CHATHACK

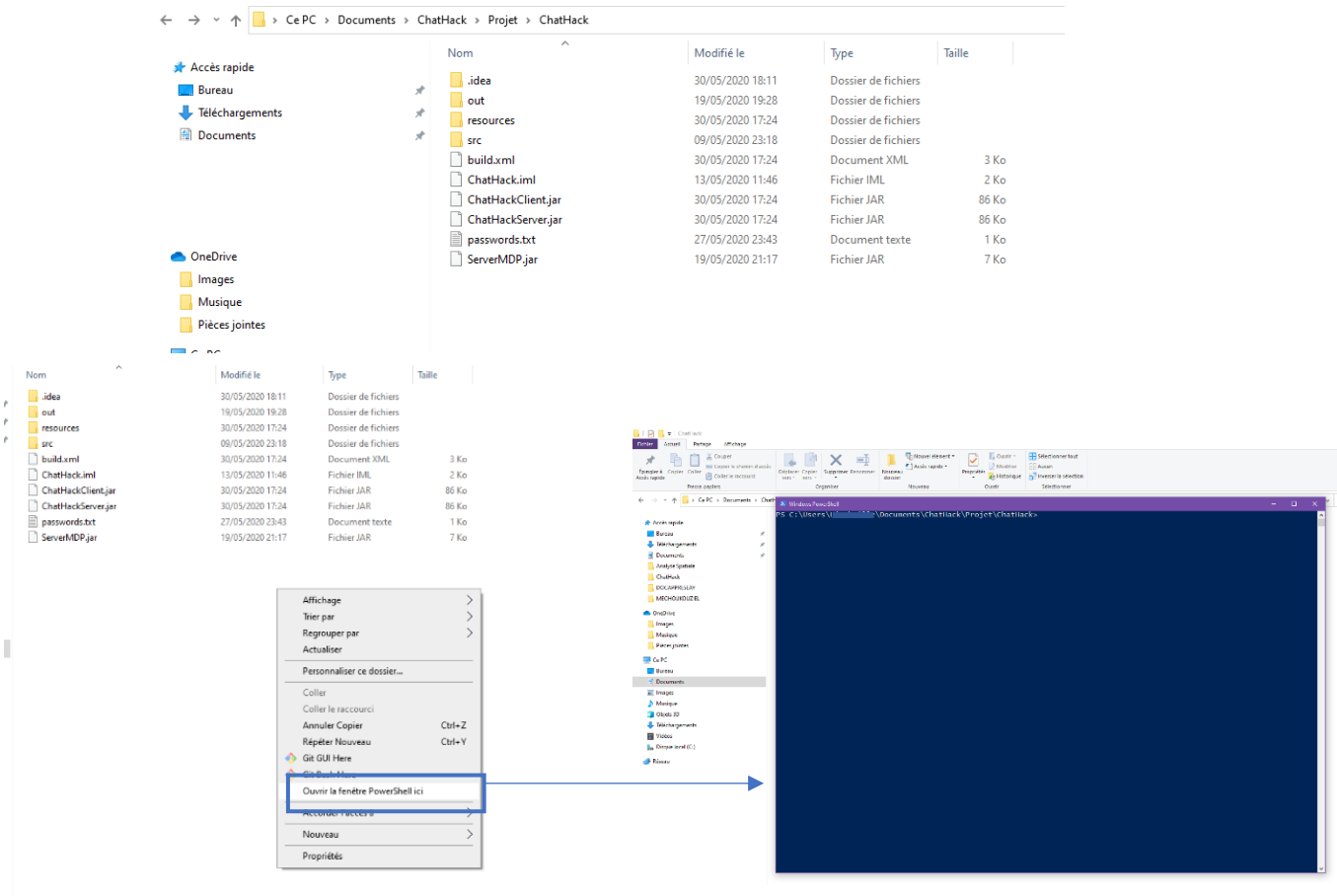
Manuel Utilisateur

MARTI Emilie & MECHOUK Lisa – ESIPE INFO2

SOMMAIRE

Génération des jars et Javadoc

Pour toutes les étapes de génération, il faut se placer dans le dossier contenant le projet et ouvrir un terminal (shift + clique droit), il faut **bien vérifier que le fichier build.xml** est présent :



Pour cette partie les commandes doivent être réalisées dans un ordre précis en commençant par l'initialisation, la compilation, création des jars. Il est possible une fois la génération des jars de les supprimer en faisant un clean.

Initialisation

La partie initialisation permet de préparer à la compilation. De ce fait elle doit être réalisée en premier. Pour ce faire il faut se placer dans un terminal à la racine du projet (cf page 3).

