Documentation de projet Programmation Distribuée Java

Heytem BOUMAZA
11926806

Yacine BELMAATI CHERKAOUI
11513398

M1 Informatique 2019-2020

Table des matières

Section I

Introduction

Le but de ce projet est de simuler un programme serveur qui lit un flux de données des réseaux sociaux sous forme de messages et commentaires puis calcule les 3 meilleurs messages en fonction d'un certain critère, après cela, les clients qui se connectent au serveur vont recevoir la réponse au format texte et xml.

Dans le but de simuler un flux de données de réseaux sociaux, le serveur lit les données d'un fichier local, ligne après ligne et calcule en même temps les 3 meilleurs messages, au fur et à mesure de la lecture, les 3 meilleurs messages sont mis à jour, et par conséquent , les clients peuvent recevoir des réponses différentes à chaque fois

Section II

Installation

II.1 Prérequis

Pour pouvoir exécuter le client et le serveur la version 8 de java doit être installé.

II.2 Compilation des sources

Le code a été écrit et exécuté sous l'environnement suivant :
— java version "1.8.0_221"
— Java(TM) SE Runtime Environment (build 1.8.0_221-b11)
— Java HotSpot(TM) 64-Bit Server VM (build 25.221-b11, mixed mode)

II.3 Lancement du serveur

Commencez par exécuter FileReaderThread, qui peut exécuter indépendamment sur une machine virtuelle différente et calculer les 3 meilleurs messages et et vous devriez pouvoir voir la phrase 'established connection with remote object', après cela, vous devez démarrer le serveur en exécutant la classe ServMain pour déployer le pool de threads du serveur.

II.4 Lancement du client

Vous exécutez la classe TEST qui créera plusieurs instances client et vous devriez pouvoir voir la réponse qui devrait être en deux formats, texte brut et xml, pour chaque client.

Section III

Description des classes

III.1 Conteneur de données

III.1.1 Classe Message

Signature

```
public class Message implements  Comparable<Message>,Serializable
```

Constructeur

```
public Message
(int idMessage,int idUser,String message , Date date,String user)
```

Arguments

- idMessage : identifiant du message
- idUser : identifiant de l'auteur
- message : contenu du message
- date : date de création du message
- user : pseudonyme de l'utilisateur

Getters

isState

```
public boolean isState()
```

Retourne vrai si le message est actif, faux sinon.

getDate

```
public Date getDate()
```

Permet d'obtenir la date de création de du message
Retour : date de création de du message

getImportance

```
public int getImportance()
```

Permet d'obtenir la date de création de du message.

getImportance

```
public int getImportance()
```

Permet d'obtenir l'importance du message qui est un entier positif.
Retour : importance du message.

getIdMessage

```
public int getIdMessage()
```

Permet d'obtenir l'identifiant du message.
Retour : identifiant du message qui est un entier positif.

getIdUser

```
public int getIdUser()
```

Permet d'obtenir l'identifiant de l'auteur.
Retour : identifiant de l'auteur qui est un entier positif.

getMessage

```
public String getMessage()
```

Permet d'obtenir le contenu du message
Retour : le contenu du message qui est une chaîne de caractére

getDate

```
public Date getDate()
```

Permet d'obtenir la date de création du message.
Retour : la date de création du message qui est un objet de type Date.

getUser

```
public String getUser()
```

Permet d'obtenir le pseudonyme de l'auteur.
Retour : le pseudonyme de l'auteur qui est un objet de type String.

Setter

setState

```
public void setState(boolean state)
```

— state : Nouvel état du message
Permet de modifier l'état

setImportance

```
public void setImportance(int importance)
```

— state : Nouvelle importance du message
Permet de modifier l'impotance du message.

setScore

```
public void setScore(int score)
```

— score : Nouveau score du message
Permet de modifier le score du message.

setIdUser

```
public void setIdUser(int idUser)
```

— idUser : Nouveau identifiant de l'auteur
Permet de modifier l'identifiant de l'auteur.

setMessage

```
public void setMessage(String message)
```

— message : Nouveau contenu du message
Permet de modifier le contenu du message.

setDate

```
public void setDate(Date date)
```

— date : Nouvelle date du message
Permet de modifier la date du message.

setDate

```
public void setDate(Date date)
```

— date : Nouvelle date du message
Permet de modifier la date du message.

setUser

```
public void setUser(String user)
```

— user : Nouveau pseudonyme de l'auteur
Permet de modifier le pseudonyme de l'auteur.

printMessage

```
public void printMessage()
```

Permet d'afficher le message à l'écran.

printMessage

```
public void printMessage()
```

Permet d'afficher le message à l'écran.

compareTo

```
@Override
public int compareTo(Message o)
```

Méthode de l'interface Comparable<Message>
Retour : La difference d'importance entre l'instance de l'objet et le paramétre, cette méthode est appelée automatiquement lorsque nous invoquons la méthode de tri 'sort()' à partir du package de collections.

decreaseScore

```
public void decreaseScore()
```

Diminue le score du message représenté par l'instance.

