HEILMANN
Brice

Compte rendu Bataille Navale :

Le but de ce TP était de créer une bataille navale avec deux clients et un serveur. Le serveur devait gérer les parties entre les deux joueurs.

Fonctionnement de mon programme :

Une fois les deux clients connectés, le placement des bateaux commence. Les joueurs doivent envoyer chacun leur tour les coordonnées de la première case du bateau et son orientation 0 ou 1 (0 pour une orientation horizontale, 1 pour verticale). L'entrée va se faire sous la forme d'un seul int, composé de 3 chiffres, par exemple si je souhaite placer un bateau à partir de la ligne 1 colonne 2 horizontalement, j'entrerai 120. Une fois la phase de placement terminée, c'est au tour de la phase de « combat ». Chacun leur tour, les joueurs entrerons les coordonnées de la case sur laquelle il souhaite tirer. L'entrée est de la forme d'un int, composé de deux chiffres, le premier pour la ligne et le deuxième pour la colonne. Par exemple si je souhaite tirer sur la case à la ligne 6 colonne 3, j'entrerai 63. Le programme s'assure que les clients ne rentrent pas n'importe quoi, et renvoie que l'entrée est invalide si tel est le cas. Après avoir envoyé les coordonnées de la case visé, le joueur se voit afficher dans sa console s'il a touché ou non un bateau, si il en a coulé un et lequel, ainsi que sa grille de jeu (où il peut voir les cases sur lesquelles il a tiré, si c'était à l'eau ou non) et sa propre grille avec ses navires et les cases sur lesquelles son adversaire a tiré. A la fin de la partie, il est proposé aux clients une revanche pour recommencer s'ils le souhaitent ou non. Le client peut également quitter la partie en entrant à son tour « quit ».

Mon programme est composé de deux packages, un package client et un package serveur. Le package client contient deux classes, la classe MainClient et le ListeningThread. Le package serveur contient quant à lui quatre classes, les classes Bateaux et Grille, ainsi que le ThreadChat et enfin le MainServer.

Côté serveur, la classe MainServer instancie les différents sockets et lance le ThreadChat (celui-ci ne se lance que si deux clients sont présents).

La classe Grille gère les grilles du jeu. Elle contient les méthodes permettant d'actualiser les grilles, de placer les bateaux, et d'afficher les grilles. Elle possède également un int qui est égale à la somme des pointes de vie de tous les bateaux. La méthode placer_un_bateau prend en argument la taille du bateau, les coordonnées de départ du bateau, son orientation, ainsi qu'une string donnant la nature du sous-marin à placer (si c'est le 1, le 2), elle est null quand ce n'est pas un sous-marin. Avec ces informations, elle place le bateau sur la grille. La grille de base n'est constituée que de 0, donc suivant la nature du bateau (donnée par sa taille et si nécessaire pour le cas du sous-marin par la string) elle remplace le 0 par le bon int, par exemple 12 pour le torpilleur. La méthode vérifier_placement_bateau

quant à elle retourne un booléen suivant si le placement du bateau est correct, c'est-à-dire si une ou plusieurs cases du bateau ne sont pas en dehors de la grille et si elles ne sont pas sur des cases d'un autre bateau. La méthode action prend en argument le message du client (donc les coordonnées de la case sur laquelle il souhaite tirer) et le PrintWriter du client en question pour lui envoyer les informations de résultats de son tir. Suivant le numéro que porte la case, un bateau sera touché (si toutes se cases ont été touchés, il est coulé et si tel est le cas il est retiré de la liste des bateaux en vie) et sinon le tir à raté et est partie dans l'eau. Les deux dernières méthodes, afficher_grille et afficher_grille_joueur affiche chez le client respectivement la grille de jeu (là où il tire,…) et sa propre grille avec ses bateaux visibles et les endroits où son adversaire a tiré.

La classe bateaux est la classe de l'objet bateau. Chaque bateau a une taille et une vie. Cette classe possède les méthodes get et set pour chaque caractéristique d'un bateau, une méthode impact qui décrémente de 1 la vie d'un bateau à chaque fois qu'il est touché et une méthode est_coule qui vérifie si un bateau est coulé ou non.

La classe ThreadChat est le thread côté serveur. Il gère la partie, elle définit tout d'abord dans son constructeur, un BufferedReader et un PrintWriter pour chaque client. Puis dans la méthode run, elle gère la partie. Elle commence par le placement des bateaux chez les 2 joueurs (où elle vérifie si un bateau est correctement placé), puis viens la bataille. A tour de rôle chaque joueur envoie les coordonnées d'une case, elle appelle alors les méthodes de la classe grille et envoie les résultats et les grilles actualisés aux clients. Quand un client n'a plus de bateaux en vie, elle donne les résultats de la partie, puis donne la possibilité de refaire une partie, sinon après envoie d'un message, elle éteint le serveur.

Côté client, la classe MainClient gère le client. Elle se connecte au serveur, et envoie les messages aux serveurs en s'assurant que ceux-ci sont correctes, c'est-à-dire de bonne taille, 3 en phase de placement et 2 en phase de jeu, et si c'est bien un int ou quit pour quitter la partie (dans ce cas le client s'éteint).

La classe ListeningThread est le thread d'écoute du client, il lui permet de récupérer les informations du serveur et de les afficher au client. Il garde également en mémoire la grille de jeu du client (celle où il tire,…). Les autres méthodes de cette classe envoient des informations au MainClient sur le déroulement de la partie (si la phase de placement est terminée, …).