Université du Québec à Chicoutimi

Département d'informatique et de mathématique

8INF415 – Systèmes distribués : TP1

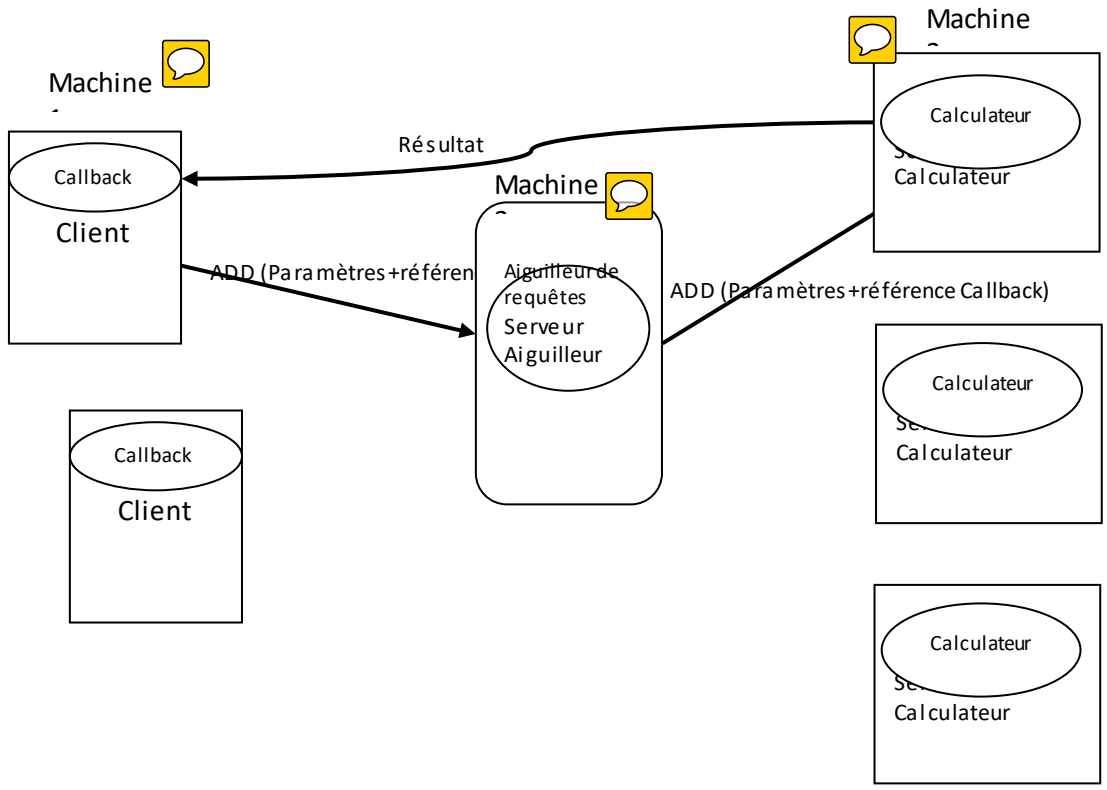
Professeur : Hamid Mcheick	Groupe : individuel
Semestre : A2018	Date de distribution : 3 septembre 2018
Pondération : 15 points	Date de remise : 26 septembre 2018

Le but du TP est de familiariser l'étudiant

1. au développement d'applications distribuées en utilisant la communication par message (Socket), par procédure, par objet (RMI), Java EE, etc.
2. aux concepts de passage de paramètres, de sérialisation, de nommage, de persistance, de la distribution de charge, de collaboration dans les systèmes distribués, etc.

Vous devez choisir UNE seule question parmi les deux suivantes.

Choix 1 : RMI - CallBack



Le but de cette question est de permettre aux plusieurs clients distribués d'utiliser les fonctionnalités des objets distribués Calculateurs.

Pour distribuer la charge de calcul d'une manière efficace et performante, un client ne peut pas contacter directement un objet Calculateur, il est obligé de contacter un objet distant «Aiguilleur de requêtes » qui, à son tour, choisit un calculateur pour lui faire suivre l'appel.

