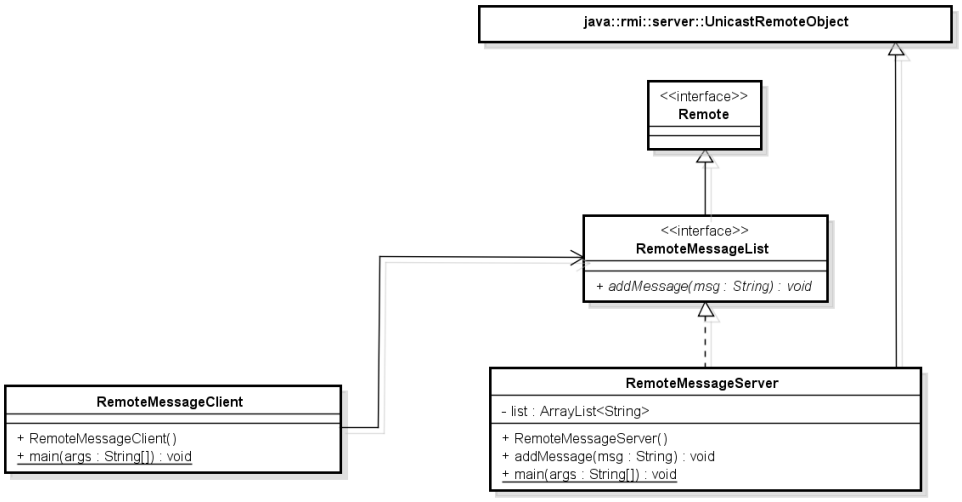
Exercises Session 7 SDJ2, RMI

Exercise 07.01

Implement a simple RMI system where the server contains a list of strings and clients can add strings to this list. Implement it as shown in the UML class diagram below

- In the `addMessage` method (on the server) print out the message being added
- Implement the client to add messages to the server in a loop.

Test it stating up the registry, the server and a few clients

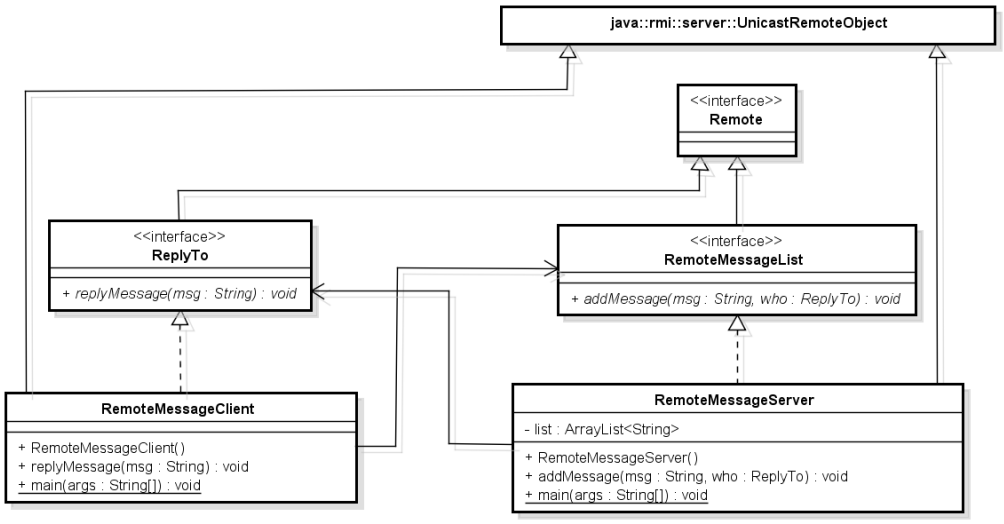


Exercise 07.02

Modify the previous exercise to become an RMI call back system as shown in the class diagram below.

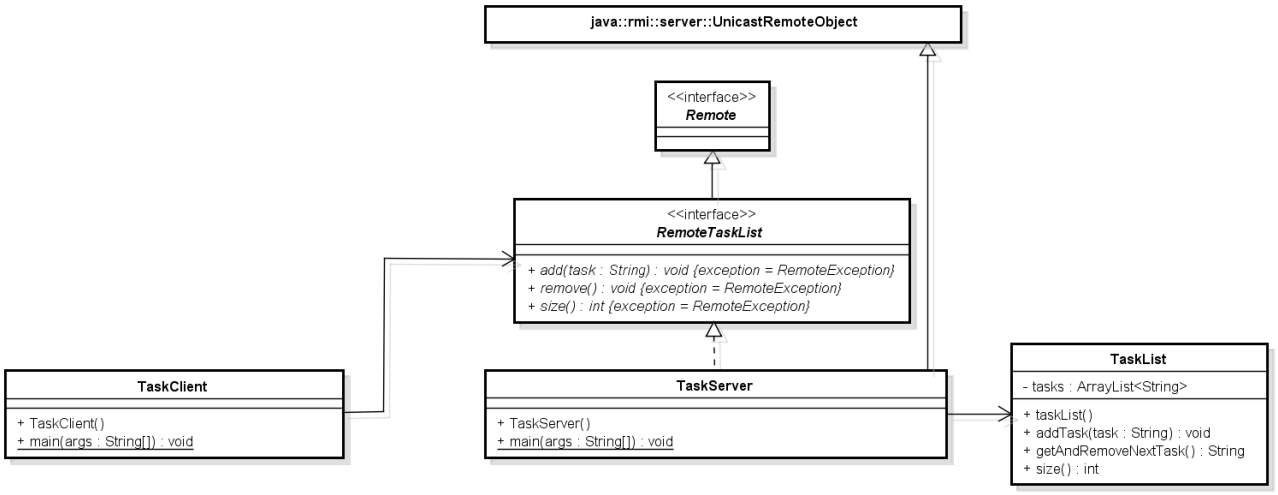
- Method `addMessage` now takes two arguments. In the method add the message to the list, print it out and reply with the message "Thank you"
- Implement the client (`RemoteMessageClient`) to implement an interface `ReplyTo` extending `Remote` and let the class extend `UnicastRemoteObject` (to become a server too). In the method just print out whatever message received.

Note: You only make one lookup in the registry – the server never makes a lookup.



Exercise 07.03

Implement an RMI system where the server has a task list - an ArrayList of Strings (or a Queue from your own implementation) and clients can add a task, get the next (first) task and get the size of the task list. Consider how to avoid concurrency problems.



(Exercise 07.04)

Use exercise 07.01 as basis to design and implement an RMI system similar to the socket "chat" system (exercise 06.01/06.02) sending `Message` objects (NOT converted to Json strings). The client could add a message to the server and the server simply prints it out (just like exercise 06.01). What if class `Message` do not implement interface `Serializable`?

(Exercise 07.05 – group work: Design an RMI call back system)

Use the previous exercise as basis to design (draw a UML class diagram in Astah) of an RMI callback system. Every time a client is adding a task then every connected clients will get a message that a task has been added.

(Exercise 07.06)

Implement and test the system you designed in the previous exercise.