Laboratório 1

Objectivos:

- Aceder remotamente a outro sistema através de cliente Secure Socket Shell (SSH)
- Medir latência no envio de mensagens entre processos locais e remotos
- Desenvolver cliente de serviço de mensagens com a tecnologia Java Remote Method Invocation (RMI)

1. Realize o acesso a uma máquina virtual (VM OVirt) cujo dados de acesso (IP, nome de utilizador e palavra-chave) foram entregues ao grupo.
   a. Instale um cliente SSH (Exemplos para o sistema operativo Windows: Bitvise - https://www.bitvise.com/ssh-client-download; Putty - https://www.putty.org/)
   b. Use o nome de utilizador e palavra-chave enviados por email
   c. Verifique que o terminal lhe dá acesso à consola do sistema operativo
   d. Verifique que consegue transferir ficheiros entre o computador local e a VM Ovirt

2. Instale o JDK 8 usando o comando "`sudo yum install java-1.8.0-openjdk-devel`" (https://linuxize.com/post/install-java-on-centos-7/#install-openjdk-8-jdk)

3. Utilizando o projeto de *sockets*, apresentado na aula e em anexo no ficheiro `Sockets.zip` (disponível no moodle), analise os resultados em cenários com cliente e servidor na máquina local versus cliente na máquina local e servidor na máquina VM OVirt.

4. Considere um serviço de mensagens desenvolvido com a tecnologia *Java Remote Method Invocation* (RMI), e cujos contratos se apresentam em seguida (contratos e serviço disponíveis em `Contracts.jar` e `Server.jar`, no moodle):

```
package contracts;
public interface IMessagingService extends Remote {
   /* for connection test only */
   String ping(String in) throws RemoteException;
   /* register a new user and its message box */
   void register(String usrName, IMsgBox mb) throws RemoteException;
   /* remove user from the messaging service */
   void unRegister(String usrName) throws RemoteException;
   /* get the name of all registered users */
   List<String> getRegisteredUsers() throws RemoteException;
   /* send a message to all registered users */
   void sendMulticastMessage(String usrName, String msg) throws RemoteException;
   /* get the message box of user 'usrName' */
   IMsgBox connetUser(String usrName) throws RemoteException;
}

public interface IMsgBox extends Remote {
   /* to be called by the service or another client */
   void messageNotification(String usr, String msg) throws RemoteException;
}
```

a. Realize uma aplicação cliente que:
   i. Regista e desregista um utilizador no serviço;
   ii. Envia mensagens para todos os utilizadores registados usando o método `sendMulticastMessage`;
   iii. Recebe mensagens de outros utilizadores através de objetos que implementam a interface `IMsgBox`;
   iv. Envia mensagens privadas para um dos utilizadores usando o método `connectUser` e o método `userNotification` da interface `IMsgBox`.
b. Teste o cliente com o servidor a correr no seu computador. Por omissão, o servidor disponibilizado lança o *registry* (serviço de registo) no porto 7000 e o serviço, com o nome `MessagingService`, no porto 7001. Caso seja necessário pode indicar outros portos na linha de comandos.
c. Teste o cliente com o servidor a correr na VM OVirt do grupo.

5. Ligue o seu cliente ao servidor geral, a correr no IP 10.62.73.69, e com o *registry* no porto 7000. O nome do serviço é `MessagingService`. Verifique que a comunicação funciona com os restantes grupos. O nome do utilizador tem de ser único no serviço sendo por isso sugerido o seguinte esquema de nomes `<turma>-G<número do grupo>-<nome>`.