CONCEPTION D'UNE APPLICATION BASEE SUR L'ARCHITECTURE CLIENT-SERVEUR

8INF957 - Programmation objet avancée - Automne 2022

Présenté par : Benjamin NGABMEN NJAWAT – NGAB27020100

DESCRIPTION DE L'OBJECTIF À ATTEINDRE

Application Jeu d'échecs développée par l'UQAC

Mode multijoueur

DESCRIPTION DE L'OBJECTIF À ATTEINDRE

L'utilisateur sera en capable de :

- Créer une nouvelle partie multijoueur et partager le numéro de son salon de jeu
- Rejoindre une partie à l'aide d'un numéro de salon
- Affronter un adversaire en ligne en jouant à tour de rôle

Pas de modification du code existant !

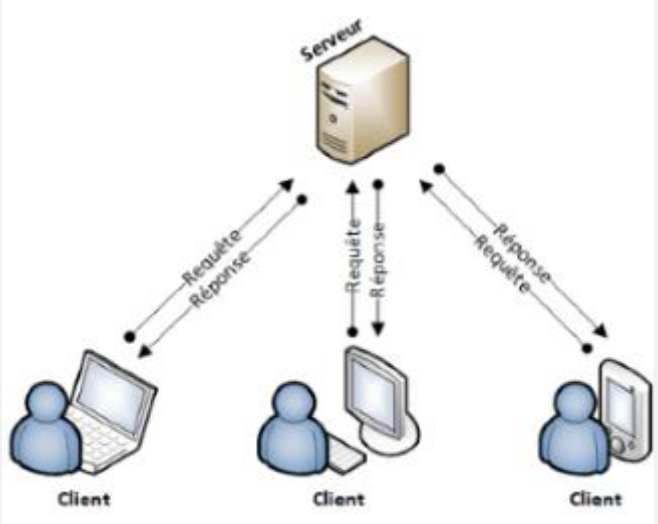
QU'EST-CE QUE L'ARCHITECTURE CLIENT-SERVEUR ?

Définition

L'architecture client/serveur désigne un mode de communication entre plusieurs ordinateurs d'un réseau qui distingue un ou plusieurs postes clients du serveur : chaque logiciel client peut envoyer des requêtes à un serveur.

Protocole

Le client et le serveur doivent bien sûr utiliser le même protocole de communication



Serveur

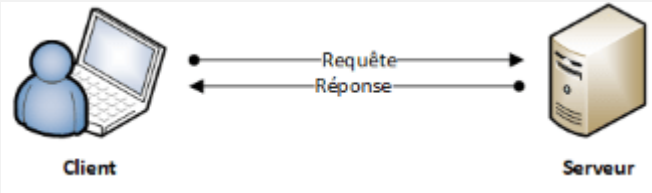
- Il est passif (ou maître);
- Il est à l'écoute, prêt à répondre aux requêtes envoyées par des clients
- Dès qu'une requête lui parvient, il la traite et envoie une réponse.

Client

- Il est actif (ou esclave) ;
- Il envoie des requêtes au serveur ;
- Il attend et reçoit les réponses du serveur.

