AUDIFFRET Sam

# Projet Bataille Navale

Fonctionnement du programme :

Le serveur attend la connexion de clients et les lient par paire via un thread pour pouvoir faire plusieurs parties de bataille navale simultanément.

Le thread avertit les clients qu'ils ont trouvé une partie et attribue un chiffre au 2 joueurs (1 pour joueur 1 et 2 pour joueur 2) lors de son initialisation.

La classe Plateau comporte 7 attributs : 5 entiers indiquant l'état des bateaux (par exemple, `etatPorteAvion` vaut initialement 5, puis 4 s'il se fait toucher etc..) ; un tableau de 10 tableaux de 10 entiers correspondant à une grille et un entier correspondant à l'état de la flotte (tous les bateaux), la partie se fini quand l'état d'une flotte vaut 0.

La grille sera remplie de la façon suivante : 0 si la case est vide (eau), 1 si la case a été touchée, ou id du bateau occupant la case (2 pour le torpilleur, 31 et 32 pour les sous-marins, 4 pour le croiseur et 5 pour le porte-avion).

La classe Plateau a également 5 méthodes et 2 constructeurs :

- La méthode `placementBateau` permet de placer un bateau en indiquant la coordonnée de placement (coordonnée de la première case suivie de la direction, exemple : F3D pour case F3 direction Droite), la taille du bateau, et l'id du bateau. Cette méthode convertit les 2 premiers charactères des coordonnées en entiers pour repérer la position dans la grille puis place le bateau case par case en vérifiant à chaque fois si la case est vide. La méthode renvoi false et arrête de placer le bateau si elle rencontre une case occupée.
- La méthode `placement` permet de vérifier la validité des coordonnées et de positionner un bateau en appelant la méthode `placementBateau` si les coordonnées sont valides. Cette méthode vérifie dans l'ordre : que les coordonnées sont bien 3 charactères, que les coordonnées ne sont pas en dehors de la grille, que la direction rentrée et valide et enfin qu'il n'y a pas de collision avec un bateau déjà placé. Pour cette dernière vérification, on appel la méthode `placementBateau` en indiquant 0 comme id : on « place de l'eau », cela évite d'avoir à remettre les cases à 0 si on s'aperçoit de la collision lors du placement du bateau. Cette méthode renvoie un String indiquant si le bateau a été placé ou non.
- La méthode `attaqueCase` permet d'attaquer une case du plateau en indiquant des coordonnées. Cette méthode convertit les 2 charactères des coordonnées en entiers pour repérer la position dans la grille, stock l'id de la case attaquée dans une variable `etatCase` puis met la case de la grille à 1 (touchée) et retire 1 à `etatFlotte`. Grâce à `etatCase`, on détermine s'il y a un bateau présent et on l'identifie, cela permet de savoir si le bateau est coulé ou juste touché, en regardant l'attribut d'état correspondant. Cette méthode renvoie « plouf » si

aucun bateau n'est touché, « touché » si un bateau est juste touché, « coulé » si un bateau est coulé, « victoire » si `etatFlotte` vaut 0 après l'attaque.

- La méthode `attaque` permet de vérifier la validité des coordonnées et d'attaquer une case en appelant la méthode `attaqueCase` si les coordonnées sont valides. Cette méthode vérifie dans l'ordre : que les coordonnées sont bien 2 charactères, que les coordonnées ne sont pas en dehors de la grille et que la case attaquée n'a pas déjà été attaquée. Cette méthode renvoie le résultat de `attaque` si les coordonnées sont valides, elle renvoie un String indiquant une erreur sinon.
- La méthode `afficherGrille` permet d'afficher la grille de façon customisé (x indique les cases attaqués, 1 les bateaux, ~ l'eau)
- Un constructeur permet d'initialiser le plateau en plaçant tous les bateaux directement, il permet d'effectuer des tests plus rapidement. L'autre constructeur initialise juste le plateau.

Déroulement du thread :

Le thread initialise un plateau pour les 2 joueurs. Les joueurs placent ensuite leur 5 bateaux tours par tour (d'abord joueur 1, puis joueur 2 pour chaque bateau). Si un bateau n'a pas pu etre placé, le serveur avertit le joueur et le joueur doit rentrer des coordonnées valides jusqu'à ce que le placement soit fait.

Ensuite la partie comment, chaque joueur attaque une case du plateau adverse tour par tour, le serveur indique à chaque tour qui des 2 joueurs doit jouer aux 2 joueurs puis attend la réponse du joueur. Si le joueur indique des coordonnées invalides, le serveur l'avertit et doit rentrer des coordonnées jusqu'à ce qu'elles soient valides. Une fois la case attaquée, le serveur envoi le résultat de l'attaque au joueur (touché, coulé, plouf...) et les coordonnées attaquées à l'adversaire. Si le résultat est « victoire », le serveur envoie « défaite » à l'adversaire puis met fin à la partie, sinon, il envoie le résultat de l'attaque à l'adversaire et l'affichage de sa grille, puis passe la main au joueur adverse.

Lorsque la partie est terminée, le serveur demande au 2 joueurs s'il veulent faire une revanche. Si les 2 réponses sont « Y », on repart au début du thread, sinon le serveur met fin au thread.

Côté client :

Le client se connecte au serveur et attend de trouver une partie, il récupère ensuite l'id attribué par le serveur dans `idJoueur`. Le client place ensuite ses bateaux comme indiqué par le serveur et créé une grille d'attaque pour stocker les cases attaquées par le joueur. Pendant la partie, quand c'est le tour du joueur, le client affiche la grille d'attaque et attend une réponse du joueur. Le client envoi ensuite les coordonnées au serveur et attend la réponse du serveur. Si l'attaque a fonctionné et que la réponse du serveur n'est pas « victoire », le client met à jour la grille d'attaque pour le prochain tour, si la réponse est « victoire », le client met fin à la partie. Lors du tour de l'adversaire, le client attend les informations du serveur pour afficher quelle case a été attaquée, le résultat de l'attaque et le plateau du joueur, si le résultat de l'attaque est « défaite », le client met fin à la partie.

Lorsque la partie est terminé, le client envoie la réponse du joueur au serveur, si la réponse du joueur est « N », le programme se termine.

Améliorations possibles :

On pourrait améliorer ce programme en faisant en sorte que les joueurs puissent placer leurs bateaux simultanément, car pour l'instant les joueurs placent leur bateaux tour par tour. Cela serait possible en lançant 2 threads, un pour chaque joueur, pour placer les bateaux, je n'ai cependant pas eu le temps d'expérimenter le multithreading.

De même, on pourrait faire en sorte que les clients envoient leur réponse simultanément concernant la fin de partie, pour savoir s'ils veulent faire une revanche ou non, car pour l'instant le serveur traite la réponse du joueur 1 puis celle du joueur 2, laissant ainsi à l'abandon le joueur 2 si le joueur 1 ne veut pas faire de revanche.