FACULTÉ DES SCIENCES ET TECHNIQUES
DE MOHAMMEDIA
UNIVERSITÉ HASSAN II DE CASABLANCA

Module Architecture distribuée

Mini Projet:

Gestion d'un Forum de discussion

Filière d'ingénieur :

LISI
Ingénierie Logicielle et Intégration des Systèmes Informatiques

Encadrant pédagogique :

- Pr. BERGUIG Yousra

Réalisateurs:

- Hsissou Fadoua

Année universitaire : 2022-2023

Table of Contents

Table of Figures

I. Introduction :

Le présent mini projet a pour objectif la création d'un forum de discussion permettant à un nombre quelconque d'utilisateurs de communiquer entre eux en temps réel. Les utilisateurs peuvent se connecter au forum, émettre des messages qui sont diffusés à l'ensemble des utilisateurs présents, connaître la liste des intervenants connectés et se déconnecter du forum. Les messages ne sont pas mémorisés par le forum.

Le fonctionnement de l'application repose sur l'utilisation de la technologie RMI (Remote Method Invocation). Le serveur Forum fournit des méthodes appelables à distance par les utilisateurs (entrer, quitter, qui, dire) et utilise des objets proxy pour diffuser les messages. Les clients User utilisent une interface graphique pour interagir avec le forum et peuvent afficher les messages reçus dans cette interface.

L'implémentation de l'application se fait à travers plusieurs classes : Forum, ForumImpl (serveur), User, UserImpl (partie cliente), proxy, proxyImpl (objet accessible à distance). Le serveur Forum utilise l'interface Forum pour définir les méthodes appelables à distance et le client User utilise l'interface User pour définir la méthode ecrire qui permet d'afficher les messages dans l'interface graphique.
L'objectif de ce mini projet est de développer les classes ForumImpl et proxyImpl en respectant les interfaces Forum et proxy respectivement. Le rapport présentera les différentes étapes de l'implémentation, ainsi que les tests effectués pour s'assurer du bon fonctionnement de l'application.

II. Implementation :

1- Implémentation côté serveur :

1-1 Forum et ForumImpl :

la classe **ForumImpl**. Cette classe implémente l'interface **Forum** qui définit les méthodes à appeler à distance par les utilisateurs .
La classe **ForumImpl** étend **`UnicastRemoteObject`** afin de faciliter la communication distante via RMI (Remote Method Invocation). Cette classe fournit des fonctionnalités qui permettent aux objets de cette classe d'être enregistrés auprès d'un registre RMI et de communiquer avec des objets distants en utilisant des méthodes distantes .

La Class **ForumImpl** définit deux variables importantes :

- La première variable est une **liste d'objets de type "proxy".** Ces objets représentent les clients connectés au Forum et sont utilisés pour diffuser les messages émis par les utilisateurs. Cette liste permet de stocker les références de ces objets et de les parcourir pour envoyer les messages à chacun des clients connectés.

- La deuxième variable "**hostname**" est une chaîne de caractères qui contient le nom de l'hôte du serveur. Elle est utilisée pour afficher le nom de l'hôte sur lequel le serveur est en cours d'exécution

les méthodes à appeler à distance par les utilisateurs :

- **La méthode entrer :**
  Cette fonction est appelée lorsqu'un client souhaite se connecter au forum. Le paramètre 'pr' est l'objet proxy du client qui est utilisé pour communiquer avec le forum.
  La fonction ajoute l'objet proxy du client à la liste des objets connectés stockée dans la variable 'Proxies'. Ensuite, elle envoie un message de confirmation de connexion au client en utilisant la fonction 'ecouter' de l'objet proxy. Ce message contient le nom de l'hôte du serveur et l'ID du client qui vient de se connecter.
  Enfin, la fonction renvoie l'ID du client pour les futures communications. L'ID est l'indice de l'objet proxy du client dans la liste 'Proxies'.

```java
@Override
public int entrer(proxy pr) throws RemoteException, UnknownHostException {

    // Ajout du proxy à la liste des objets connectés
    Proxies.add(pr);
    // Envoi d'un message de confirmation de connexion au client
    pr.ecouter("client connecté a "+InetAddress.getLocalHost().getHostName()
            + " avec Id "+ Proxies.indexOf(pr));
    // Renvoi de l'ID du client pour les futures communications
    int id = Proxies.indexOf(pr);
    return id;

}
```

