Further Java Ticklet 4*

In order to gain a star in the mark sheet you must complete this exercise. Completing the exercise does not gain you any credit in the examination. In this exercise you will extend your implementation of the Java chat server from Workbook 4 to send mobile code to the client. The code that you send to the client should display a Swing chat client interface on the client machine, thereby upgrading the client from a text-based application to a graphical one.

The first thing to write is a simple Swing GUI which is able to interact with your Java chat server. The precise layout of the GUI is up to you, but at the very least it should contain a box in which the user can type in new messages, an area of the screen devoted to displaying messages from other users, and a means of changing the nickname of the user. Test your Swing GUI with your existing server and make sure it works reliably.

Start by starting and cloning the ticklet4star repository on Chime https://www.cl.cam.ac.uk/teaching/current/FJava/ticklet4star. Then copy your implementation of the Java chat server for Ticklet 4 as well as your implementation of the Swing GUI client into `uk.ac.cam.crsid.fjava.tick4star`. As you may remember from Workbook 2, in order to upgrade the client in-place, you will first need to send the definitions of all the classes inside `NewMessageType` messages. Finally, you should send the definition of another new class which extends `Message` and contains a method with the appropriate run-time annotation to create an instance of your Swing GUI; since you've already sent the definitions of all the required classes, the Java chat client should then display the upgraded interface correctly!

In order to send the definitions of the classes you need to be able to write the definitions into an array of bytes. You might find the following snippet of code helpful:

```
Class clazz = ... //get a reference to the class you want to serialise
String name = clazz.getName();
InputStream is = clazz.getClassLoader().getResourceAsStream(
                 name.replaceAll("\\.", "/") + ".class");
... //Use "is" to copy the definition of "clazz" into an array of bytes
```

When testing your code, please make sure you run the client outside the IntelliJ environment, since otherwise your client is likely to find the correct classes, not because they have been sent correctly over the socket, but because IntelliJ will find them in the filesystem for you!

When sending new class definitions to the client, it turns out to be non-trivial to send anonymous inner classes; the work-around is simple (avoid using anonymous inner classes). Extra credit (a double star!) will be awarded to those students who produce a correct implementation of the server which is able to send anonymous inner classes to a client without modifying the source of `DynamicObjectInputStream`. If you succeed in doing this, please email `ticks1b-admin@cl.cam.ac.uk` once you have passed the unit test for this ticklet to explain what you did. (No email, no double star.)

You may change the implementation of your server as you see fit to support this new functionality, but you must *not* change your implementation of the Java chat client. In other words, your implementation should continue to work with the code you wrote for Ticklet 2 and 2*.

Submission

When you are satisfied you have completed everything, you should commit all outstanding changes and push these to the Chime server. On the Chime server, check that the latest version of your files are in the repository, and once you are happy schedule your code for testing. You can resubmit as many times as you like and there is no penalty for re-submission. If, after waiting one hour, you have not received a final response you should notify `ticks1b-admin@cl.cam.ac.uk` of the problem. You should submit a version of your code which successfully passes the automated checks by the deadline, so don't leave it to the last minute!