TYPES D'ARCHITECTURES CLIENT-SERVEUR

Architecture 2 tiers

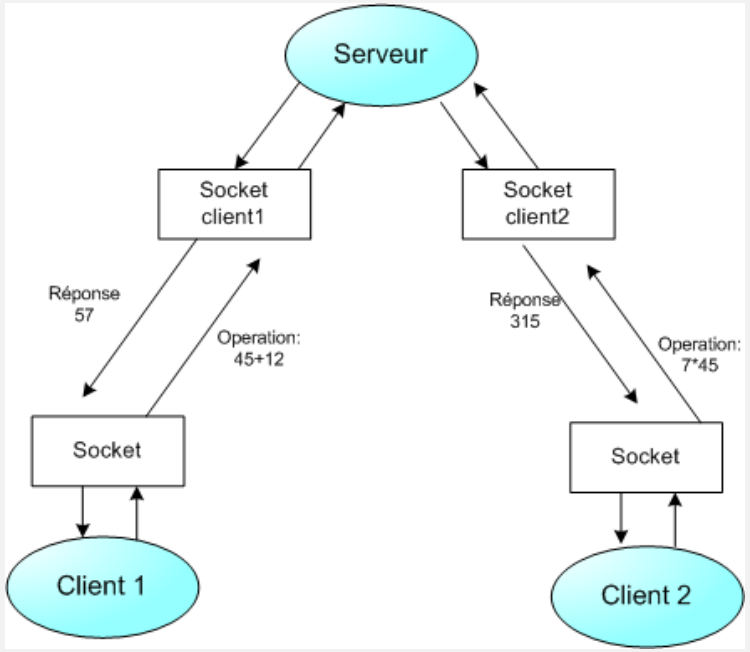


Architecture 3 tiers

Architecture n tiers

C'EST QUOI LES SOCKETS ?

- Les sockets servent à communiquer entre deux hôtes Client - Serveur à l'aide d'une adresse IP et d'un port,
- Ces sockets permettront de gérer des flux entrants et sortants afin d'assurer une communication entre les deux (le client et le serveur), soit de manière fiable à l'aide du protocole TCP/IP, soit non fiable mais plus rapide avec le protocole UDP.



