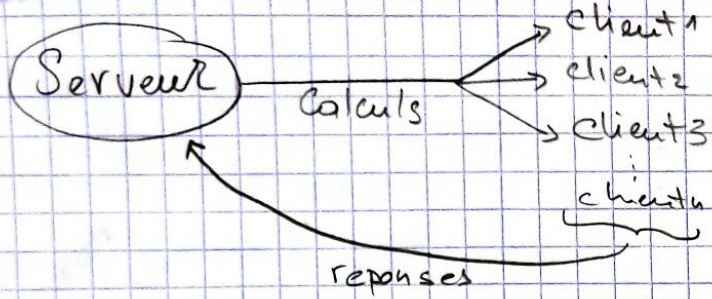
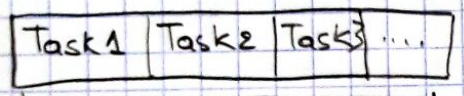


Calcul = Job
tâche = Task



Job →
- chaque Job est identifié par un ID → jobId : long
- un job est composé de nbTasks tâches qui sont numérotés de 0 à nbTask-1

. un fichier compte rendu est placé dans le Github afin de noter en détail ce que chacun a réalisé avec date de réalisation.

faut pas oublier qu'il y aura une soutenance autant noter dès maintenant

Workflow

1) Un client se connecte au serveur
2) Le client se voit attribuer une tâche à la connexion
3) Le client execute la tâche
4) Le client envoi au serveur son résultat
5) On recommence

★ Le serveur retourne :

- l'URL d'un JAR et le nom d'une de ses classes qui implemente l'interface Worker, JobId ... etc

↳ Ces deux parametres font reference à un job réalisé par la classe en question

- Exemple : regarder l'exemple des nombres premier cité au début du sujet de projet

---

1). le client fait une requête GET afin de recuperer une tâche du serveur

2). la reponse du serveur est au format HTTP et contient un JSON encodé en UTF-8 qui représente les informations d'un Job

. Si aucun Job n'est disponible le serveur demande au client de se reconnecter dans 800 seconde

3). le client crée une instance de la classe donnée à partir du JAR recuperé et l'utilise afin d'effectuer son calcul

4). le client répond par une requête POST

* Le protocole limite la taille des requêtes à 4096, si elle le dépasse le client doit le specifier

* Verifier les cas d'erreurs cités dans le sujet du projet à la fin de la partie communication

Le client

* Prend en ligne de commande le clientID, l'@ du serveur et le port
* Utilisation de la class WorkerFactory afin de créer une instance implementant l'interface Worker
* Ne pas creer plusieurs instances d'un Worker si ce n'est pas necessaire
* Verifier que la sortie du Worker est valide
* Respecter le temps demandé par le serveur
* LE CLIENT NE DOIT JAMAIS S'ARRETER

- On peut se baser sur l'exemple du sujet pour realiser le client.
- On a déjà le format des requêtes/reponses.
- On enregistre dans des variables global statiques les differentes reponses possibles du serveur et on les utilise dans les cas appropriés pour effectuer nos tests

1) Mise en place du code basique qui permet une connexion à un serveur

2) Effectué une requête GET à la connexion
   - Verifier la reponse et sa validité
   - Stocker la reponse car on en aura besoin pour le POST
   - Parser la reponse pour utiliser ses informations (librairie Jackson)

   Si le serveur repond qu'il n'a pas de Job refaire le GET 300ms après

3) Recuperer le JAR
   - Creer une instance du Worker
   - lancer le calcul
   - Verifier sa reponse
   - parser la reponse et la stocker

4) La reponse du Worker et la toute première reponse du serveur sont utilisés pour envoyer le POST au serveur

* Parser les JSON en MAP → regarder la lib Jackson déjà utilisé dans le projet JAVA