UML au format image et lisible (2 points)
- Une javadoc complète (2 points)
- Des tests unitaires (3 points)
- Mise en œuvre des concepts du cours : (4 points)
  - Des collections
  - Des exceptions
  - Des entrées et sorties fichiers
  - Des classes abstraites voir des interfaces
  - Du formatage pour faire de belles sorties consoles

Ne pas respecter les consignes suivantes peut retirer des points :

- La classe Main ne doit pas être trop longue
- L'usage d'un dépôt git public est obligatoire avec historique et push réguliers.

Déroulement

Pour réaliser ce projet, vous disposerez de 6h de TD à la fin du module ainsi que le temps que vous y accorderez en dehors de ces derniers. Les deux premières heures de cours seront essentiellement des discussions autour de la conception du jeu. A l'issue de ces deux heures, vous devrez avoir proposé un diagramme UML ainsi qu'un diagramme de séquence. Même si ce n'est pas la partie la plus joyeuse du projet, c'est sans doute la partie la plus importante car si vous n'avez pas bien pensé votre architecture ou les relations entre les différentes classes, faire tourner le jeu sera complexe. Durant les 4 heures de TD restantes, nous seront à votre disposition pour répondre à vos questions et vous débloquer en cas de problème.

A l'issue de ce projet, nous vous demanderons de rendre l'ensemble des sources.

Le TP est à rendre, individuellement ou par binôme, le dimanche 7 avril à 23h55 AU PLUS TARD. AUCUN retard ne sera autorisé, le dépôt sera automatiquement fermé au delà de la date et de l'heure indiquées.

Format à suivre OBLIGATOIREMENT pour le nom du fichier à déposer :

Le nom de votre fichier, si vous avez travaillé seul : S2A_NOM.zip (ou S2B, S2C, S2D selon votre groupe)

Le nom de votre fichier si vous avez travaillé en binôme : S2A_NOM1_NOM2.zip (ou S2B, S2C, S2D selon votre groupe)