III.1.2 Classe Comment

Constructeurs

```
public Comment (Date date , int idCommentaire , int idUser ,
String Comment, String user , int pidCommentaire)
```

Arguments
- date : date de création du commentaire
- idCommentaire : identifiant commentaire doit être unique
- idUser : identifiant de l'auteur du commentaire
- Comment : contenu du commentaire
- user : pseudonyme de l'utilisateur
- pidCommentaire : commetaire père dans l'arboressence des commentaire

```
public Comment (int idCommentaire,int idUser,Date date,
String Comment, String user ,int pidMessage)
```

Arguments
- date : date de création du commentaire
- idCommentaire : identifiant commentaire doit être unique
- idUser : identifiant de l'auteur du commentaire
- Comment : contenu du commentaire
- user : pseudonyme de l'utilisateur
- pidMessage : Message auquel le commentaire est attaché

Getter

getDate

```
public Date getDate()
```

Permet d'obtenir la date de création de du commentaire.

Retour : Date de création du commentaire

getIdCommentaire

```
public int getIdCommentaire()
```

Permet d'obtenir l'identifiant du commentaire

Retour : Identifiant du commentaire qui est un entier positif.

getIdUser

```
public int getIdUser()
```

Permet d'obtenir l'identifiant de l'auteur.

Retour : Identifiant de l'auteur qui est un entier positif.

getUser

```
public String getUser()
```

Permet d'obtenir le pseudonyme de l'auteur.

Retour : Pseudonyme de l'auteur qui est une chaîne de caractére.

getPidCommentaire

```
public int getPidCommentaire()
```

Permet d'obtenir le commentaire père dans l'arboresence des commentaire.

Retour : L'identifiant du commentaire père ou -1 si le commentaire est une racine.

getPidMessage

```
public int getPidMessage()
```

Permet d'obtenir le message attaché au commentaire si le commentaire est une racine dans l'arboresence des commentaire

Retour : L'identifiant du message ou -1 si le commentaire n'est pas la racine.

Setters

setDate

```
public void setDate(Date date)
```

— date : Nouvelle date

Permet de modifier la date.

setIdCommentaire

```
public void setIdCommentaire(int idCommentaire)
```

— idCommentaire : Nouveau identifiant pour le commentaire

Permet de modifier l'identifiant du commentaire.

setIdUser

```
public void setIdUser(int idUser)
```

— idUser : Nouveau identifiant pour l'auteur

Permet de modifier l'identifiant de l'auteur.

setComment

```
public void setComment(String comment)
```

— comment : Nouveau contenu

Permet de modifier le contenu du commentaire.

setUser

```
public void setUser(String user)
```

— user : Nouveau pseudonyme

Permet de modifier le pseudonyme de l'auteur.

setPidCommentaire

```
public void setPidCommentaire(int pidCommentaire)
```

— pidCommentaire : Nouveau identifiant du commentaire père

Permet de modifier l'identifiant du commentaire père.

setPidMessage

```
public void setPidMessage(int pidMessage)
```

— pidMessage : Nouveau identifiant du message

Permet de modifier l'identifiant du message.

III.2 Implémentation client-serveur

III.2.1 Classe FileReadThread

Utilité c'est la classe thread qui contient toutes les méthodes pour lire le fichier et calculer les 3 meilleurs messages, ce thread est initié indépendamment avant même le serveur car il s'exécute sur une autre machine virtuelle.

Methodes

nextRand

```
public int nextRand()

  Génère un nombre aléatoire compris entre 1 et 3 et le renvoie;
```

getMessageTokens

```
public Message getMessageTokens(String Line)
```

Prend en entrée la ligne courante lue par le thread et extrait les parties du message pour créer un nouvel objet message et le renvoie.

getCommentTokens

```
public comment getCommentTokens(String line)
```

Prend en entrée la ligne courante lue par le thread et extrait les parties du commentaire pour créer un nouvel objet comment et le renvoie.

updateImportanceOfMessagesWhenCommentReadOrDecreased

```
public void  updateImportanceOfMessagesWhenCommentReadOrDecreased(Comment c)
```

Cette méthode prend un objet de type comment en entrée et met à jour le score d'importance du message auquel ce commentaire appartient, cette méthode est appelée à chaque fois qu'un commentaire est lu ou que le score d'un commentaire est diminué.

decreaseScores

```
public void decreaseScores()
```

Cette méthode est appelée toutes les 30 secondes, elle diminue le score de chaque message et commentaire lus de 1.

updateThe3BestMessages

```
public void updateThe3BestMessages()
```

Appelé chaque fois qu'un commentaire ou un message est lu (toutes les 1 à 3 secondes), cette méthode calcule les 3 meilleurs messages en triant simplement la liste des messages en fonction de leur score d'importance, puis en choisissant les 3 premiers.

removeInactiveMessages

```
public void removeInactiveMessages()
```

Cette méthode est appelée toutes les 30 secondes, après la diminution du score, pour supprimer tous les messages dont le score est égal à 0, et par conséquent, ils ne peuvent pas être considérés comme les meilleurs messages.

getThe3BestMessages

```
public void getThe3BestMessages()
```

Renvoie la liste des 3 meilleurs messages, c'est la méthode appelée par le serveur sur le remote object.

run

```
public void run()
```

La méthode d'exécution invoquée par le thread une fois créé, il utilise les méthodes mentionnées ci-dessus, pour plus de détails, vous devriez examiner le code source.

