Documents et conception – BattleShipLike

Objectifs – Concevoir et développer une application *console* client-serveur en utilisant la méthode de communication par socket synchrone

- Votre application devra permettre de jouer au jeu de BattleShipLike en réseau (*2 joueurs*). (*à tour de rôle*).
- Vous devrez pouvoir jouer sur 2 ordinateurs différents* (*application serveur sur un, le client sur un autre*). ***Pour des raisons de sécurité réseau, ceci n'est pas nécessairement possible!***
- Vous devrez utiliser les mécanismes appropriés afin d'effectuer les communications entre le client et le serveur.
- Vous devrez permettre la saisi des informations afin d'établir correctement la connexion entre le client et le serveur (*adresse IP et port du serveur pour la connexion*).

Date de remise (*Projet final*)

- À remettre au plus tard ***Vendredi le 12 Septembre à 23h59***

Dates de remise intermédiaires (*Livrables partiels*)

- Plan de développement ***Mercredi 3 septembre à 14h59***
  - Analyse des besoins, calendrier, planification des tests.
  - Diagrammes : cas d'utilisation, de séquences, de classes.
- Tests unitaires principaux ***Mercredi 3 septembre à 14h59***

Remise

- L'ensemble des documents de conception et d'analyse
- Remettre sur LÉA un document word contenant :
  - le lien GIT de votre dépôt Git
  - les noms des coéquipiers
  - La liste (titre) des documents produits
- Il est **fortement suggéré d'utilisé GIT** (*adéquatement*) tout au long de votre développement, pas seulement pour déposer le produit fini!

Directives générales

- Ce travail doit être réalisé en ***équipe***
  - Utilisez les mêmes équipes que celle du travail d'hier
- La solution sera compilée sur un environnement standard VS2022
  - **Une solution qui ne compile pas = programme non fonctionnel = ZÉRO**

Critères d'évaluation

- Fonctionnement du programme et qualité de la documentation (3)
- Utilisation adéquate des concepts de la POO, validations, gestion d'exceptions et tests (1)
- Modularité, structure et architecture du code (1)

Consignes / Contraintes

Vous devrez respecter les consignes et contraintes suivantes :

- Vous devez utiliser les sockets synchrones. (Vous pouvez réutiliser votre projet effectué en ***Intro Services de Données*** *en adaptant le tout pour les besoins du présent travail)*
- Votre programme doit gérer adéquatement (*autant que possible*) les exceptions et effectuer les validations nécessaires au bon fonctionnement des traitements du programme. Exemple :
  - Perte de connexion réseau;
  - Client qui ferme son application sans demander/valider avec le serveur;
  - *Pas besoin pour l'instant de gérer du côté client le serveur qui fermerait.*
- Vous devez ***programmer*** vos implémentations ***en utilisant les séquences produites en équipe.***
- Le serveur est en attente continuelle d'une connexion client tant qu'une partie n'a pas été débutée (*suite à la connexion d'un client*).
- Une fois la connexion établie entre un client et le serveur, les 2 joueurs devraient pouvoir effectuer autant de partie qu'ils le désirent et ce **sans rétablir une nouvelle connexion.**
- Votre programme devra permettre la connexion client/serveur peu importe où les programmes client et serveur sont exécutés*. (*Paramétrer les informations « hôte » avant d'établir la connexion*). ***Attention aux chemins d'accès (relatifs vs absolus), emplacement de l'exécutable, etc…***
- **Aucun des 2 joueurs** (*client ou serveur*) **ne peut prendre unilatéralement une décision**.
  - Vous devez donc valider que l'autre joueur n'a pas **« triché ».**
- Lorsque le client décide de cesser de jouer (*après une partie*) le serveur se remet en mode attente (*d'une nouvelle connexion*).
- Il ne peut y avoir qu'une seule connexion client concourante sur le serveur.

Informations importantes en lien avec les consignes

- Ne pas confondre *«* Gestion d'erreur » et validations.
- Éviter autant que faire se peut, l'utilisation de break et continue.
  - ***Toute utilisation devra être justifiée***
- Minimiser et standardiser les informations transmises entre le client et le serveur.
  - Vous devez utiliser le principe de ***sérialisation*** vue la/les sessions précédentes

Livrable final du mini-projet #2

- Vous devez livrer tout le nécessaire afin d'avoir une application fonctionnelle client-serveur afin de jouer au BattleShipLike.
- Après le projet, vous devrez présenter à l'enseignant votre projet et les différents documents de conception.
- Votre projet devra contenir un projet de test afin de tester différents éléments de votre projet;
    - Vous devriez mettre en place des tests qui valident les éléments cruciaux de vos implémentations.

Exigence du programme (règle du jeu) - BattleShipLike

- La grille de jeu est de 4x4 (*taille fixe pour ce projet*);
- Chaque joueur à 1 (*un*) seul bateau de taille 1x2
    - Les 2 «cases» occupées par le bateau doivent être contiguës (*collées*)(***Pas de diagonale***) :
- Chaque joueur va jouer à tour de rôle et pourra attaquer 1x par tour;
- Que le coup ait touché un BalleShip ou non, le tour passera à l'autre joueur;
- Vous devez mettre à jour votre interface visuelle immédiatement après avoir envoyé votre « attaque » :
    - Vous ne pouvez pas attendre « l'attaque » de l'autre joueur pour mettre à jour votre affichage;
    - Ceci aura donc des impacts sur la mécanique (*les échanges/communications*);
- La partie termine lorsqu'un joueur a touché les 2 emplacements du BattleShip de l'adversaire (*bateau coulé*);
- Le client (*qu'il gagne ou perde la partie*) peut décider de jouer une autre partie ou de quitter le jeu ;
    - Le serveur rejouera également si le client veut rejouer ou se remettra en attente si le client quitte.