Module 23: Spring RMI

Exercise 23.1 – Spring RMI Calculator

The Setup:

This exercise starts with an example application that demonstrates basic RMI functionality. Once you feel comfortable with how the application works, the exercise will be to create a more complex RMI application.

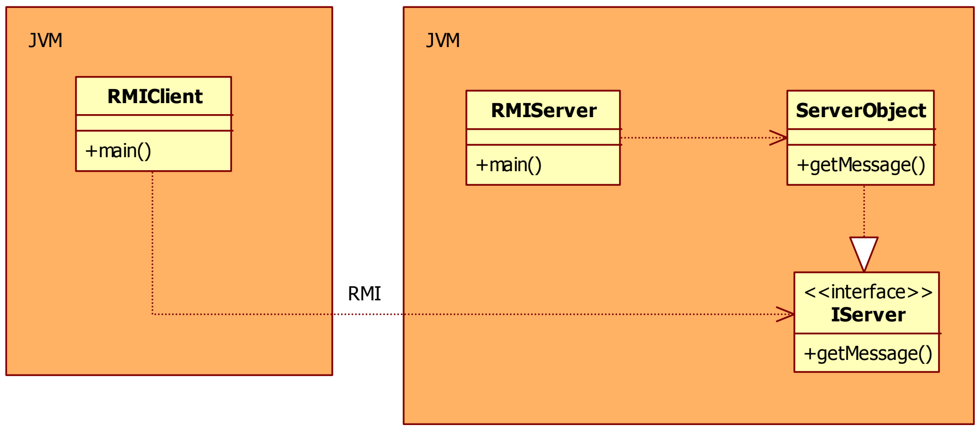
This exercise uses two projects, **exercise23_1-server**, and **exercise23_1-client**. Start by opening **exercise23.1-server**, and add the normal spring context and lgo4j dependencies.

Then, and this is very important, right click on the project and select 'Clean and Build'. Doing so will create a JAR file for the server project which our client project will use.

Next open the client project, and add not just the spring context and log4j dependencies, but also the following exercise23_1-server dependency (only works if you built it in the previous step).

```
<dependency>
    <groupId>cs544</groupId>
    <artifactId>exercise23_1-server</artifactId>
    <version>1.0-SNAPSHOT</version>
</dependency>
```

The Application:



The RMIServer application makes an IServer object (implemented by ServerObject) available over RMI. The RMIClient application (although running in a separate JVM) can then connect to the IServer object and call getMessage(). The getMessage() method takes a Person object as argument and returns the string: "Hello" followed by 'firstName' and 'lastName'.

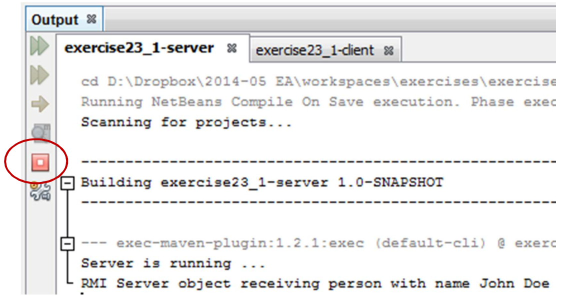
To run the code, first start **RMIServer.java**, which should continue running and output:

```
Server is running ...
```

Then while the server is still running run **RMIClient.java**, which should output:

```
Receiving result: Hello John Doe
```

You can switch back and forth using the tabs in the output area, and once you're done terminate the RMI server with the stop icon shown below:



```
cd D:\Dropbox\2014-05 EA\workspaces\exercises\exercise
Running NetBeans Compile On Save execution. Phase exec
Scanning for projects...

------------------------------------------------------
Building exercise23_1-server 1.0-SNAPSHOT
------------------------------------------------------

--- exec-maven-plugin:1.2.1:exec (default-cli) @ exerc
Server is running ...
RMI Server object receiving person with name John Doe
```

The Exercise:

Look through the code, and once you are comfortable with how the example code works, create a small RMI calculator application. Your RMI calculator server should maintain its state by means of a private int attribute, much like the number on the screen of a calculator.

The server should have one or more methods that modify and then return its state. Since this makes it a stateful RMI server, you will also need to add the synchronized keyword to your method signatures for thread safety.

An example method would be:

```java
public synchronized int calc(char operator, int number)
```

The client application should send calculation commands to the server, printing the command and the result from the server to the console.

Once you have completed your calculator, add stopwatch functionality to the client to output the duration of each remote call. Although you can do this with AOP, it's easier to just create a stopwatch, start the stopwatch before the call, and stop it after the call to check the duration.