Figure 1: La méthode « entrer »

- **La méthode Dire :**

  Lorsqu'un utilisateur appelle la méthode dire, le serveur Forum parcourt la liste des Proxy et envoie le message à tous les utilisateurs, sauf celui qui a envoyé le message , elle appelle la méthode **`ecouter(msg)`** de l'objet **`proxy`**, qui permet de diffuser le message `msg` au client correspondant à cet objet **`proxy :`**

```java
@Override
public void dire(int id, String msg) throws RemoteException {

    // Diffusion du message à tous les clients sauf l'émetteur
    for (int i = 0; i < Proxies.size(); i++) {

        proxy p = Proxies.get(i);
        // Vérification que le proxy n'est pas l'émetteur
        if (i != id) {

            p.ecouter(msg);
        }
    }
}
```

Figure 2: La méthode « dire»

- **La méthode Qui :**

  La méthode `qui()` permet de récupérer la liste des clients connectés au forum. Elle renvoie une chaîne de caractères qui contient les informations de chaque client connecté, telles que lHostname du client, ainsi que l'identifiant attribué par le forum.

```java
@Override
public String qui() throws RemoteException, UnknownHostException {

    StringBuilder sbuffer = new StringBuilder();
    // Construction de la liste des clients connectés
    for (proxy p : this.Proxies) {

        sbuffer.append(p.toStringp()).append("\n");
    }
    return sbuffer.toString();
}
```

Figure 3: La méthode « qui() »

- **La méthode quiter** :

  Lorsque l'utilisateur appelle la méthode quiter, l'objet Proxy correspondant est supprimé de la liste et un message est envoyé à tous les utilisateurs pour les informer du départ de l'utilisateur.

```java
@Override
public void quiter(int id) throws RemoteException {

     // Suppression du client de la liste des objets connectés
    this.Proxies.remove(id);
}
```

Figure 4: a méthode « quiter »

1-2 Le Server RMI :

La méthode main() commence par créer un registre RMI en utilisant la méthode static createRegistry() de la classe LocateRegistry, qui prend un numéro de port en paramètre (dans cet exemple, 1099).

Ensuite, la méthode crée une instance de la classe ForumImpl, qui implémente l'interface Forum et fournit les méthodes nécessaires pour le chat à distance.

Enfin, la méthode utilise la méthode static rebind() de la classe Naming pour lier l'objet forum à l'URL **"rmi://localhost:1099/IRCServer".** Cela signifie que les clients qui se connectent à cette URL pourront invoquer les méthodes de l'objet ForumImpl à distance.

```
    LocateRegistry.createRegistry(1099);
    ForumImpl forum = new ForumImpl();
    Naming.rebind("rmi://localhost:1099/IRCServer", forum);
```

Figure 5: Le Server RMI

2- Implémentation d' Object Proxy :

L'interface **proxy** est une interface distante qui étend l'interface Remote de RMI. Elle définit deux méthodes

La classe **proxyImpl** implémente l'interface **proxy** et hérite de la classe **UnicastRemoteObject** de RMI , Elle implémente également l'interface **Serializable**, ce qui permet à l'objet d'être sérialisé et désérialisé .

Cette classe est utilisée pour créer une instance d'un objet distant qui peut être utilisé par le serveur pour communiquer avec un client distant , Elle contient un **attribut User** qui représente l'utilisateur associé à ce proxy et Elle définit deux méthodes :

