TP2 Java: chat JavaFx

**Rapport technique**

À l’attention de M. Carrino

**Java –TP2**Fleury Anthony, Guerne Jonathan   
Mai 2017

# Introduction

Le but de ce projet est de réaliser un Chat en java permettant à un ou plusieurs clients d’envoyer des messages à un serveur. Le chat doit avoir un seul canal contenant tous les messages c.à.d. qu’il n’y a pas de notion de message privé entre clients par exemple.

Il y a deux types de messages à pouvoir envoyer et un troisième bonus :

* Envoie de message texte classique
* Communication vocale (comme un appel pas comme des enregistrements)
* (**Bonus**) envoie de fichiers

L’application doit être réaliser en JavaFx

# Fonctionnement de l’application

Notre implémentation possède deux parties distinctes : la partie client et la partie server. Un client s’enregistre au près d’un serveur afin de pouvoir recevoir les différentes Messages échangés sur le chat. Plusieurs clients peuvent être connectés en simultané au même serveur.

L’application client est une application graphique mais le serveur est un simple programme en ligne de commande.

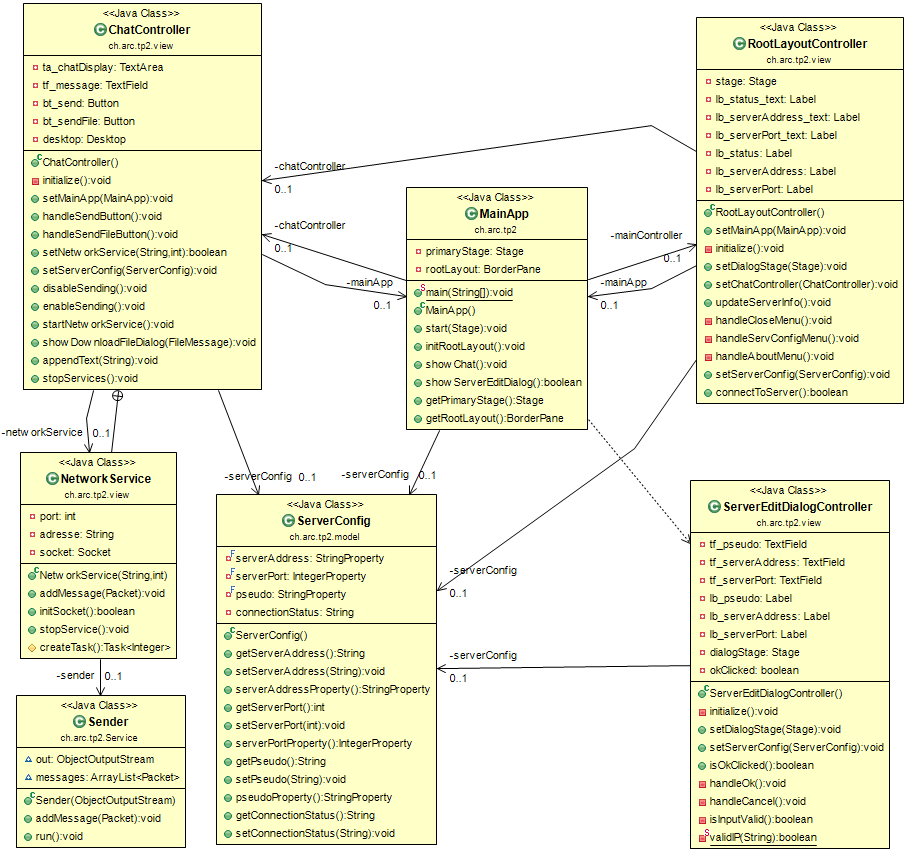
## Packets

|  |  |
| --- | --- |
|  | Dans le but d’obtenir plus d’information qu’un simple String nous avons choisi de transférer des objets entre le serveur et les clients. Pour organiser les objets envoyés nous avons créé une interface Packet sérialisable dont tous les objets à envoyer hériterons.  Les implémentations concrète de Packet dans notre projet sont TextMessage pour envoyer un message dans le chat en précisant l’auteur et FileMessage pour transmettre un fichier.  On peut remarquer qu’il n’a pas de packet lié à la transmission de données audio. Nous n’avons pas eu le temps d’implémenter cette partie (plus de détail dans la conclusion) |