Une fois le terminal ouvert, il faut y entrer la commande suivante :

```
ant init
```

```
PS C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack> ant init
Buildfile: C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack\build.xml

init:
    [mkdir] Created dir: C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack\out\production
    [mkdir] Created dir: C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack\build

BUILD SUCCESSFUL
Total time: 0 seconds
PS C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack>
```

Le message BUILD SUCCESSFUL nous indique que tout a bien fonctionné

Compilation

La partie compilation permet compiler les projets avant de créer les Jars. De ce fait elle doit être réalisée avant la partie de création des jars. Pour ce faire il faut se placer dans un terminal à la racine du projet (cf page 3).

Une fois le terminal ouvert, il faut y entrer la commande suivante :

```
ant compile
```

```
PS C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack> ant compile
Buildfile: C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack\build.xml

init:

compile:
    [javac] C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack\build.xml:23: warning: 'includeantruntime' was not se
t, defaulting to build.sysclasspath=last; set to false for repeatable builds
    [javac] Compiling 38 source files to C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack\out\production

BUILD SUCCESSFUL
Total time: 1 second
```

Le message BUILD SUCCESSFUL nous indique que tout a bien fonctionné

Création des jars

La partie de création des jars permet créer les jars depuis les fichiers .class. De ce fait elle doit être réalisée après la partie de compilation. Pour ce faire il faut se placer dans un terminal à la racine du projet (cf page 3). Les jars créés seront placés dans un dossier build.

Une fois le terminal ouvert, il faut y entrer la commande suivante :

```
ant makeJars
```

```
PS C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack> ant makeJars
Buildfile: C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack\build.xml

init:

compile:
    [javac] C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack\build.xml:23: warning: 'includeantruntime' was not se
t, defaulting to build.sysclasspath=last; set to false for repeatable builds

makeJars:
      [jar] Building jar: C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack\build\ChatHackServer.jar
      [jar] Building jar: C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack\build\ChatHackClient.jar

BUILD SUCCESSFUL
Total time: 0 seconds
```

Le message BUILD SUCCESSFUL nous indique que tout a bien fonctionné

Clean

La partie du clean permet de supprimer les jars déjà existant. Elle doit être réalisé lorsque l'utilisateur souhaite supprimer les jars précédemment créés. Pour ce faire il faut se placer dans un terminal à la racine du projet (cf page 3).

Une fois le terminal ouvert, il faut y entrer la commande suivante :

```
ant clean
```

```
PS C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack> ant clean
Buildfile: C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack\build.xml

init:

clean:
   [delete] Deleting directory C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack\build
   [delete] Deleting directory C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack\out\production

BUILD SUCCESSFUL
Total time: 0 seconds
```

Le message BUILD SUCCESSFUL nous indique que tout a bien fonctionné

Javadoc

La partie javadoc permet de générer la Javadoc depuis les fichiers sources. Pour ce faire il faut se placer dans un terminal à la racine du projet (cf page 3). La javadoc sera alors présente dans le dossier doc également présent à la racine du projet.

Une fois le terminal ouvert, il faut y entrer la commande suivante :

```
ant javadoc
```

```
PS C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack> ant javadoc
Buildfile: C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack\build.xml

javadoc:
    [mkdir] Created dir: C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack\doc
  [javadoc] Generating Javadoc
  [javadoc] Javadoc execution
  [javadoc] Loading source files for package fr.upem.net.tcp.chathack.client...
  [javadoc] Loading source files for package fr.upem.net.tcp.chathack.server...
  [javadoc] Loading source files for package fr.upem.net.tcp.chathack.utils.context...
  [javadoc] Loading source files for package fr.upem.net.tcp.chathack.utils.frame...
  [javadoc] Loading source files for package fr.upem.net.tcp.chathack.utils.reader.frame...
  [javadoc] Loading source files for package fr.upem.net.tcp.chathack.utils.reader.utils...
  [javadoc] Loading source files for package fr.upem.net.tcp.chathack.utils.visitor...
  [javadoc] Constructing Javadoc information...
  [javadoc] Standard Doclet version 14.0.1
  [javadoc] Building tree for all the packages and classes...
  [javadoc] Building index for all the packages and classes...
  [javadoc] Building index for all classes...

BUILD SUCCESSFUL
Total time: 1 second
```

Le message BUILD SUCCESSFUL à la fin nous indique que tout a bien fonctionné

Il est également possible de supprimer la Javadoc grâce à la commande suivante :

```
ant clearDoc
```

Pour ce faire il faut également se placer dans un terminal à la racine du projet (cf page 3). La javadoc sera alors supprimée.

```
clearDoc:
   [delete] Deleting directory C:\Users\Lisabeille\Documents\ChatHack\Projet\ChatHack\doc

BUILD SUCCESSFUL
Total time: 1 second
```

Le message BUILD SUCCESSFUL à la fin nous indique que tout a bien fonctionné

Il est également possible de lancer plusieurs de ces commandes en même temps, tout en respectant l'ordre indiqué dans la documentation :

```
ant init compile makeJars
```

Mise en place de l'environnement

L'intégralité du projet doit être lancé sous JRE13.

