

---

## 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 allow chat session data to be shared or *federated* between two or more Java chat servers. Sharing of chat session data between servers may be desirable in order to improve scalability or as a basis for supporting more advanced features such as permitting clients to send local as well as global messages (something you're welcome to explore, but is not necessary for this tick).

Copy your implementation for Ticklet 4 into `uk.ac.cam.crsid.fjava.tick4star`. You should modify your implementation of `ChatServer` so that it creates two `ServerSocket` objects, one which clients connect to in precisely the same way as they did in Workbook 4, and one which other Java chat servers can connect to in order to share serialised copies of all the `Message` objects which are passed through their own instance of `MultiQueue`. As a result, your program must accept two port numbers on the command line, the first represents the client port and the second represents the federation port which is used by other servers. In order to specify how a set of servers should be connected together, when your server starts, it should accept a list of zero or more other servers which it should connect to as arguments on the command line. Consequently if the user provides zero arguments or one argument, your program should terminate after printing:

```
Usage: ChatServer <client port> <fed port> [fedsrv1:port fedsrv2:port ...]
```

If any of the requested federated servers are not available, your server implementation should ignore the federation request from the user and print

```
Warning: Cannot connect to 'fedserverX:port'. Ignoring.
```

If the format of the host or port number is wrong, your implementation should ignore the entry and print a warning; for example if the user enters `example.com:twenty`, your program should output

```
Warning: cannot interpret 'example.com:twenty' as 'fedsrv:port'. Ignoring.
```

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.

## A Simple Example

Once you believe you have completed this exercise, you should generate two jar files to support testing. One, called `chatclient.jar`, should contain the code for Ticklet 2 with the manifest file specifying the main class `uk.ac.cam.crsid.fjava.tick2.ChatClient`. The other, called, `chatserver.jar` should contain your new implementation of the Java chat server and specify the main class as `uk.ac.cam.crsid.fjava.tick4star.ChatServer`.

Open up two terminal windows and execute the following commands, one in each terminal window:

```
java -jar chatserver.jar 1234 5678
```

```
java -jar chatserver.jar 2222 1111 localhost:5678
```

Then start up a further two terminal windows and execute the following commands, one in each terminal:

```
java -jar chatclient.jar localhost 1234
```

```
java -jar chatclient.jar localhost 2222
```

Your new implementation of the Java chat server should mean that the two clients you started above can talk with one another as if they were connected to the same Java chat server.

## Hints 'n' tips

- You might find it helpful to construct a class called `FederatedHandler` which operates in a similar way to `ClientHandler` but can be used to handle both incoming requests for federation from remote servers, as well as make federation requests to other servers.
- Beware of message loops: if two servers A and B are connected, a message written by a client to server A should be copied to server B, but this same message should not then be sent back from server B to server A! (You may find it helpful to recall that it is perfectly permissible to add fields to a serialisable class and keep the same version number—recipients of the class which are not expecting the additional field will simply ignore it.)
- You may find it helpful to test your server in collaboration with another student. If you do so you might want to disable the invocation of the `run` method in any new code sent over the socket in your implementation of the Java chat client so that you are not vulnerable to remote code execution exploits.

## Submission

Please put the source code and byte code of the package `uk.ac.cam.crsid.tick4star` and (your possibly modified version of) `uk.ac.cam.cl.fjava.messages` into a jar file called `crsid-tick4star.jar`. Please email the jar file to `ticks1b-java@cl.cam.ac.uk`. You should receive a response via email within an hour. If you do not, please send an email to `ticks1b-admin@cl.cam.ac.uk`.