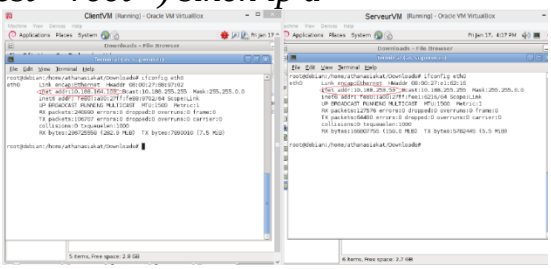
Au début, on va implémenter les exercices 1, 2, 3 dans une VM – on aura 2 projets – un pour le serveur et un pour le client, et on test le code avec deux consoles – une pour le serveur et une pour le client. Puis on va changer le code pour qu'il marche avec les deux VMs (une VM aura le rôle Serveur et l'autre le rôle Client.

Programmes Serveur/Client sur les machines virtuelles

Avant de compiler le code source, nous devons vérifier les adresses inet du Client et Serveur. Il faut assurer qu'ils sont différents et donc les deux machines virtuelles seront indépendantes. Afin de vérifier les adresses inet, nous devons taper au terminal:

>: ifconfig eth0 (si on est « root ») sinon ip a



Ctrl + Alt + F3 : New terminal
ls : voir ce qu'il exist dans le repos (current)
cd nomRepos : pour entrer dans le nomRepos
sudo su : pour connecter sur root

La figure 1 : présente les résultats reçus en tapant la commande au-dessus.
Nous pouvons voir que les deux VM ont différentes adresses inet et ainsi ils sont indépendants.

Dans la phase de test, on aura besoin de l'adresse inet de Serveur *(pour cet exemple : 10.188.253.53);* afin que le Client puisse acquérir une connexion avec lui.

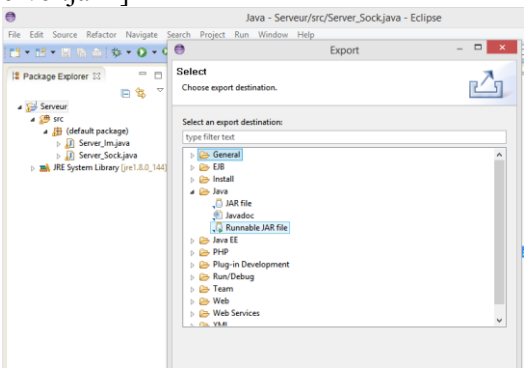
Les programmes Java peuvent communiquer via des sockets (Pour faciliter la communication entre un serveur et un client). Le serveur s'exécute en premier et attend pour les demandes de client.

Pour le serveur, nous allons définir un socket serveur qui fonctionne sur le port 9999:
ServerSocket serverSocket = new ServerSocket (9999);

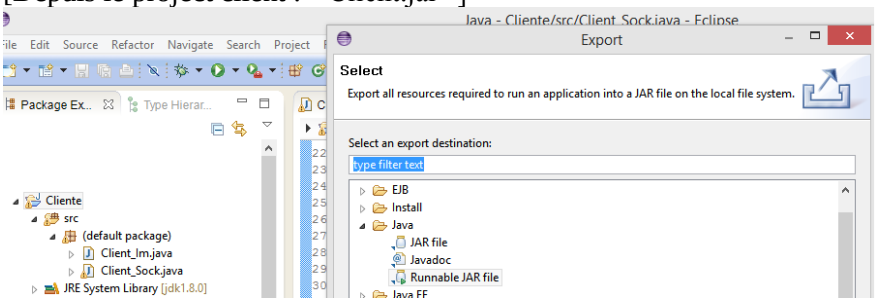
Le serveur doit écouter et accepter la connexion:
Socket clientSocket = serverSocket.accept ();
Le client essaie de se connecter au port 9999 sur la machine spécifiée (adresse inet).
Socket socket = new Socket (inetAdresse, 9999);

#Program 1- Programme Serveur / Client
(Envoyer des messages)

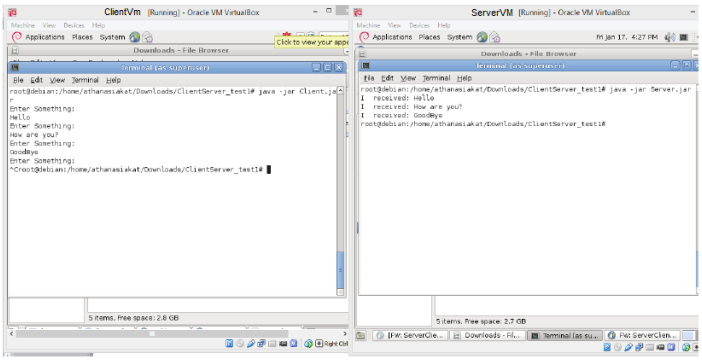
Ce programme, c'est votre premier effort pour établir un lien entre les deux VMs.
Dans ce cas-là, le client envoie un message simple (une simple chaîne) au serveur et le serveur le reçoit.
Au début, on lance le serveur, puis le Client.
Une fois fini le code pour le Client et le Serveur on produit les exécutables, jar.
[Depuis le project serveur : ''Server.jar'']



[Depuis le project client : ''Client.jar'']



Ainsi, du côté serveur, on tape:
>: java -jar Server.jar
Du côté client, nous tapons:
>: java -jar Client.jar