Serveur MDP

Un serveur MDP est fourni pour faire la vérification du couple {login + mot de passe} lorsqu'un client se connecte. Comme le serveur MDP ne gère pas les inscriptions, il faut créer un fichier contenant les alias + mot de passe déjà connus par le serveur.

Le format du contenu du fichier de sauvegarde est le suivant :

login$password

L'alias et le mot de passe sont séparés par un dollar `$`.

On peut ensuite lancer le server MDP en se plaçant dans le dossier qui contient le jar avec la commande suivante dans un terminal, le fichier des mots de passe doit se trouver au même endroit que le jar ServerMDP :

```
java -jar ./ServerMDP.jar <port_mdp> < nom_fichier_complet >
```

```
java -jar ./ServerMDP.jar 7778 .\resources\save.txt
```

- L'argument <port_mdp> est un port libre de la machine.

Serveur ChatHack

Le serveur ChatHack est fourni sous la forme d'un jar. Pour le lancer, il faut se placer dans le dossier où l'exécutable se trouve et faire la commande suivante :

```
java -jar ./ChatHackServer.jar <port_chathack> <address_mdp> <port_mdp>
```

```
java -jar ./ChatHackServer.jar 7777 localhost 7778
```

- L'argument <port_chathack> est un port libre de la machine.
- L'argument <address_mdp> est l'adresse de la machine qui a lancé le serveur MDP.
- L'argument <port_mdp> est le port sur lequel écoute le serveur MDP.

Client ChatHack (Windows)

Le client ChatHack est fourni sous la forme d'un jar. Le client se définit par un alias.

Il peut se connecter à ChatHack de deux façons différentes : avec ou sans mot de passe. Seuls les clients déjà enregistrés (présents dans le fichier de sauvegarde du serveur MDP) peuvent se connecter avec une paire login + mot de passe.

Sans mot de passe

Pour le lancer, il faut se placer dans le dossier où l'exécutable se trouve et faire la commande suivante :

```
java -jar ./ChatHackClient.jar <login> <address_chathack> <port_chathack> <path_fichier>
```

```
java -jar ./ChatHackClient.jar etienne localhost 7777 C:\Users\Users\Documents\ChatHack\Projet\ChatHack\resources\
```

Attention : Le path doit se terminer par « \ ».

- L'argument <login> est l'alias du client. Un alias déjà enregistré auprès du serveur MDP ou déjà présent parmis les clients connectés de ChatHack résulte en refus de connexion par le serveur ChatHack.
- L'argument <address_chathack> est l'adresse de la machine qui a lancé le serveur ChatHack.
- L'argument <port_chathack> est le port sur lequel écoute le serveur ChatHack.
- L'argument <path_fichier> est le chemin des fichiers que l'on souhaite envoyer et recevoir

Avec mot de passe

Pour le lancer, il faut se placer dans le dossier où l'exécutable se trouve et faire la commande suivante :

```
java -jar ./ChatHackClient.jar <login> <password> <address_chathack> <port_chathack> <path_fichier>
```

```
java -jar ./ChatHackClient.jar arnaud townhall localhost 7777 C:\Users\User\Documents\ChatHack\Projet\ChatHack\resources\
```

Attention : Le path doit se terminer par « \ ».

- L'argument <login> est l'alias du client. Un alias non enregistré auprès du serveur MDP résulte en refus de connexion par le serveur ChatHack.
- L'argument <password> est le mot de passe du client.

Une paire <login> + <password> qui ne correspond pas à la paire <login> + <password> enregistrée auprès du serveur MDP (par exemple, si le mot de passe est mal écrit) résulte en un refus de connexion par le serveur ChatHack.

- L'argument <address_chathack> est l'adresse de la machine qui a lancé le serveur ChatHack.
- L'argument <port_chathack> est le port sur lequel écoute le serveur ChatHack.
- L'argument <path_fichier> est le chemin des fichiers que l'on souhaite envoyer et recevoir

Client ChatHack (Linux)

Le client ChatHack est fourni sous la forme d'un jar. Le client se définit par un alias.

