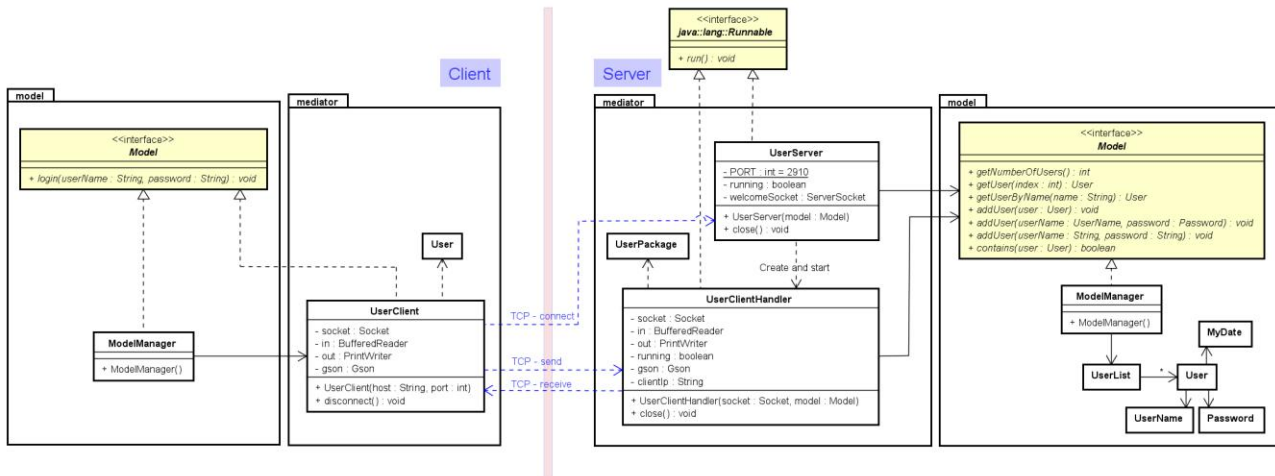


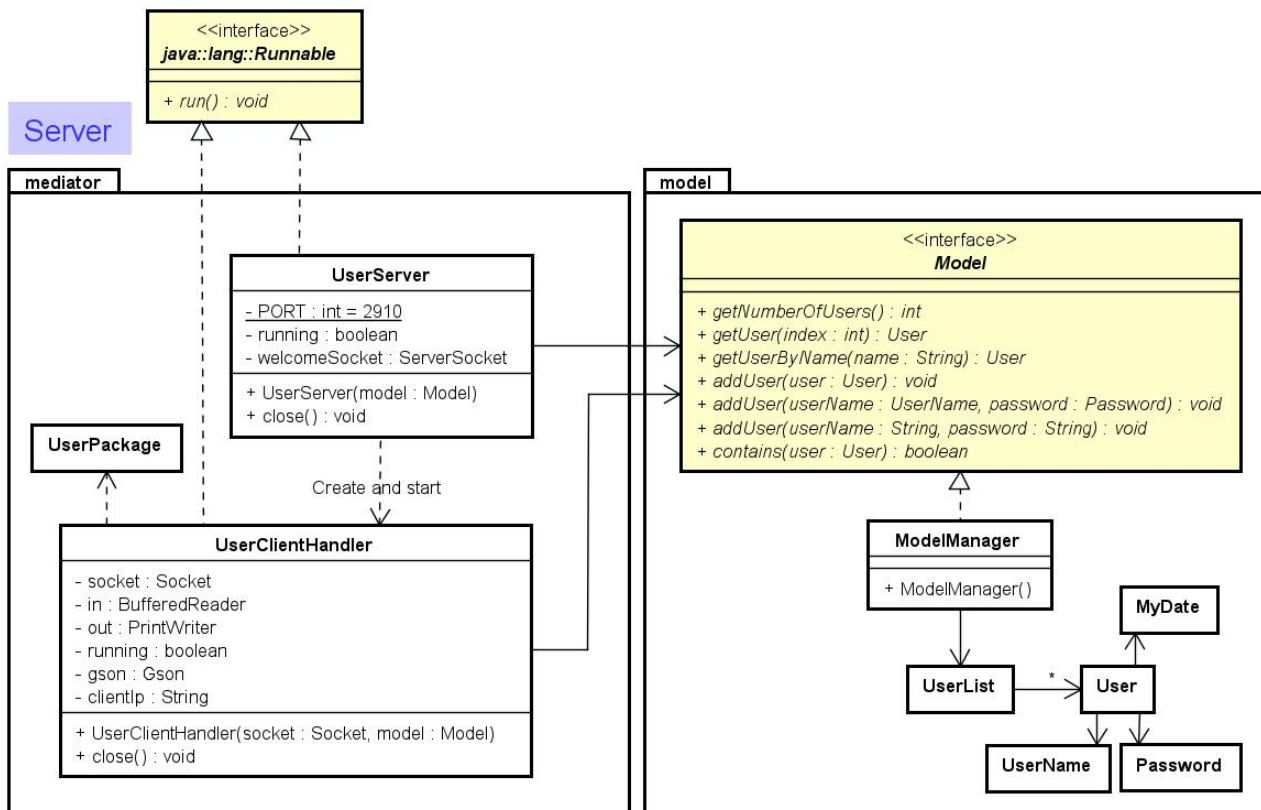
## Exercise: A login using server and client model

The purpose for this exercise is to implement a program to login from a client application onto a server application. The program is a simple console version having a server model and an empty client model and using JSON to send login objects from client to server containing username and password.



