Prof. Dr. A. Poetzsch-Heffter
Dipl.-Inform. Kathrin Geilmann

University of Kaiserslautern
Department of Computer Science
Software Technology Group

Advanced Aspects of Object-Oriented Programming (SS 2011)

Practice Sheet 11

Date of Issue: 28.06.11
Deadline: 5.07.11
(until 10 a.m. as PDF via E-Mail)

Exercise 1 Questions

a) Add one or more questions to the document at `http://bit.ly/lGbs2v`.

Exercise 2 Asynchronous vs synchronous Calls

```
class A {
   C c;
   B b;
   void m1() {
      c.m(5);
      b.m2();
   }
}
```

```
class B {
   C c;
   void m2() {
      c.m(9);
   }
}
```

```
class C {
   void m(int i) {
      System.out.println("i=" + i);
   }
}
```

For this exercise, we assume that standard Java has been extended with asynchronous method calls and the programmer can choose between them.

a) Consider the given classes. What is the output of the following code in case of synchronous method calls and what happens if all calls are asynchronous? Why?

```
C c = new C();
A a = new A();
a.c = c;
B b = new B();
a.b = b;
b.c = c;
a.m();
```

b) Compare asynchronous and synchronous communication between objects in a multi-threaded program context. What are the advantages and disadvantages of each communication technique?

Exercise 3 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 `http://softech.informatik.uni-kl.de/Homepage/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 10.2.

Exercise 4 Distributed Programming with RMI

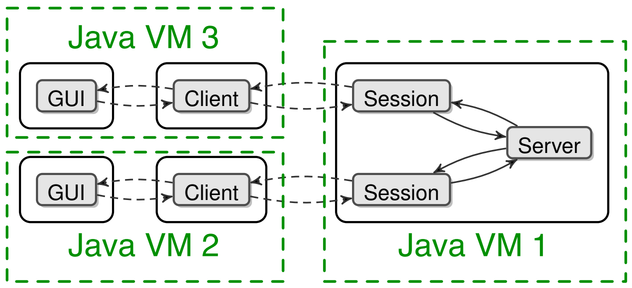
a) Have a look at the Sun RMI tutorial at `http://java.sun.com/docs/books/tutorial/rmi/index.html`

b) Can RMI programs handle callbacks? What is the advantage of RMI over Web Services?

c) Can you receive a reference to a remote object, that is not registered at a naming service? If so, how?

Exercise 5 RMI-JCobox Chatsystem

In the lecture a chatsystem that runs in a single JavaVM was presented. We want to distribute it according to the following picture.



The communication between the different JavaVms shall be done by RMI. On page 101f in `http://softech.cs.uni-kl.de/twiki/pub/Homepage/JanSchaefer/schaeferthesis.pdf` you can see a simple example how to use JCoBox with RMI.

a) Implement the complete system. You can use the GUI implementation, which we provide on the lecture's website. As in the exercises before, the server part of your system shall be able to handle an arbitrary number of clients, which may connect and disconnect at any time, and it shall support the `.bye` and `.history n` commands.