Exercise 1

Introduction to Java Chat

1 Goal

The goal of this exercise is to remind you some concepts of Java. To that end you will implement the first building blocks of a local chat application. You will reuse these basic blocks in the following exercises.

2 Guidelines

We present here in more details the different indications that will help you to implement the local chat application. Furthermore, we will give you several classes and interfaces in order to make your work easier. Please have a look at the end of the statement to find information how to download the code.

2.1 Application Overview

The application simulates a very simple distributed chat but in a local environment. To that end, the main class (`CentralizedChat`) creates an instance of a `ChatServer` and two instances of `ChatClient` and test several methods of the `ChatClient`: (1) `connect(String username)`, (2) `disconnect()` and (3) `sendMessage(IMessage msg)`. Please have a look at the implementation of `CentralizedChat` to know in more details how it works.

After a call to the `connect(String username)` method of a `ChatClient`, this client must be connected with the `ChatServer`, i.e., the server holds a reference to the client. After a call to the `disconnect()` method of a `ChatClient`, this client must be disconnected from the `ChatServer`, i.e., the server does not hold anymore a reference to the client. The result of a call to the `sendMessage(IMessage msg)` method of a `ChatClient` is that the `ChatServer` simply outputs the `msg` in a console. Finally, your implementation of a `ChatServer` must store in a list the different connected `ChatClient`.

The specification of the `ChatClient`, `ChatServer` and `Message` is given through Java interfaces (via `IChatClient`, `IChatServer` and `IMessage` respectively). Please read them carefully.

The user should be able to launch the application via the following command:

```
java ch.epfl.lpd.ids.CentralizedChat
```

And here is a possible output of this command:

```
Client 1 is connected.
 Message received: Header: header 1 Data: message 1
Client 2 is connected.
 Message received:  MyHeader: header 2 MyData: message 2
Client 1 is disconnected.
Client 2 is disconnected.
```

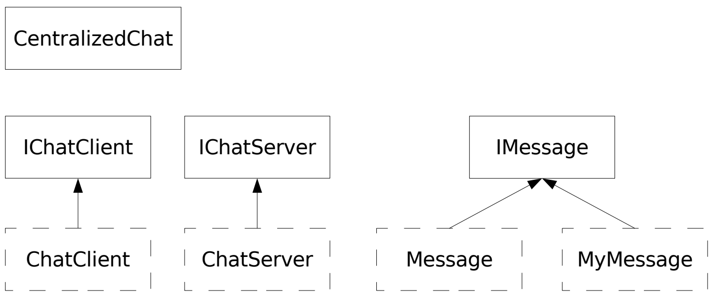

Figure 1: Chat Architecture

2.2 General Architecture

The chat application should follow the architecture presented in Figure 1.

You will receive the code of the interfaces `IChatClient`, `IChatServer`, `IMessage` as well as the class `CentralizedChat` represented by the plain rectangles of Figure 1. Your job is to implement the classes represented by the dashed rectangles (i.e., `ChatClient`, `ChatServer`, `Message` and `MyMessage`). The idea underneath having two different implementations of `IMessage` is to make you understand how the polymorphism in Java works.

In short, the main class `CentralizedChat` calls the different methods of the instances of `ChatClient` which in turn call the methods on the instance of the `ChatServer`.

3 Sources

You can download the different sources of the application here:

http://lpdwww.epfl.ch/teaching/ids.html

Look on this page for the *Src of Ex1*.

4 Due

You have to give, for March 21th, the complete source code of your application. You can send your code, **without** the compiled classes, in a **.zip** file archive (or **.tgz**) to the following email address: *Sebastien.Baehni@epfl.ch*. The archive **has** to contain a script file used for compiling your code (either on Windows or on Linux). Your email should arrive before noon and its subject has to be: *IDS EX1*. Please put also in the content of your mail the firstname and name of your partner.

References

[1] Java API Documentation. *http://java.sun.com/j2se/1.5.0/docs/api*.