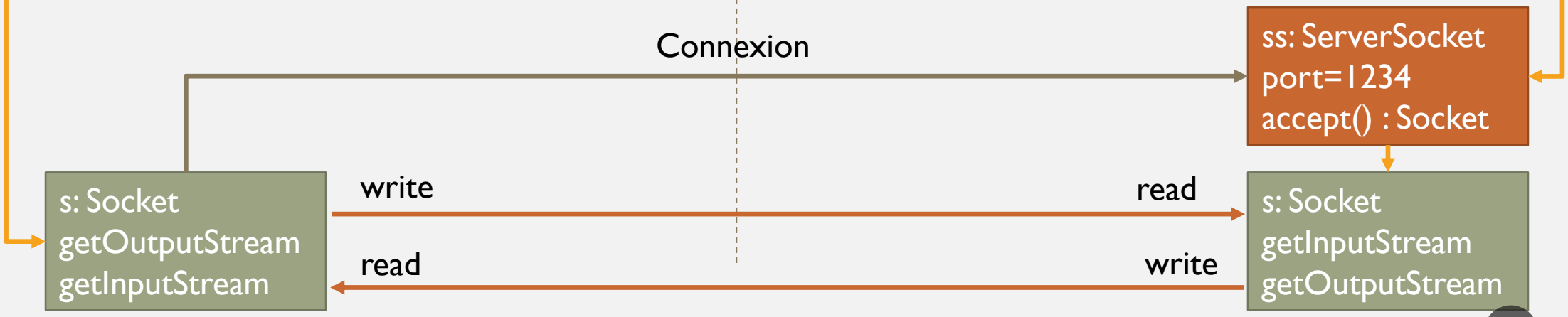
PRINCIPE DE FONCTIONNEMENT

Client

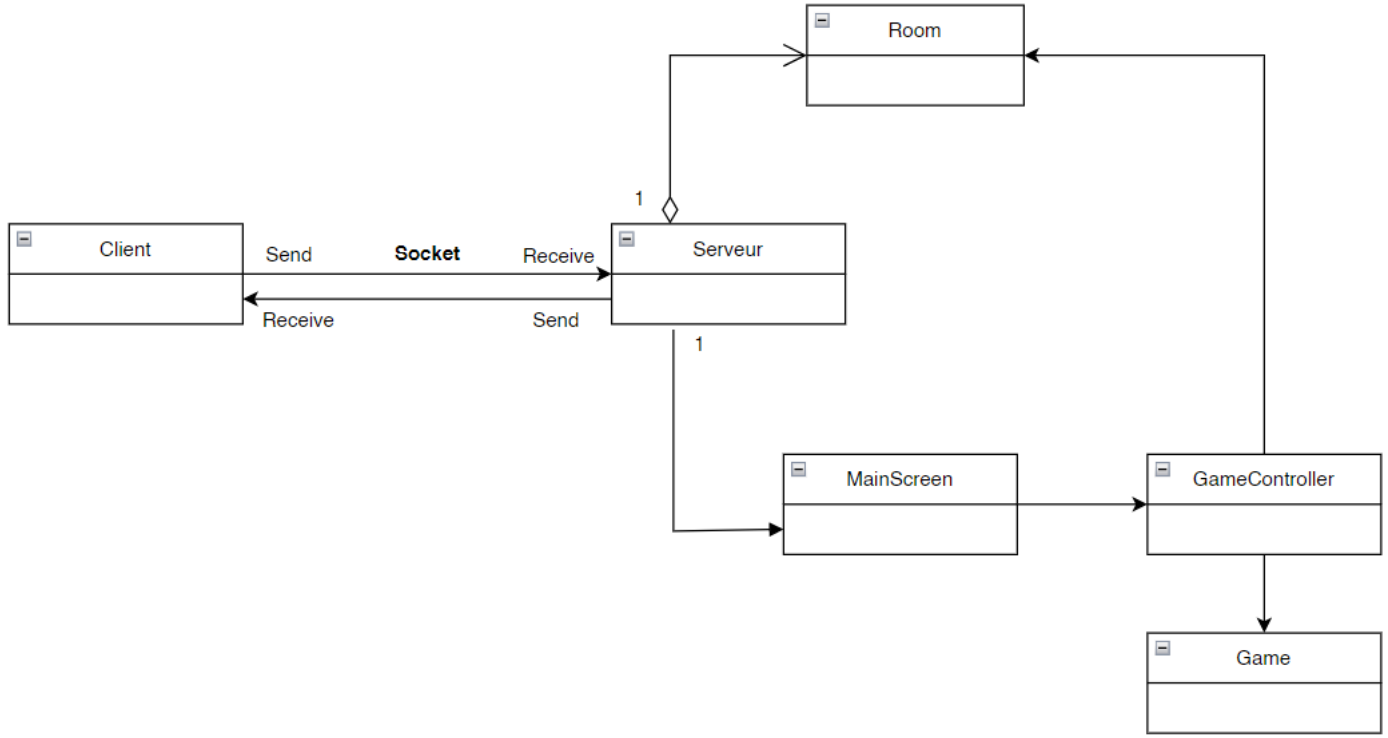
```
Socket s = new Socket(« localhost », 1234);

InputStream is = s.getInputStream();
OutputStream os = s.getOutputStream();
os.write(23);
Int rep = is.read();
System.out.println(rep);
```

Serveur

```
ServerSocket ss = new ServerSocket (1234);

Socket s = ss.accept();

InputStream is = s.getInputStream();
OutputStream os = s.getOutputStream();
Int nb = is.read();
Int rep = nb*2;
os.write(rep);
```



MODÈLE DE NOTRE SOLUTION



DÉMONSTRATION

DIFFICULTÉS RENCONTREES

- Extension des fonctionnalités sans modification du code existant
  - Solution : Utilisation de l'héritage et création des aspects

- Compréhension de la programmation multithreading
  - Solution : Beaucoup de documentation sur le sujet

AMELIORATIONS POSSIBLES

Intégrer une interface graphique :	Réduire le couplage entre les classes :	Gestion des exceptions et de la qualité réseau	Utiliser une architecture client-serveur n tiers pour plus de performances
• Implémenter une architecture MVC	• Adopter des patrons de conception adéquats		

MERCI POUR VOTRE ATTENTION