- **ecouter(String msg)** : cette méthode est appelée par le serveur pour envoyer un message à un client distant. Elle affiche le message dans la console et appelle la méthode ecrire(String msg) de l'objet User associé à cet objet proxy.
- **toStringp() :** cette méthode Elle retourne une chaîne qui contient l'ID de l'utilisateur associé à cet objet et le nom de l'hôte du client.

```
public class proxyImpl  extends UnicastRemoteObject implements Serializable,proxy{

    private User user;

    public proxyImpl(UserImpl irc) throws RemoteException {

        super();
        this.user = irc;
    }

    @Override
    public void ecouter(String msg) throws RemoteException {

            System.out.println("Received message: " + msg);
            this.user.ecrire(msg);
    }

    @Override
    public String toStringp() throws RemoteException, UnknownHostException {

         return "User ID : " +user.getId() + " User Hostname : " +InetAddress.getLocalHost().getHostName();
    }


}
```

Figure 6: La Class ProxyImpl

3- Implémentation côté Client :

une classe **UserImpl** qui implémente l'interface **User** qui contient la méthode **getId**(). Cette méthode permet de renvoyer l'identifiant du client connecté et la méthode **ecrire** qui permet d'afficher une chaîne de caractères (msg) dans la fenêtre de chat de l'utilisateur.

Ensuite, la classe UserImpl crée une interface utilisateur (GUI) à l'aide de la bibliothèque Swing. Cette interface est composée de quatre boutons : **connecter, écrire, qui et quitter.**

Le bouton "**connecter**" se connecte au serveur RMI en utilisant la méthode Naming.lookup() et récupère une référence à l'objet distant Forum. Ensuite, la classe proxyImpl est utilisée pour créer un objet proxy qui est ensuite enregistré auprès du forum en appelant la méthode entrer(). Cette méthode renvoie un identifiant unique attribué par le serveur.

Le bouton "**écrire**" permet d'envoyer un message brodcaste sur le forum . Lorsque le bouton est cliqué, la méthode dire() est appelée sur l'objet distant Forum avec l'identifiant du client et le message à envoyer.

Le bouton "**qui**" renvoie la liste des utilisateurs connectés sur le forum en appelant la méthode qui() sur l'objet distant Forum.

Enfin, le bouton "**quitter**" permet de quitter le forum en appelant la méthode quitter() sur l'objet distant Forum.

Ces boutons permettent de réaliser les opérations de base d'un chat en utilisant la technologie RMI pour la communication entre les clients et le serveur.

III. L'Application :

1- Interfaces User :

Les captures d'écran de l'application montrent une interface simple et intuitive pour la gestion d'un serveur de chat en ligne. L'interface permet de visualiser les utilisateurs connectés, de **recevoir et d'afficher les messages envoyés** par les utilisateurs, ainsi que de se **connecter** et se déconnecter du serveur :

