ITB
Enterprise Computing
Thomas Feilhauer

Exercise Assignment 1

Online tutorials:
Java RMI:
https://docs.oracle.com/javase/tutorial/rmi/index.html
https://docs.oracle.com/javase/7/docs/platform/rmi/spec/rmiTOC.html
https://docs.oracle.com/javase/7/docs/technotes/guides/rmi/index.html

1: Simple Java RMI application

Implement a simple **server** with Java RMI that computes the **square number** of the input value. The server should accept an **integer** as input and return the calculated square number of the input value back to the client.
Write a simple client that can be called from a command line window to test the server.
Develop server and client applications as separate projects!
Start the RMI registry in a separate process from a directory that doesn't contain any Java class-files (also in sub-directories)!
Test your application by **running client and server on different machines** with the **server** being **started** from a **local drive** (not a network drive)!
Figure out, what happens during server startup if the codebase is not set appropriately or if the codebase directory doesn't contain the interface class file. Why is this so?
What happens if the rmiregistry hasn't been started? Why?

After that terminate the rmiregistry and change your server so that the rmiregistry is started from the server program with
`LocateRegistry.createRegistry(Registry.REGISTRY_PORT);`
What does this mean for how the server has to be started? What effects does this have on the codebase?

If you encounter problems with remote access using RMI:
It may happen that the client tries to access an IP address different from the one on which the remote object was exported, see http://www.javacodegeeks.com/2013/11/two-things-to-remember-when-using-java-rmi.html:
To call a method on a remote object the RMI client must first retrieve a remote stub object from the RMI registry. This stub object contains the server address that is later used to connect to the remote object when a remote method should be called (the connection to the RMI registry and the connection to the remote object are two completely different things). By default the server will try to detect his own address and pass it to the stub object. Unfortunately the algorithm that is used to detect the server address doesn't always produce a useful result (depending on the network configuration).
It is possible to override the server address that is passed to the stub object, by setting the system property java.rmi.server.hostname on the RMI server.

- This can either be done in Java code
  - System.setProperty("java.rmi.server.hostname", "<<rmi server ip>>");
- or by adding a Java command line parameter:
  - -Djava.rmi.server.hostname=<<rmi server ip>>