## Step 1: Server side

This server-side application looks like this:



A few comments to the server side application

- Model package is given (uploaded). The interface contains methods to add and get logged in users. A user contains username and password and a timestamp when created / logged in. Adding a user may

throw exceptions if username or password do not follow some requirements, or if a user with the same username is already in the list.

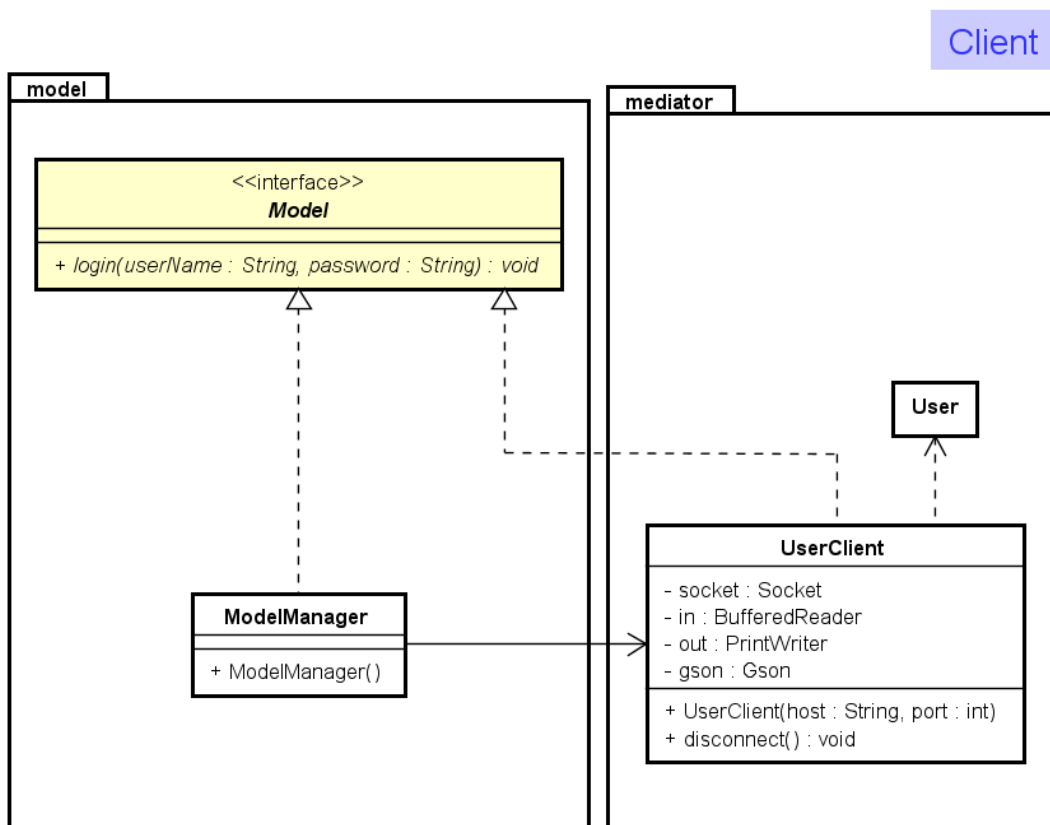
- Class `UserPackage` is a class containing a username and password as simple Strings – to be used when converting from Json what the client is sending, when trying to login (be added to list). This class is also given.
- Classes `UserServer` and `UserClientHandler` are the two classes to implement.
- Class `UserServer` implements `Runnable` and in its `run` method it has an infinite loop waiting for connecting clients. When a client is connected, an object of `UserClientHandler` is created, a thread is created with this object, and the thread is started.
- Class `UserClientHandler` is handling the communication with one client. It also implements `Runnable` and its `run` method do the following:
  - Read a (Json-)string from the client [Note: You have to add gson library/jar to the IntelliJ module]
  - Convert the Json string to a `UserPackage`
  - In a try-catch block call the model method `addUser(String, String)`. If success, send the string "Success: you are now logged in" to the client, if in catch block, then send the exception message instead.

Try to implement the method with proper printouts, exception handling and controlled client termination.

- Create a class with a `main` method in which 1) create the model and 2) create and start a Thread with a `UserServer` object, i.e. starting the server.

## Step 2: Client side

Create another module in IntelliJ to the client-side application. The client-side application looks like this:



A few comments to the client side application

- `Model` interface is given (uploaded). Note that it has the same name as the interface on the server side. Therefore, you should create the client and server in different IntelliJ modules.
- Class `User` (also given) is in contents and variable names identical to `UserPackage` on the server side. The idea is to send a `User` object converted to JSON, (and the part you already made: convert it back on the server side to a `UserPackage` object).
- `UserClient` implements the `Model` interface (like `ModelManager` does). In `UserClient` constructor you connect to the server and in the `login` method you:
  - Create a `User` object, convert it to a JSON string and send this to the server [*Note: You have to add `gson library/jar` to the IntelliJ module*]
  - Receive a string from server (if this was the type you were sending from server)
  - If the string does not start with "Success" or you catch an exception, then throw an exception with a proper message (in most cases, use the result from server).
- Class `ModelManager` implements the `Model` interface and in the `login` method, you delegate to the `UserClient` instance variable.
- Create a class with a `main` method in which you get username and password from keyboard and call the `login` method from the model. Continue asking for the two strings until you call the `login` method without catching an exception. If an exception is caught then print out the message.