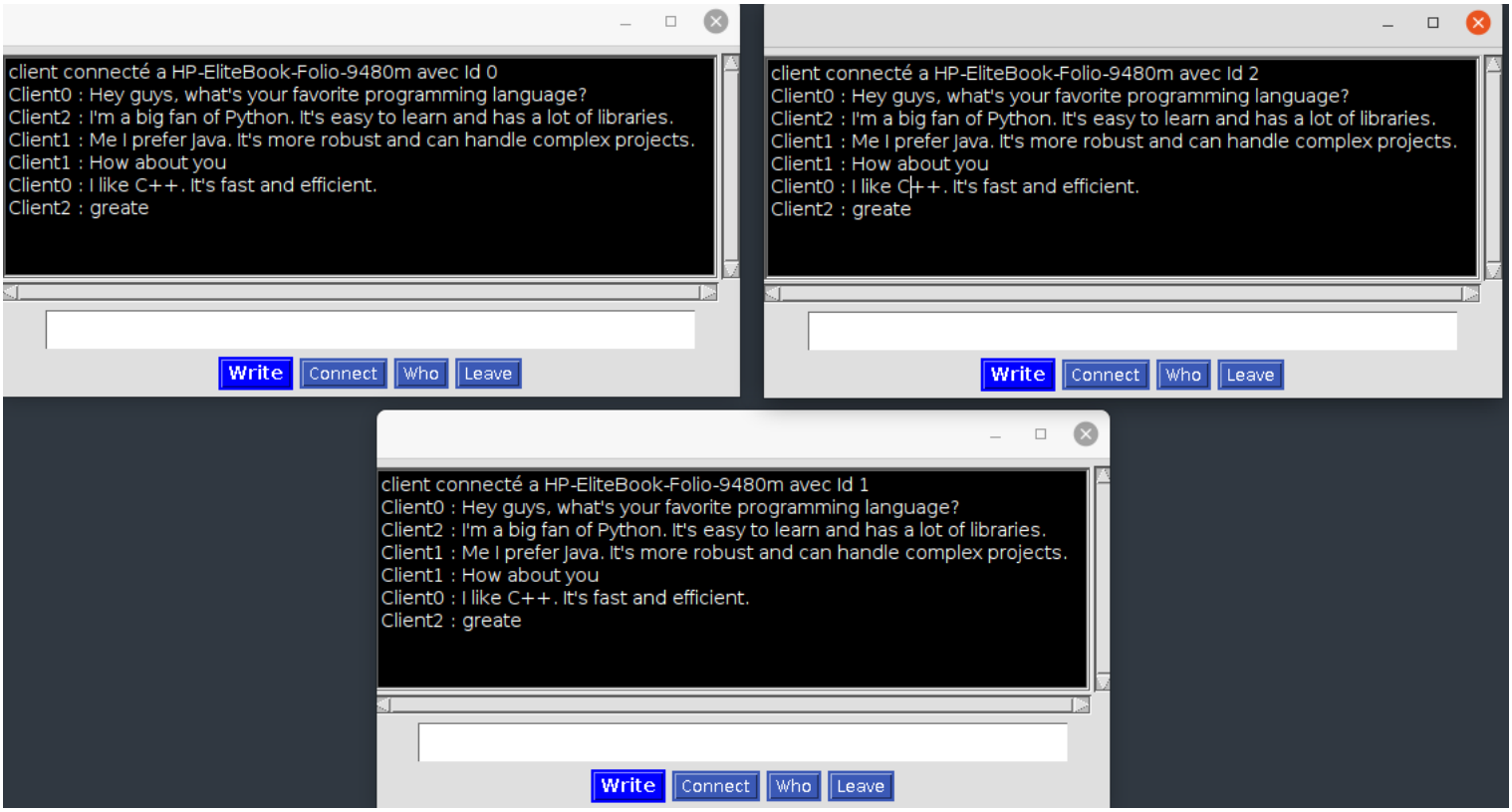

Figure 7: Conversation between three people

l'application de chat permet également aux utilisateurs de voir la liste des autres utilisateurs connectés en cliquant sur le bouton "**who**". Cela affichera une liste de tous les clients actuellement connectés au forum, ce qui peut être utile pour identifier les personnes avec lesquelles on peut communiquer :

