Activitat 2.2: Stubs i Skeletons RMI

1.- Dissenyar i implementar un client-servidor xat sobre RMI.

L'objecte remot ChatServer implementa la interfície IChatServer que disposa del mètodes essencials demanats a l'enunciat:

```
void Connectar(IControl ctrl) throws RemoteException;
void Desconnectar(IControl ctrl) throws RemoteException;
String[] usuaris() throws RemoteException;
void Enviar(IControl ctrl, String missatge) throws RemoteException;
```

He variat els paràmetres del mètodes per tal de que el client passi en tot moment un objecte de control que implementi la interfície IControl amb els mètodes:

```
public String getID() throws RemoteException;
public void Talk(String id, String text) throws RemoteException;
```

Aquest objecte ens servirà per:

a) Identificar l'usuari que esta realitzant l'acció.
b) Mantindre un llistat d'usuaris als quals enviar el text que s'envia al servidor.

La interfície IControl hereta de Remote i per tant pot actuar com un objecte remot a la banda del client, d'aquesta forma quan algú envii text al servidor, aquest farà un recorregut pel llistat de IControl i anirà executant el mètode Talk que serà executat a cada client.

Per testejar:

Compilació i generació de stubs i skeletons:

```
javac *.java
rmic ChatServer
rmic Control
```

Terminal 1:

```
rmiregistry
```

Terminal 2:

```
java ChatServer
```

Terminal 3 (es possible executar tants clients com volguem):

```
java ChatClient
```

El client ens denamarà un nick, aquest no te perquè ser únic al servidor, i a continuació podrem enviar missatges que els rebrà tothom que estigui connectat. Per llistar els usuaris teclejarem “/users” i per sortir “/quit”.

2.- Compileu el codi RMI amb "rmic -keep". Expliqueu quina és la funció de cadascun d'aquests fitxers.

L'aplicació rmic generà els stubs i skeletons que s'usen per RMI. En el nostre cas hem d'executar-ho per a cada objecte remot:

```
rmic -keep ChatServer
rmic -keep Control
```

El primer genera:

ChatServer_Skel.java/ChatServer_Skel.class
ChatServer_Stub.java/ChatServer_Stub.class

El segon genera:

Control_Skel.java/Control_Skel.class
Control_Stub.java/Control_Stub.class

Els skeletons són utilitzats a la banda del servidor per oferir l'objecte remot, mentre que els stubs es la representació de l'objecte remot a la banda del client i possibilita un accés transparent. A més, també compleixen la funcionalitat d'adaptació de les dades entre les diferents arquitectures (Marshalling i Unmarshalling).

Autor: Sergio Blanco Cuaresma