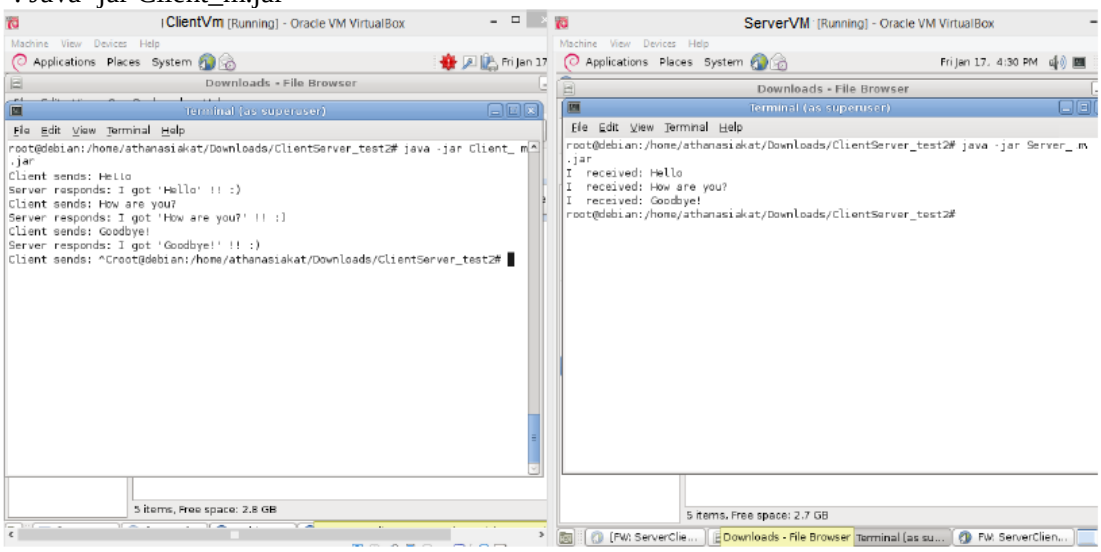


La figure 2 présente les résultats obtenus en exécutant le programme

#Programme 2- Programme Serveur / Client (Envoyer / Recevoir des messages)

Dans ce cas-là:
Le client envoie un message simple (une chaîne simple) au serveur et le serveur le reçoit. Ensuite, le serveur renvoie au client qu'il a reçu le message avec le message entier.

On produit deux exécutables (pour le serveur et pour le client) et puis du côté serveur, on tape :
>: java -jar Server_m.jar
Du côté client, on tape:
>: Java -jar Client_m.jar



La figure 3 présente les résultats obtenus en exécutant le programme.

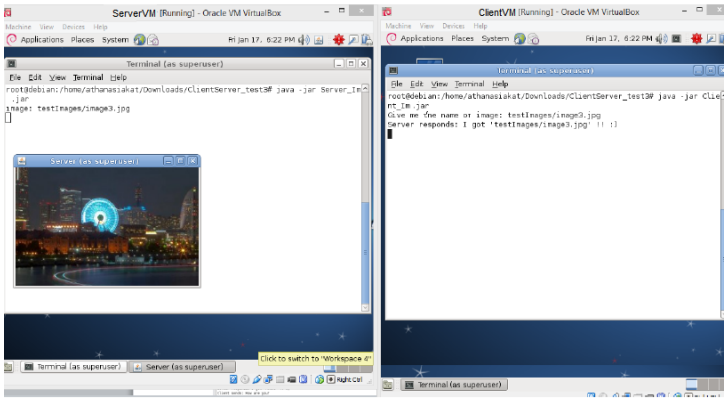
Program 3 - Programme Serveur / Client
(Envoyer un chemin d'image / Serveur sert la demande)

Dans ce cas-là :

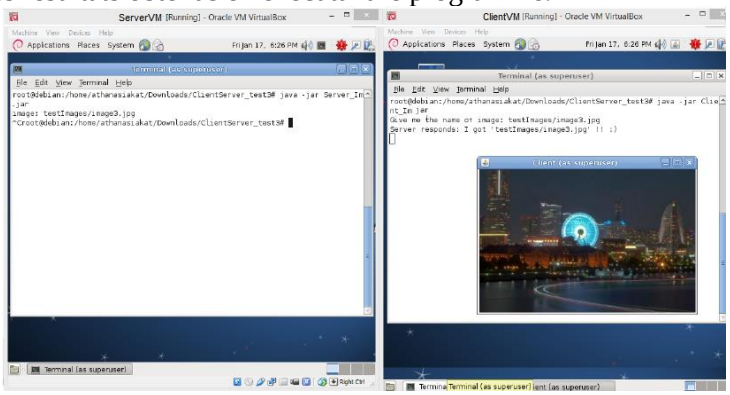
Le client envoie le chemin complet d'une image (repos) avec le nom d'image (String) au serveur; le serveur le reçoit et ouvre l'image de son côté. Ensuite, le serveur renvoie au client qu'il a reçu l'image avec le chemin et le nom d'image.

On va créer trois méthodes dans la classe du serveur.
Plus précisément :
La methode: *connecter () : définitions sockets pour la communication*
La methode: *imageName (clientSocket): qui renvoie le chemin complet de l'image avec son nom.*
La methode: *sendImage (image): qui envoie l'image au client.*
Dans la classe du client, nous allons également créer trois méthodes. Plus précisément :
La methode *connexion () : définitions sockets pour la communication*
La methode *nameOfImage (serveur):* qui envoie le chemin complet de l'image ainsi que son nom.
La methode *readImage ():* qui présente l'image dans le côté client.

On produit deux exécutables (pour le serveur et pour le client) et puis du côté serveur, on tape :
>: java -jar Server_im.jar
Du côté client, on tape:
>: Java -jar Client_im.jar



La figure 4 présente les résultats obtenus en exécutant le programme.



https://www.tecmint.com/ifconfig-command-examples/