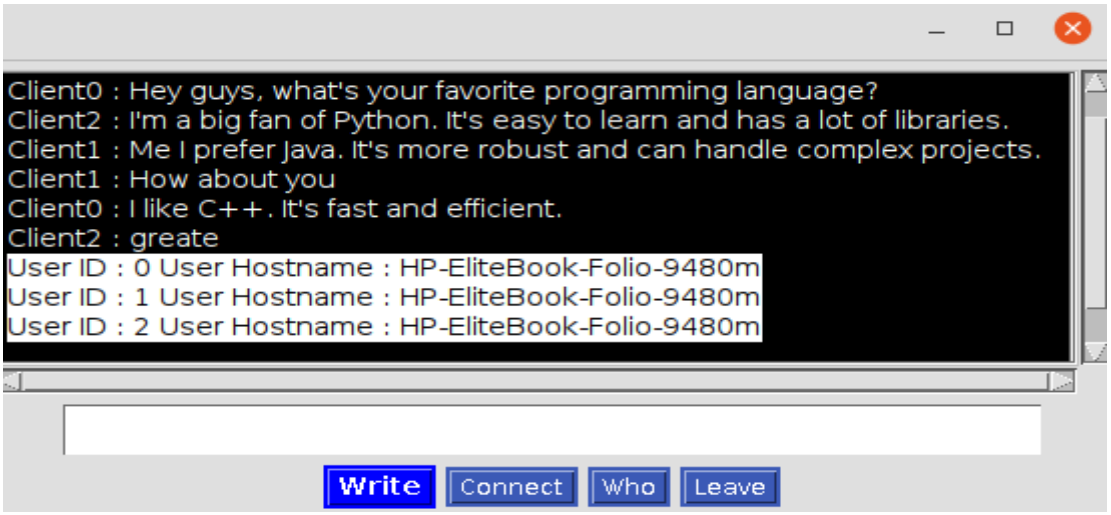

Figure 8: Visualiser les utilisateurs connectés

Lorsqu'un client se connecte au forum, un identifiant unique lui est attribué pour identifier ses messages. Dans les captures d'écran fournies, le client avec l'identifiant 1 a cliqué sur le bouton "**leave**" pour se déconnecter du forum Cette action permet de mettre à jour la liste des utilisateurs connectés pour les autres clients du forum :

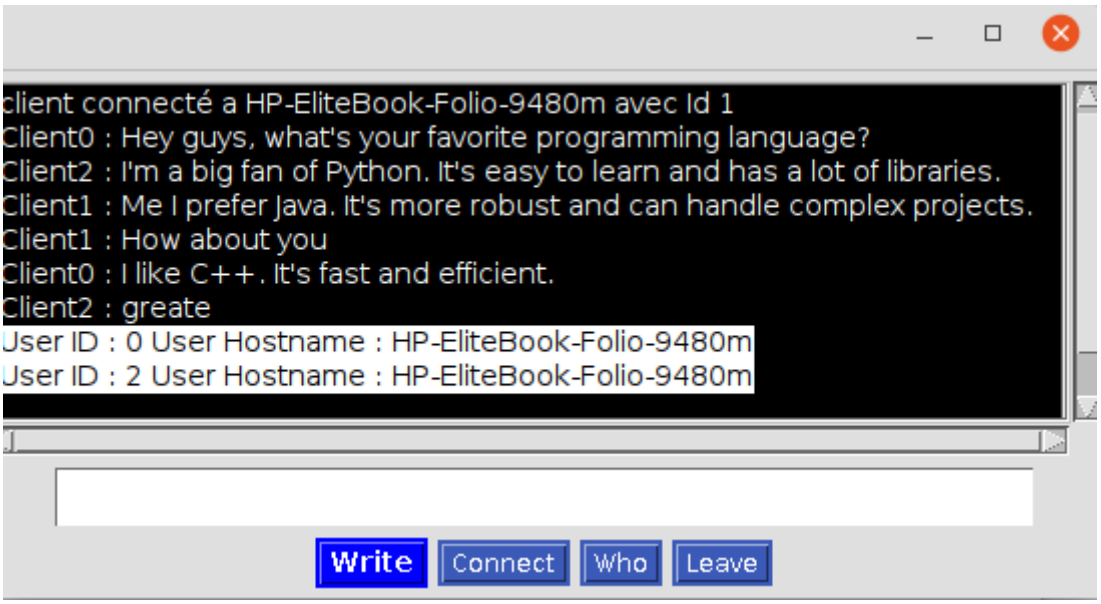

Figure 9: Déconnecter du Forum

IV. Conclusion :

Le mini projet que j'ai réalisé avait pour objectif la gestion d'un Forum de discussion pouvant faire intervenir un nombre quelconque d'intervenants utilisant une application User. J'ai créé une interface simple et intuitive permettant de visualiser les utilisateurs connectés, de recevoir et d'afficher les messages envoyés par les utilisateurs, ainsi que de se connecter et se déconnecter du serveur.

L'application a été réalisée en utilisant le RMI et est composée des fichiers Forum.java, ForumImpl.java, User.java, UserImpl.java, proxy.java et proxyImpl.java. J'ai implémenté les différentes classes et interfaces en respectant les consignes et en ajoutant des fonctionnalités supplémentaires pour améliorer l'expérience utilisateur.

En conclusion, ce mini projet m'a permis de renforcer mes compétences en programmation Java et en utilisation du RMI. J'ai pu créer une application fonctionnelle et adaptée aux besoins spécifiques d'un forum de discussion en ligne.