III.2.2 Interface SocialNetworkRemoteInterface

Signature

```
public interface SocialNetworkRemoteInterface extends Remote
```

Methodes

getThe3BestMessages

```
List<Message> getThe3BestMessages() throws RemoteException
```

La méthode qui sera appelée sur le remote object.

III.2.3 Classe ServMain

Signature

```
public class ServMain
```

Constructeur

```
public ServMain(int port, int PoolSize)
```

Arguments
- port : le numéro de port à utiliser pour le serveur.
- PoolSize : taille du thread de pool à déployer sur le serveur.

Getters

getServerSocket

```
public ServerSocket getServerSocket()
```

Retourne le socket de serveur.

Setters

setServerSocket

```
public void setServerSocket(ServerSocket serverSocket)
```

Permet de modifier le socket du serveur.

Methodes

handleRequests

```
public void handleRequests()
```

Cette méthode gère les demandes des clients à venir en instanciant une instance de la classe ServerResponderSlave qui répondra au client.

main

```
public static void main(String[] args)
```

La méthode principale où nous lançons le serveur et le préparons pour recevoir les demandes des clients.

III.2.4 Classe ServerResponderSlave

Signature

```
public class ServerResponderSlave implements Runnable
```

Methodes

run

```
public void run()
```

```
Méthode de l'interface Runnable il accède à l'objet distant et appelle
la méthode getThe3BestMessages  puis il envoie le résultat au client, dans
deux formats, texte brut et xml.
```

III.2.5 Classe SocialNetworkClient

Signature

```
public class SocialNetworkClient
```

Constructeur

Signature

```
public SocialNetworkClient(int id,int port)
```

Arguments
- id : L'identifiant de client.
- port : Numéro de port.

Methodes

connectToServer

```
public void connectToServer() throws Exception
```

C'est la méthode qui permet à l'instance cliente de se connecter au serveur, de recevoir une réponse et de l'afficher.

III.3 Le protocole handler

III.3.1 Classe Handler

Signature

```
public class Handler extends URLStreamHandler
```

Methodes

openConnection

```
@Override
protected URLConnection openConnection(URL u) throws IOException
```

Retourne une instance de la classe SocialNetworkURLConnection.

III.3.2 Classe SocialNetworkContentHandler

Signature

```
public class SocialNetworkContentHandler extends ContentHandler
```

Methodes

getMessageTokens

```
public Message getMessageTokens(String line)
```

Arguments
— line : la ligne qui a été lu.
Déconstruit la ligne qui a été lue et construit un objet message et le renvoie.

getCommentTokens

```
public Comment getCommentTokens(String line,int type)
```

Arguments
— line : la ligne qui a été lu.
— type : le type de commentaire lu (0 pour un commentaire associé à un autre commentaire, 1 pour un commentaire associé à un message).
Déconstruit la ligne qui a été lue et construit un objet commentaire et le renvoie.

getContent

```
public Object getContent ( URLConnection urlc ) throws IOException
```

utilise les 2 méthodes ci-dessus pour lire le fichier, puis retourne l'objet qui a été construit, soit un message, soit un commentaire.

delete

```
public void delete(String filename, int startline, int numlines)
```

Arguments

— filename :URL du fichier.

— startline :le numéro de la ligne à partir de laquelle la suppression commence .

— numlines :nombre de lignes à supprimer.

Cette méthode a été écrite pour deux raisons : premièrement, pour s'assurer que chaque fois que getContent() est appelé, il lira une nouvelle ligne parce que la ligne qui a été lue sera supprimée, deuxièmement, pour simuler un flux de données et de cette façon nous n'avons pas besoin de connaître le nombre de lignes à l'avance.

III.3.3 Classe SocialNetworkContentHandlerFactory

Signature

```
public class SocialNetworkContentHandlerFactory implements ContentHandlerFactory
```

Methodes

createContentHandler

```
public ContentHandler createContentHandler ( String mimeType )
```

Renvoie une instance de la classe

III.3.4 Classe SocialNetworkURLConnection

Signature

```
public class SocialNetworkURLConnection extends URLConnection
```

Constructeur

```
public SocialNetworkURLConnection ( URL url ) throws MalformedURLException
```

Arguments

— l'URL de la ressource

Methodes

connect

```
@Override
public void connect() throws IOException
```

III.4 Le teste

III.4.1 La classe TEST

Methodes

main

```
public static void main(String[] args)
```

Ici nous créons de nombreux clients et les connectons au serveur, vous pouvez créer autant de clients que vous le souhaitez chacun avec son identifiant unique et les connecter au serveur en appelant connectToServer().