PROJET BATAILLE NAVALE

L'objectif de ce projet est d'utiliser les notions vues en cours sur les threads et les Sockets afin de réaliser une bataille navale entre plusieurs clients.

Lors de la réalisation de ce projet j'ai choisi l'option où le serveur s'occupe de tout gérer et le client se contente juste d'afficher. Dans ma solution, tout fonctionne à part la partie sur la déconnexion subite d'un joueur. Il n'y a pas d'erreur affiché mais la partie continue pour l'un des joueurs sans qu'il puisse gagner ou perdre.

Point Important avant de commencer, vous pouvez réduire le nombre de bateaux en mettant en commentaire certaines lignes de la fonction `boatAplacer()` dans la classe Joueur pour gagner du temps. Ensuite, pour placer un bateau, il faut rentre la coordonnée de son extrémité sous la forme « B7 » par exemple et ensuite dans un deuxième temps choisir sa direction entre Horizontal et Vertical en rentrant « true » ou « false ». Par défaut, pour toute input non conforme, la commande retenue est false. Finalement pour tirer il suffit de rentrer une casse sous la forme « H2 ».

Le fonctionnement du programme se fait à l'aide de threads. Dans un premier temps quand deux personnes se connectent au serveur elles rentrent dans une partie en lançant le thread ThreadBataille. Ensuite, la partie se lance en lançant 2 nouveaux threads qui permettent aux deux joueurs de placer simultanément leurs bateaux. Finalement, la partie au tour par tour est lancé jusqu'à ce qu'un des deux joueurs ait coulé tous les bateaux.

Le principe de shoot se fait en deux parties : la première va appeler `getShoot()` de la classe Joueur sur le joueur qui reçoit et en fonction de la réponse cela va, dans une deuxième temps, être transmis au Joueur émetteur pour qu'il remplisse sa grille.

La partie la plus compliquée dans mon programme se passe dans la classe Joueur qui gérer les grilles et la liste des bateaux. Je vais décrire chacune des méthodes présente dans cette classe :

-`emptyGrid()` : Renvoie une grille vide pour l'initialisation.

-`boatAplacer()` : Liste des bateaux à placer lors du placement.

-`DeleteBoatAplacer()` : Supprime les bateaux de la liste boatListAplacer.

-`resetGrid()` : Réinitialise les grilles avec des grilles vides.

-`setBoat(String pos, int length, boolean direction)` : Place un bateau sur la grille et revoie si oui ou non le bateau a pu se placer correctement. Cette méthode fait appel à `isBoatValid()`.

-`isBoatValid(String pos, int length, boolean direction)` : Renvoie si le bateau peut être placé à une position donnée.

-`getShoot(String pos)` : Retourne un bool {isValid, isTouch, isSunk} et place une indication sur ownGrid en fonction de si un bateau touché ou non.

-`shoot(String pos, boolean touch)` : Modifie la grille guessGrid pour marquer un tir touché ou non.

-`isLoose()` : Renvoie true si tous les bateaux du joueur sont coulés.

-`gridPosition(String pos)` : Renvoie un tableau d'int de coordonnées correspondant aux input de type « A2 ».

La classe Boat contient des fonctions assez simples pour placer des bateaux en calculant la position automatiquement ou encore pour savoir si un bateau est touché ou coulé.

Pour finir la méthode `printGrid(char[][] grid)` permet de générer une String avec en entrée un tableau. J'ai utilisé une astuce pour pouvoir inscrire des sauts de lignes dans la String car l'ajout d'un « \n » ne fonctionne pas sur mon Eclipse. J'ai donc utilisé cette ligne pour pouvoir avoir un rendu correct.

```
String Newligne=System.getProperty("line.separator");
```

Pour conclure, ce projet m'a permis de mieux comprendre comment fonctionnait les Socket ainsi que les threads. De plus, de nombreuses améliorations sont possibles comme l'implémentation d'une interface graphique par exemple.