THE UNIVERSITY OF SYDNEY

COMP2121 Lab 4

Remote Method Invocation

The goal of this lab is to make an object remotely accessible. The machine where the object is located plays the role of the server. Client machines will be able to invoke methods of this object remotely. The work is divided into four consecutive exercises.

Make sure to run Linux on your computer or reboot it.

Exercise 1: Remote Object Interface

The first task is to write the server interface indicating the methods that can be invoked remotely. This interface should be present both at the client and server sides.

A necessary package is `java.rmi`. You have to implement a `RemoteCounter` interface specifying the behaviour of an object that extends the `Remote` interface. It simply consists of a method `inc` taking no argument and returning an `int`.

Duration: 10 min

Exercise 2: Server Implementation of the Interface

This task consists in writing the `RemoteCounterImpl` implementation of the previous counter interface. It will simply export an `inc` method that takes no argument but returns as an `int` the incremented counter value.

However, as a remote object it should be available remotely. The simplest solution consists in extending the `java.rmi.UnicastRemoteObject` class. (It defines various constructors and static methods to make remote objects available. By construction, such objects will be automatically made available and standard sockets will be used transparently for communication.)

```
...
public class RemoteCounterImpl extends UnicastRemoteObject implements RemoteCounter {
...
```

For simplicity, this implementation will also contain the `main` method that creates the server. It notifies the *name server* of the server-side machine of the existence of the counter object using the static methods of `java.rmi.Naming`. Actually, the name server associates a name, "THE_SERVER_NAME" in the example below, to the remote object. Choose an appropriate name.

```java
public static void main(String args[]) {
    ...
    // Bind this object instance to the name "RMICounterObject"
    Naming.rebind("THE_SERVER_NAME", counterServer);
    ...
}
```

Duration: 15 min

Exercise 3: The RMI Client

First, the client has to contact the name server to obtain the remote stub of the `Remote` object. This is achieved using the static `lookup` method of the class `Naming` and referring to the right object name using a URL of the form: `rmi://host:port/name`.

By default you can use, `rmi://127.0.0.1/THE_SERVER_NAME` to refer to the local machine with default port 1099. Once you create a new instance of the server, simply print the result returned by its `inc` method on the standard output.

Duration: 15 min

Exercise 4: Invoking the Remote Method

First, try to make the RMI client connect to the server on the local machine. To make sure the counter server registers itself to the name server, the name server `rmiregistry` must be started first. This can be done as follows on unix-like machines (under Windows, use `start rmiregistry` or `rmiregistry &` from your classpath folder):

```
    $ remiregistry &
```

Then, executes the counter server in the background before running the client.

After you made the RMI work on the local machine, make sure to kill the `rmiregistry` and the counter server. Copy the files onto a remote machine and update the server IP address to which the client connects to before re-invoking the method from a remote machine.

Duration: 10 min