

Advanced Aspects of Object-Oriented Programming (SS 2015)

Practice Sheet 11

Date of Issue: 01.07.15
Deadline: 07.07.15
(before the lecture as PDF via E-Mail)

Exercise 1 JCoBox-Chatsystem

We come back to the chatserver of exercise 10.2. This time we want to implement the server using JCoBox. You find the information on how to compile and run JCoBox programs and the required jar files at <https://softtech.cs.uni-kl.de/homepage/en/software/JCoBox/>.

- a) Write a server that handles multiple clients and accepts new clients at any time. All messages send by any client shall immediately be displayed at all clients. Prefix messages with a unique id for each client. Implement the `.bye`-command to close the connection between some client and the server.

- Do not use the suffix `.java` for your files, the `jcobox` compiler will overwrite them.
- In order to handle the network connections, do not use `ServerSocket` or `Streams` directly, use the wrapper classes provided on the webpage. You are free to extend the provided wrappers if you need more methods, but follow the advices in the comments.

- b) Implement the `.history n` command.

Consider the following questions. If needed change your implementation such that these questions are negated.

- Does sending a long history influence the chat for other clients? Can it block the server or cause lags?
- Is it possible that the client, which requested the history, misses messages by others while receiving the history?

- c) Which guarantees does your implementation provide about the order of messages? Compare with your thread-based solution of exercise 10.1.

Exercise 2 ThermoControl System Using Synchronous Messages

- a) Implement the ThermoControl system from chapter 6.2 using synchronous messages. You can use the code provided on the webpage as a starting point.

The class `AirConditioner` includes a thread for simulation of the system. Leave this part of the implementation unchanged and adapt the rest of the class. This way you can try your system after you have implemented the other components as well.

- The `ControlPanel` should be implemented in Swing. It should show the current and the desired temperature. Additionally the GUI should allow to change the desired temperature using an increment and a decrement button.
- The communication between the components should be realized using the class `SynchronousQueue` from the package `java.util.concurrent`.
- The class `ThermoSensor` is complete and can be used as is.

Exercise 3 Distributed Programming with RMI

- a) What is the advantage of RMI over Web Services?
- b) How can RMI programs handle callbacks? Describe a scenario and what client and server would have to do!
- c) Can you receive a reference to a remote object, that is not registered at a naming service? If so, how?