ACS APP

Orveau Rémi Hubert Nathan Lescorbie Valentin Rucher Thibault Lenglet Jolan

Sommaire

1. [Répartition des rôles 2](#_bookmark0)
2. [Diagramme UML 2](#_bookmark1)
3. [Diagramme de séquence 3](#_bookmark2)

# Répartition des rôles et difficulté

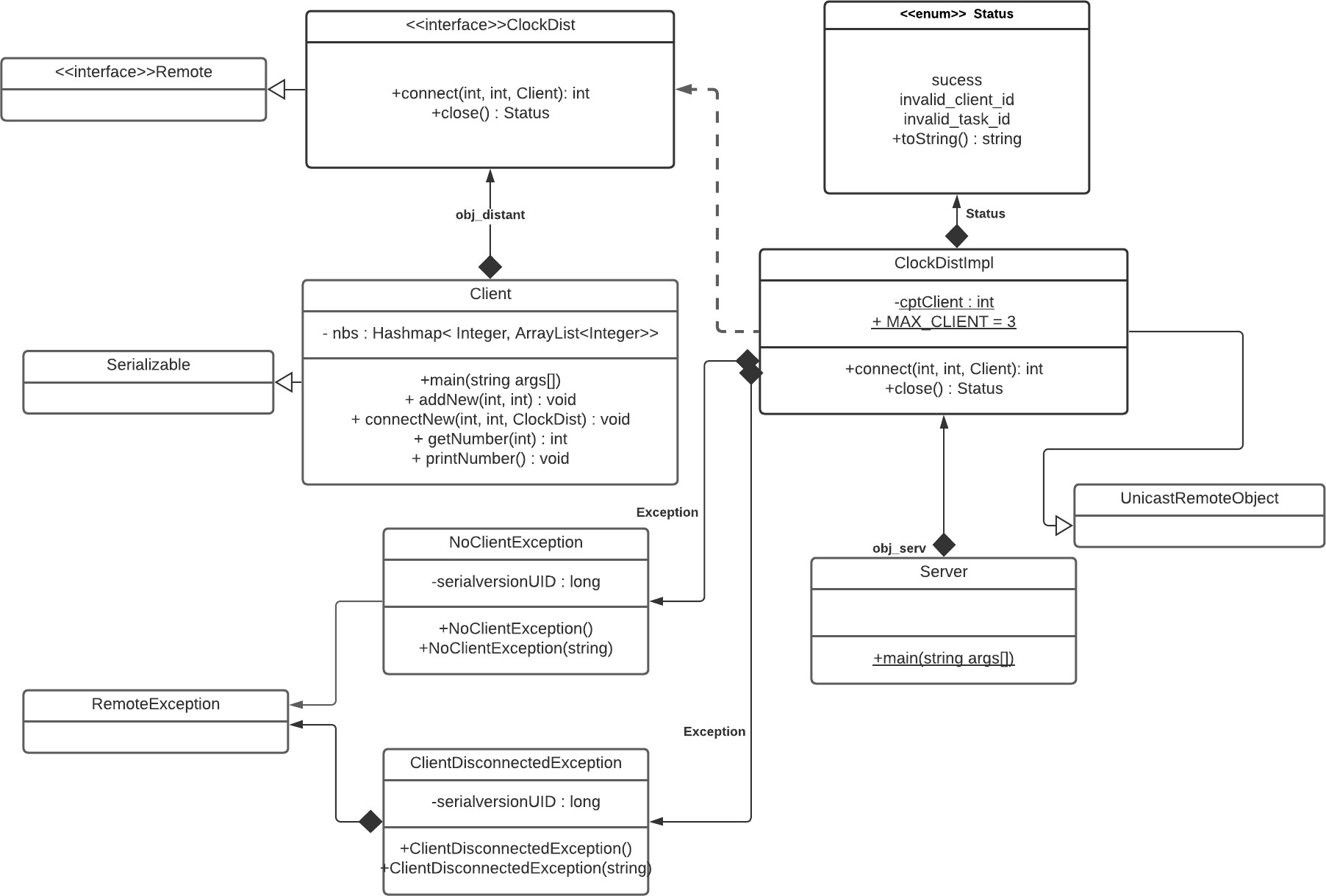
Gestion du temps : Valentin Animateur : Thibault

Scribe : Rémi

L’ensemble du groupe à commencer par échanger sur le projet, nous avons mis en commun nos idées puis avons tous ensemble travaillé sur le diagramme UML. L’échange des idées et le travail réparti dans un groupe de 5, nous a permis de travailler rapidement.

Nous avons décidé que ça serait le serveur qui envoie des nombres régulièrement au client, de plus côté client c’est un thread qui gère l’ajout dans son ArrayList. Une grosse difficulté rencontrée a été que nous devions envoyer une référence du client au serveur pour qu’il puisse ajouter ces données. Malheureusement le client envoyé ne modifie pas.

# Diagramme UML



# Diagramme de séquence

# Validations

# Tâche principale du client effectivement exécutée en parallèle de la requête au serveur

# -> MultiClientTest5

# make test5

# Assurance que le client reçoit bien n valeurs aléatoires toutes les x secondes

# -> MutlriClientTest2

# Make test2

# Client pouvant lancer plusieurs requêtes au même serveur

# -> MultiClientTest1 & MultiClientTest4

# make test1

# make test4

# Limitation du nombre d’échanges

# -> MutltiClientTest3

# make test3

# Gestion de la charge du serveur (le cas échéant)

# -> MutltiClientTest3 make test3

# Synchronisation souple entre le client et le serveur (client bloqué, à sa demande, si valeur non dispo

# -> MultiClientTest6

# make test6