## Serveur

|  |  |
| --- | --- |
| Diagramme de classe du server | Pour gérer plusieurs client le serveur utilise un ServerDispatcher qui va permettre de gérer les envoie en broadcast à tous ces clients, chaque client possède un socket, un clientListener et un clientSender ces éléments sont liés entre eux par la classe clientInfo.  ClientSender et ClientListener permettent respectivement d’envoyer des Packets à un client précis et d’écouter les Packets émanent d’un client précis.  Après avoir reçu un message le ClientListener le transmet au ServerDispatcher qui valui se charger de transmettre ce message en broadcast à tous les clients (même l’émetteur original).  À noter que le diagramme de classe de Packet devrait être présent ici puisqu’il est utilisé par les classes ServerDispatcher, ClientSender et ClientListener. Il n’a pas été rajouté pour des soucis de lisibilité, la même problématique est apparue pour le diagramme de classe de la partie client |

## Client



Le client est une application JavaFx qui va permettre d’afficher le chat mais également de configurer l’adresse et le port du serveur. Il Utilise le Modèle MVC pour transmettre les données de configurations vers l’affichage.

L’affichage principal se fait dans un RootLayoutController, Le RootLayoutController possède un ChatController qui va se charger d’afficher le chat (textArea pour les messages reçus, textfileds pour l’envoie et autres boutons). Lors de la configuration du serveur on utilise ServerEditDialogController pour modifier le pseudo de l’utilisateur sur ce serveur, l’adresse du serveur ou encore le port.

La class Network service est une classe interne à ChatController elle va permet d’initier la connexion au serveur et elle pourra ensuite être utilisé comme un service en arrière-plan pour recevoir les différents Packets émanant du serveur. L’envoie se fait lui avec la classe Sender qui sera instancié par le NetworkService.

# Choix Architectural

Nous avions opté dès le début du projet pour un chat permettant la connexion de plusieurs clients sur le même serveur. Le serveur a été inspiré et réadapter d’un exemple trouvé en ligne à cette adresse : <http://inetjava.sourceforge.net/lectures/part1_sockets/InetJava-1.9-Chat-Client-Server-Example.html>

L’utilisation du ServerDispatcher permettant de facilement gérer le transit des messages entre les clients. L’utilisation du polymorphisme pour les objets transmis (tous sont des Packets) nous a permis de travailler exclusivement avec la classe parent dans le serveur sans porté d’intérêt à quelle classe concrète avait été transmise, ça a rendu les modifications plus simples et ça permettra de facilement implémenter l’ajout de nouveaux types de Packets.

Le modèle MVC utiliser pour le client permet également d’avoir un squelette d’application plus solide qui lui aussi permettra de facilement implémenter de nouvelle fonctionnalité par la suite tout en restant propre.

# Déploiement et utilisation

Pour que le programme de chat puisse fonctionner il est impératif que le serveur soit d’abord mis en place. Il peut être lancé sur n’importe quelle machine il faut juste être attentif au fait que les clients et le serveur doivent pouvoir communiquer. Il faut donc vérifier que le serveur soit accessible avec le couple adresse, port.

|  |  |
| --- | --- |
| Screenshot du programme de chat | Une fois l’application lancée le programme tente de se connecter au serveur localhost avec le port 52017.  L’utilisateur peut changer ces infos en ouvrant la fenêtre de configuration sous Edit > Edit Server Configuration  Pour envoyer un message on peut le tapez dans le textField en bas de l’écran et ensuite utiliser la touche Enter ou le bouton Send pour l’envoyer. L’envoie de fichier se fait en cliquant sur le bouton avec le logo upload (à côté de send) |

# Conclusion

Nous sommes globalement satisfaits du programme, il est dommage de ne pas avoir pu implémenter la gestion d’un chat audio mais la difficulté d’implémentation, notre emploie du temps chargé ainsi que la faible quantité de temps mise à notre disposition nous ont encouragés à laisser tomber ce point au profit du transfert de fichier qui c’est avérer plus simple à mettre en place.

Le projet pourrait être plus approfondit en implémentant, comme dit plus haut, la gestion de l’audio ou encore en ajoutant des fonctionnalités liées à la réception de fichier (lien pour retrouver les fichiers dans le chat, popup pour télécharger ou non le fichier).