Il peut se connecter à ChatHack de deux façons différentes : avec ou sans mot de passe. Seuls les clients déjà enregistrés (présents dans le fichier de sauvegarde du serveur MDP) peuvent se connecter avec une paire login + mot de passe.

Sans mot de passe

Pour le lancer, il faut se placer dans le dossier où l'exécutable se trouve et faire la commande suivante :

```
java -jar ./ChatHackClient.jar <login> <address_chathack> <port_chathack> <path_fichier>
```

```
java -jar ./ChatHackClient.jar etienne localhost 7777 /home/user/Desktop/resources/
```

Attention : Le path doit se terminer par « / ».

- L'argument <login> est l'alias du client. Un alias déjà enregistré auprès du serveur MDP ou déjà présent parmis les clients connectés de ChatHack résulte en refus de connexion par le serveur ChatHack.
- L'argument <address_chathack> est l'adresse de la machine qui a lancé le serveur ChatHack.
- L'argument <port_chathack> est le port sur lequel écoute le serveur ChatHack.
- L'argument <path_fichier> est le chemin des fichiers que l'on souhaite envoyer et recevoir

Avec mot de passe

Pour le lancer, il faut se placer dans le dossier où l'exécutable se trouve et faire la commande suivante :

```
java -jar ./ChatHackClient.jar <login> <password> <address_chathack> <port_chathack> <path_fichier>
```

```
java -jar ./ChatHackClient.jar arnaud townhall localhost 7777 /home/user/Desktop/resources/
```

Attention : Le path doit se terminer par « / ».

- L'argument <login> est l'alias du client. Un alias non enregistré auprès du serveur MDP résulte en refus de connexion par le serveur ChatHack.
- L'argument <password> est le mot de passe du client.

Une paire <login> + <password> qui ne correspond pas à la paire <login> + <password> enregistrée auprès du serveur MDP (par exemple, si le mot de passe est mal écrit) résulte en un refus de connexion par le serveur ChatHack.

- L'argument <address_chathack> est l'adresse de la machine qui a lancé le serveur ChatHack.
- L'argument <port_chathack> est le port sur lequel écoute le serveur ChatHack.
- L'argument <path_fichier> est le chemin des fichiers que l'on souhaite envoyer et recevoir

Exemple de mise en place de ChatHack
(Windows)

On se place dans la même machine. Deux utilisateurs, alice et bob, seront enregistrés auprès du serveur MDP et un utilisateur, felix, se connectera à ChatHack sans s'enregistrer.

Dans le dossier \ChatHack\ on copie les exécutables ServerMDP.jar, ChatHackServer.jar et ChatHackClient.jar.

On crée le fichier save.txt dans le dossier resources qui doit contenir :

```
alice$meteor77
bob$ragnAr12
```

Ce fichier représente deux utilisateurs inscrits sur ChatHack, alice et bob, avec leurs mots de passe respectifs séparés par un dollar $.
On créé un dossier permettant de stocker les fichiers à envoyer et à recevoir, son chemin correspond au <path_fichier> et doit se terminer par \.
On lance le serveur MDP avec la commande suivante :

```
java -jar ./ServerMDP.jar  7778 .\resources\save.txt
```

.\resources\ représente le chemin du dossier utilisé pour les fichiers.

On lance le serveur ChatHack avec la commande suivante :

```
java -jar ./ChatHackServer.jar 7777 localhost 7778
```

On lance les clients ChatHack avec les commandes suivantes :

```
java -jar ./ChatHackClient.jar alice meteor77 localhost 7777 .\resources\
java -jar ./ChatHackClient.jar bob ragnAr12 localhost 7777 .\resources\
java -jar ./ChatHackClient.jar felix localhost 7777 .\resources\
java -jar ./ChatHackClient.jar alice localhost 7777 .\resources\
```

Seuls les trois premiers clients seront acceptés dans le chat ChatHack. Le quatrième client a essayé d'utiliser un alias déjà pris et enregistré auprès du serveur MDP donc il ne sera pas accepté.

Exemple de mise en place de ChatHack

(Linux)

On se place dans la même machine. Deux utilisateurs, alice et bob, seront enregistrés auprès du serveur MDP et un utilisateur, felix, se connectera à ChatHack sans s'enregistrer.

Dans le dossier /ChatHack/ on copie les exécutables ServerMDP.jar, ChatHackServer.jar et ChatHackClient.jar.