- Chaque client doit créer un objet Callback pour permettre à un objet calculateur de lui renvoyer le résultat d'une façon asynchrone.
- L'objet calculateur permet d'effectuer les opérations de calcul de base, addition, soustraction, multiplication et division sur des entiers, il renvoie le résultat d'une façon asynchrone à l'aide d'une référence d'un objet Callback.
- L'objet «Aiguilleur de requêtes » joue le rôle d'un proxy pour le client, reçoit de la part des clients les appels vers les méthodes de calcul, et comme il n'implémente pas ces méthodes, il choisit un objet calculateur distant, et envoie l'appel vers l'objet choisit.
- D'autre part, l'objet «Aiguilleur de requêtes » permet d'enregistrer des nouveaux objets calculateur pour les faire utiliser dans la distribution de calcul.

1. Écrire l'interface Calculateur. Ainsi que son implémentation.
2. Écrire l'interface Aiguilleur. Ainsi que son implémentation.
3. Écrire la classe client qui teste des fonctions de calcul d'un objet calculateur.
4. Nommer pour chacune des machines (Machine1, Machine2, et Machine3), les noms des classes nécessaires au bon fonctionnement de l'application.

Choix 2 : FacebookServer **Socket - TCP**

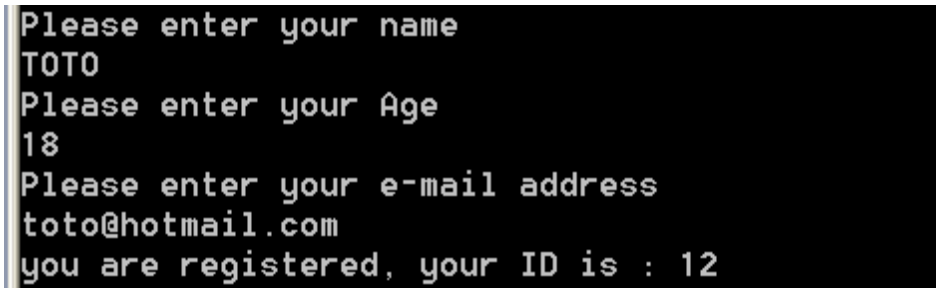
Le but de cette question est de construire une première version de Facebook. L'application utilise la communication Socket en mode TCP ou RMI. L'application est composée d'un serveur central FacebookServer, ainsi que des clients distribués.

Une fois un client est connecté au serveur, un objet Profile est créé pour ce client. Comme le serveur doit interagir avec plusieurs clients simultanément, la classe Profile doit agir comme un Thread.

