SAE 3.03 *Puissance 4 en réseau*

1 Sujet

Vous devez développer une application client-serveur pour jouer à Puissance 4.

Il y a plusieurs aspects à cette application :

- le client qui doit se connecter au serveur pour afficher les joueurs en ligne, démarrer une partie, afficher le jeu, voir les scores et saisir les entrées du joueur
- le serveur qui gère les différentes parties et les joueurs
- le protocole de communication entre le client et le serveur

Le jeu se déroule dans un terminal, en entrant les commandes sous forme de texte.

1.1 Client

Le client se connecte au serveur en précisant l'adresse IP de celui-ci (ou son nom), ainsi que son nom de joueur (par exemple `java Client 192.168.1.10 nomJoueur` . Il voit ensuite la liste des joueurs en attente d'une nouvelle partie. Chaque joueur est représenté par son nom, son nombre et son pourcentage de victoires.
Le client sélectionne un adversaire, et si celui-ci accepte la demande, la partie commence.

1.1.1 Déroulé d'une partie

Le jeu se déroule sur une grille de 6 lignes et 7 colonnes. Chaque joueur dispose d'une couleur de jeton (vous choisirez un caractère différent pour représenter chaque joueur dans le terminal). A tour de rôle les joueurs placent leur pion dans une des colonnes. Le vainqueur est le premier joueur qui réussit à aligner 4 jetons de sa couleur (horizontalement, verticalement ou en diagonale).
Si la grille est remplie, sans qu'aucun des deux joueurs n'ait réussi à aligner quatre jetons, il y a égalité.
Un joueur peut quitter une partie en cours, dans ce cas, il sera déclaré perdant.
Une fois la partie terminée, les deux joueurs retournent dans la liste des joueurs en attente. Le client peut quitter le serveur.
Lorsqu'un joueur est en cours de partie, il n'est plus visible dans la liste des joueurs en attente.

1.1.2 Historique des parties

Le joueur peut également avoir accès à la liste de ses précédents matchs avec le nom de ses adversaires, le résultat du match et la date. Il voit son nombre total de matchs, de victoires de défaites et de nuls.
Pendant que le joueur est dans ce menu, il peut recevoir des demandes de nouvelle partie

1.2 Serveur

Le serveur doit accepter des connexions de la part des clients, gérer les joueurs en attente. Il doit également gérer une partie, récupérer les entrées des joueurs, détecter la victoire d'un joueur ou la fin d'une partie. La partie est entièrement gérée par le serveur, le serveur devra envoyer au client l'état du plateau du jeu après chaque coup. Plusieurs parties peuvent se jouer en même temps.

Le serveur doit conserver les statistiques de chaque joueur.

Le serveur devra toujours être à l'écoute du client, même si celui-ci n'est pas censé communiquer. Par exemple si c'est au tour du joueur 1, et que le joueur 2 envoie un message, il devra avoir une réponse du serveur. Veillez également à la robustesse, le serveur ne doit pas s'arrêter si le client envoie une commande non prévue.

1.3 Protocole

Votre programme client-serveur doit suivre un protocole d'échanges de messages sous forme de texte. Voici quelques exemples de requêtes du client et de réponse du serveur.

- `connect` *nomJoueur* : demande de connexion du client avec son nom de joueur. Le *nomJoueur* doit être une chaîne de caractères sans espace, avec au minimum 3 caractères, et 10 caractères au maximum. Le serveur répond `OK` si le joueur peut se connecter, `ERR` *message* si le nom du joueur n'est pas valide (le joueur est déjà présent par exemple, ou si le nom n'est pas précisé
- `ask` *nomJoueurAdverse* : demande de nouvelle partie contre un joueur. Le serveur répond `OK` si la demande est valide, `ERR` *message* si le nom du joueur adverse n'est pas valide (pas présent dans les joueurs en attente par exemple)
- `play` *nomColonne* : pendant une partie active, le joueur dépose un jeton dans la colonne mentionnée. *nomColonne* doit être un entier entre 1 et 7. Le serveur réponse `OK` si le coup est valide, `ERR` *message* si le coup n'est pas valide (la colonne est remplie, ou ce n'est pas le tour du joueur,...)
- ...

A vous de compléter cette spécification du protocole.

1.4 Extensions

1.4.1 Persistance

Faites en sorte que le serveur soit persistant. Si celui-ci redémarre, il doit conserver les scores de chaque joueur.
De la même manière, vous pouvez faire un client qui se souvient du serveur, et du nom de l'utilisateur.

1.4.2 Interface graphique

Développer une interface graphique pour le client

2 Déroulement de la SAÉ

La SAÉ se fait par groupe de quatre. Elle se déroule du 02/12 au 20/01. Le développement se fait en Java.

- Rendu intermédiaire le 20/12/2024 :
  - Diagramme de classes
  - Le protocole de communication entre le client et le serveur
  - Un prototype de l'application

- Rendu final Le **18/01/2025** sur Célène :
  - Application fonctionnelle
  - Documentation (javadoc)
  - Rapport : Celui-ci doit contenir
    - Manuel d'utilisation
    - Justification des choix techniques
    - Diagramme des classes final
    - Protocole complet détaillé
- Une soutenance de 15 minutes pour présenter votre travail. La date vous sera communiquée ultérieurement

Résumé	Développer une application communiquant sur le réseau via des sockets, et utilisant des threads
Compétences Visées	• Compétence 3 : Administrer des systèmes informatiques communicants complexes • Compétence 6 : Travailler dans une équipe informatique
Heures SAÉ	6 heures entre le 02/12 et le 18/01
Liste des ressources mobilisées	• R3.05 : Programmation Système • R3.06 : Architecture des réseaux
Livrables sur Célène	• Code source commenté • Javadoc • Rapport