Prof. Dr. A. Poetzsch-Heffter
Mathias Weber, M.Sc.

University of Kaiserslautern
Department of Computer Science
Software Technology Group

Advanced Aspects of Object-Oriented Programming (SS 2013)

Practice Sheet 12

Date of Issue: 02.07.13
Deadline: 09.07.13
(before the lecture as PDF via E-Mail)

Exercise 1 Distributed Programming with RMI

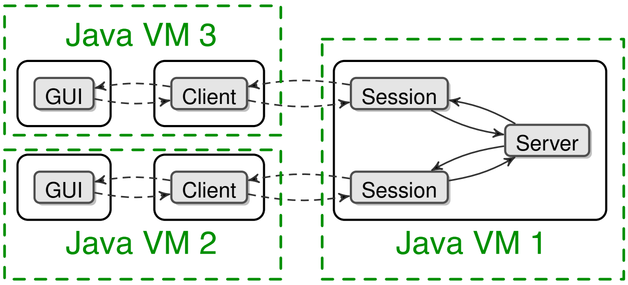
a) Have a look at the Java RMI tutorial at `http://docs.oracle.com/javase/tutorial/rmi/index.html`

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 2 RMI 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 Java VMs shall be done by RMI. The implementation should be done in plain Java instead of JCoBox.

Implement the complete system. You can use the interfaces definitions presented in the JCoBox slides as a starting point. The GUI should consist of an area the chat is shown in and an input field to enter the message to send. 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.

Exercise 3 Swing

As the Swing-Framework is not thread-safe, two rules apply to each swing-application (from the javadoc):

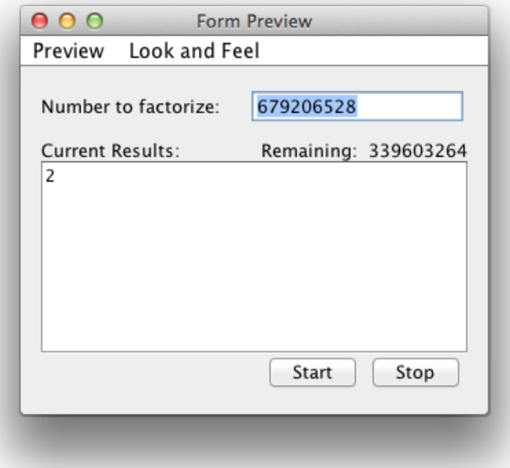
- *Time-consuming tasks should not be run on the Event Dispatch Thread. Otherwise the application becomes unresponsive.*
- *Swing components should be accessed on the Event Dispatch Thread only.*

`SwingWorker` is designed for situations where you need to have a long running task run in a background thread and provide updates to the UI either when done, or while processing.

a) Write a factorization program with the following properties:

- Calculate a prime factorization of the input
- Print the intermediate results into the GUI
- The user should be able to stop the calculation whenever he wants.

The GUI may look like the following:



Number to factorize: 679206528

Current Results: Remaining: 339603264

2

b) Which threads are involved in the execution of a `SwingWorker`? What is the interface between the GUI and the worker?