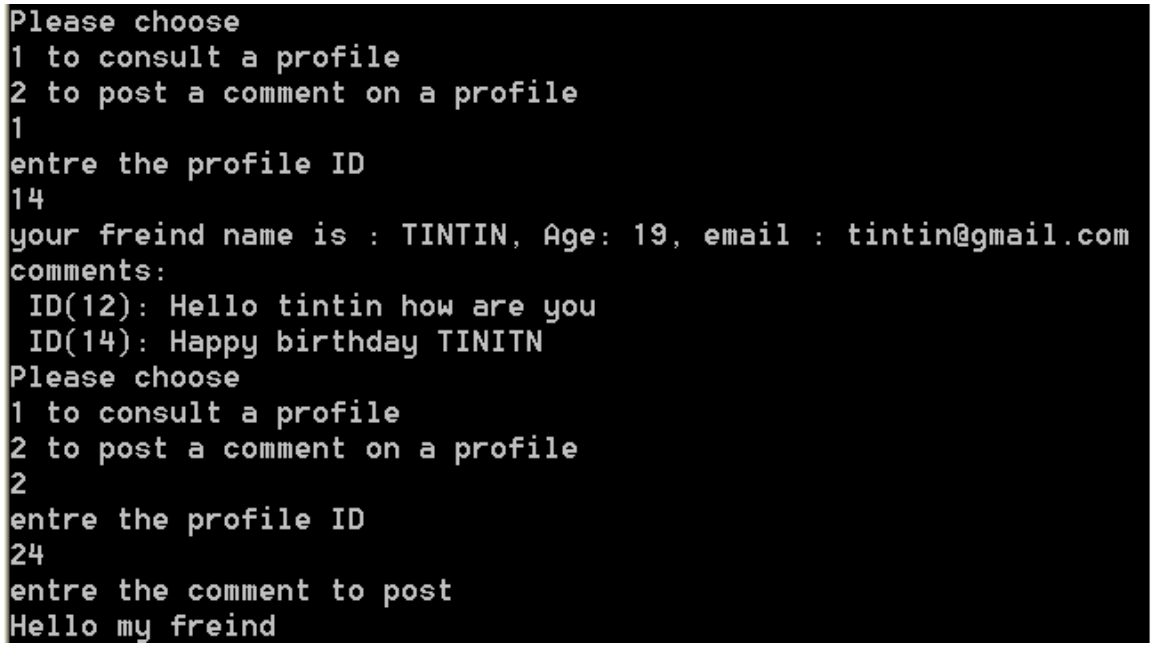
Chaque client est représenté par un objet Profile côté serveur. Chaque objet Profile possède un identificateur ID unique attribué par le serveur durant sa création. L'objet Profile doit stocker les informations relatives au client correspondant (Name, Age, et email). Il sauvegarde aussi les commentaires ajoutés sur le profile par d'autre client, ces commentaires seront stockés dans une structure de données (tableau, Vector, Hashtable ou HashMap) contenant l'ID des utilisateurs qui ont ajouté les commentaires ainsi que les commentaires ajoutés, comme le montre le tableau suivant:

ID	**Comments**
12	Welcome to Facebook
2	I am your friend from the school, Titi, did you remember me
13	Happy birthday TOTO

Pour assurer la connexion avec le serveur, le client demande à l'utilisateur d'entrer les informations concernant son Profile, dans le but d'envoyer ces informations à l'objet Profile correspondant au côté serveur.
L'écran de client doit afficher les messages suivants :



Ensuite, le client construit les informations du Profile sous forme d'un message texte sous le format suivant:
Name:TOTO;Age:18;email:toto@hotmail.com
Ce message sera envoyé sous format texte vers l'objet Profile créé au côté serveur.
Supposons qu'un utilisateur connaît les ID des autres clients connectés au serveur. Après la création de l'objet Profile, le client va aussi permettre à l'utilisateur de consulter le profile des autres utilisateurs identifiés par un ID, et d'ajouter des commentaires sur leurs profiles, comme l'indique la capture d'écran coté client :



1. Écrire le code des classes Profile et FacebookServer, la classe Profile doit permettre de répondre à la requête du client:
i. Elle doit stocker les informations concernant le client correspondant.
ii. Elle permet de consulter le profile des autres clients.
iii. Elle permet d'ajouter des commentaires au Profile des autres clients.
2. Écrire le code du client (une ou plusieurs classes) qui permet d'accomplir les exigences citées ci-dessus.

3. Écrire les modifications du code qui doivent être faites (du côté serveur seulement) pour permettre à deux clients d'établir une session de messagerie (Chat) entre eux.

Annexe :
Class String

Class Hashtable

Un exemple d'utilisation d'une table de Hashage:
Cet exemple crée une table de Hachage "Hashtable" d'objets ObjectA. Il utilise les nom « one », « two » comme clés.

```
Hashtable numbers = new Hashtable();
numbers.put("one", Objet1);
numbers.put("two", Object2);
```

où Objet1, Objet2, sont des instances de l'ObjectA
Pour trouver un ObjetA identifié dans le Hastable par une clé two, on utilise le code suivant :

```
ObjectA n = (ObjectA) numbers.get("two");
```

Livrables : Copie complète du TP1 (en classe) contenant :

1) le code complet qui fonctionne sur Eclipse

2) un scenario (trace) d'exécution et un manuel d'utilisation pour la question choisie.

Vous devez montrer l'exécution de ce programme pour évaluer ce TP.