On crée le fichier save.txt dans le dossier resources qui doit contenir :

```
alice$meteor77
bob$ragnAr12
```

Ce fichier représente deux utilisateurs inscrits sur ChatHack, alice et bob, avec leurs mots de passe respectifs séparés par un dollar $.
On créé un dossier permettant de stocker les fichiers à envoyer et à recevoir, son chemin correspond au <path_fichier> et doit se terminer par /.
On lance le serveur MDP avec la commande suivante :

```
java -jar ./ServerMDP.jar 7778 ./resources/save.txt
```

./resources/ représente le chemin du dossier utilisé pour les fichiers.

On lance le serveur ChatHack avec la commande suivante :

```
java -jar ./ChatHackServer.jar 7777 localhost 7778
```

On lance les clients ChatHack avec les commandes suivantes :

```
java -jar ./ChatHackClient.jar alice meteor77 localhost 7777 ./resources/
java -jar ./ChatHackClient.jar bob ragnAr12 localhost 7777 ./resources/
java -jar ./ChatHackClient.jar felix localhost 7777 ./resources/
java -jar ./ChatHackClient.jar alice localhost 7777 ./resources/
```

Seuls les trois premiers clients seront acceptés dans le chat ChatHack. Le quatrième client a essayé d'utiliser un alias déjà pris et enregistré auprès du serveur MDP donc il ne sera pas accepté.

ChatHack

Chat global

Pour écrire un message sur le chat global, qui sera lu par tous les clients connectés de ChatHack, il suffit d'écrire un message dans la console et de taper sur entrée comme l'exemple ci-dessous :

```
arnaud has entered the chat
Welcome to ChatHack !
etienne has entered the chat
alice has entered the chat
Bonjour à tous
arnaud: Bonjour à tous
```

```
alice has entered the chat
Welcome to ChatHack !
arnaud: Bonjour à tous
Bonjour arnaud
alice: Bonjour arnaud
```

```
etienne has entered the chat
Welcome to ChatHack !
alice has entered the chat
arnaud: Bonjour à tous
alice: Bonjour arnaud
```

Message privé

Afin d'envoyer un message privé à un client connecté, il suffit d'écrire un message précédé par @<alias_destinataire> et de taper sur entrée.
Le destinataire doit accepter la demande de connexion privée avec la commande $accept. Il peut la refuser avec $refuse auquel cas ils ne pourront pas échanger.

Arnaud

```
Welcome to ChatHack !
etienne has entered the chat
alice has entered the chat
Bonjour à tous
arnaud: Bonjour à tous
alice: Bonjour arnaud
@alice Bonjour Alice
Send to : alice -> Bonjour Alice
Message receive from : alice -> Bonjour Arnaud
```

Alice

```
alice has entered the chat
Welcome to ChatHack !
arnaud: Bonjour à tous
Bonjour arnaud
alice: Bonjour arnaud
Someone wants to send private messages to you. Do you accept ? ($accept/$refuse)
$accept
User accepted the private connection
Message receive from : arnaud -> Bonjour Alice
@arnaud Bonjour Arnaud
Send to : arnaud -> Bonjour Arnaud
```

Envoi de fichier

Afin d'envoyer un fichier à un client connecté, il suffit d'écrire le nom du fichier précédé de /<alias_destinataire>. Les fichiers reçus auront comme nom : received_nomdufichier.

Attention : Les envois de fichier ne peuvent se faire qu'une fois une connexion établie entre deux utilisateurs.

/etienne kitty.png

Etienne

```
@etienne hello
Send to : etienne -> hello
/etienne kitty.png
```

Arnaud

```
Someone wants to send private messages to you. Do you accept ? ($accept/$refuse)
$accept
User accepted the private connection
Message receive from : arnaud -> hello
Receiving file kitty.png ...
```

Demande de déconnexion

Un client peut demander à se déconnecter de ChatHack avec la commande &.

```
&
arnaud has disconnected
You are disconnected

Process finished with exit code 0
```

Récapitulatif des commandes

Message -> permet d'envoyer directement un message en broadcast

@login message -> Permet d'envoyer un message privé au client nommé login

/login file.png -> Permet d'envoyer un fichier au client nommé login

$accept -> Permet d'accepter une demande de connexion privé

$refuse -> Permet de refuser une demande de connexion privée

& -> Permet la déconnexion d'un client

Fonctionnalités et Bugs

Fonctionnalités	Etat
Message en Broadcast	
Message privé	
Envoie de fichier privé	
Déconnexion	
Connexion au serverMDP	

Fonctionnel

Non Fonctionnel

ChatHack possèdent toutes les fonctionnalités attendues et n'ont aucun bug connu à ce jour (30/05/2020)