# Practical 2 (assessed)

**Theme**

- testing, deploying and using a given peer-to-peer DS written using the Mantaray JMS implementation
- understanding parts of the Java JMS code, especially the routing subsystem
- making use of concurrency
- completing an addition of acknowledgements for routed messages

**Key concepts:** routing, asynchronous and synchronous messaging, JMS

## 2.1. Start up and essential configuration

a) *Log in to **Windows XP**.*

b) *Prepare to open BB quiz Practical 2 when required.*

This is an **assessed practical** that contributes to your module score.

As you work through the tasks described in this sheet, you will be occasionally asked to answer a quiz question related to the task you just completed.

It is recommended that you start the quiz within the **first 20 minutes** of the practical.

c) *Download file `lab2-chat-p2p.zip` from BB and extract it somewhere.*

For example, you can create a folder called `CS3250-DS-1011-WIN` in your H drive and extract the archive there. Beware that the default extractor creates two folders named `lab2-chat-p2p` one inside the other.

d) *Open the extracted files as an Eclipse Java project.*

Eclipse can be found in the Start menu in the CS sub-menu. Use Eclipse 3.6.

Switch to the Java perspective and create a new Java project, specifying the correct folder `lab2-chat-p2p` (ie the one that has a subfolder called `src`).

The project does not compile at this stage because it does not have access to the Mantaray JMS library.

e) *Include the Mantaray library in the project build path.*

Right-click on the project in the Project Explorer and select Build Path > Configure Build Path. In the dialogue, select the Libraries pane, within it click Add Library, select User Library and click Next. Then click on the User Libraries button to open another panel. Add a library and call it Mantaray. Then click on Add JARs and browse to the file `C:\mantaray\manta.jar` in order to tell Eclipse that it forms part of the library. Similarly add all files in the folder `C:\mantaray\ext`. (Add them all in one step by selecting them all using `Ctrl-A`.) When you are finished adding the JARs, click OK to close the panel and select the Mantaray library for inclusion in the project.

Once all the panels are closed, the project should compile without errors.

## 2.2. Testing the given simple peer-to-peer messaging system

a) *Start three separate peers (A, B, C) on your own computer.*

Locate the class `WisdomsPeerGUI` in the `wisdoms.peer.gui` package, open it and run it as a Java application.

The first time the GUI opens, close it because it would fail to operate correctly. After closing the GUI, open Run Configurations and edit the WisdomsPeerGUI configuration. Switch to the Arguments tab and paste the following into the VM arguments area:

```
-DmantaConfig=C:\mantaray\config\default_config.xml
```

This time clicking the run icon should bring up a fully working GUI for the peer. Click it altogether three times to launch three independent peers.

Give each peer a unique name A, B and C and click their Start buttons.

You may need to answer Unblock to a Windows firewall warning popup one or more times at this stage.

b) *Enter a wisdom to peer A.*

Enter some wise text into the peer's wisdom entry field and press Add.

c) *Check for new wisdoms at all peers.*

Since the peers have not been connected yet, the wisdom stays on peer A only.

d) *Connect A with B and B with C.*

Enter the name of peer B, including the hostname, so that it looks something like `B@eas-372pc99`, into peer A and press Add. Analogously with peer C.

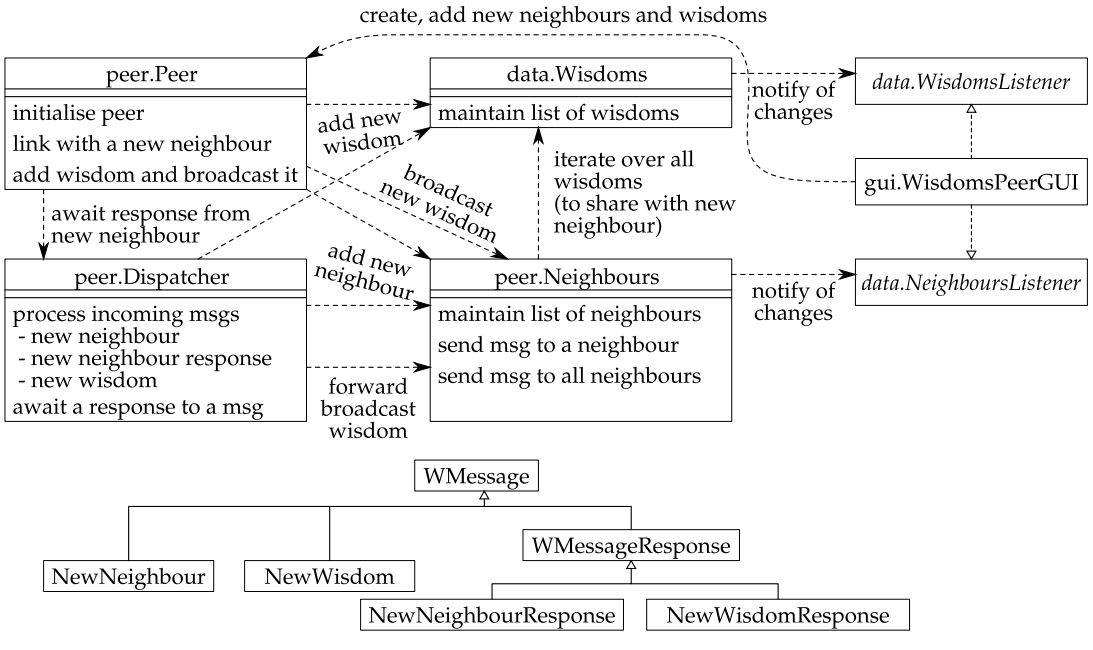
e) *Check what neighbours and wisdoms peer B knows about.*

f) *Repeat exercises (b) and (c).*

✎ **Quiz.** Answer and save **Question 1**, which is related to this exercise.

## 2.3. Inspecting the system's source code

a) *Consider the design of the main peer classes*



b) *The peer is programmed to respond to each* `NewNeighbour` *message it receives. Locate the line in the given files that contains a command that is* specifically *responsible for sending these responses.*

✎ **Quiz.** Answer and save **Question 2**, which is related to this exercise.

c) *Locate the method in the given files in which one of the peer's executing threads blocks waiting for the response.*

✎ **Quiz.** Answer and save **Question 3**, which is related to this exercise.

d) *Locate the* ***line*** *in the given files on which there is a command whose execution causes the thread that is waiting for the response to stop waiting.*

✎ **Quiz.** Answer and save **Question 4**, which is related to this exercise.

### 2.4. Adding a signature in front of forwarded wisdoms.

a) *Modify the program so that each wisdom arriving from another peer is forwarded with its text a little modified as follows: A short signature text should be added to the start of the wisdom text.*

b) *Test your code with one or more fellow students*

✎ **Quiz.** Answer and save **Question 5**, which is related to this exercise.

### 2.5. Sending wisdoms synchronously

a) *Change the peer code so that whenever a* `NewWisdom` *message is received, an appropriate response is sent back to the sender.*

b) *Adjust the code that sends a* `NewWisdom` *message to a neighbour so that it waits for a response and if it does not arrive within 2s, it removes the neighbour from the neighbours set.*

> **Hint.** You will most likely need access to the main instance of `Dispatcher` but this will not be available via any fields. To solve this problem, adjust the method that is called by the one who initiates the wisdom sending — add a new parameter of type `Dispatcher` through which a reference to the dispatcher object will be made available.

c) *Test the new code by executing two peers, connecting them, then shutting down one of them. When sending a new wisdom, the former neighbour should disappear from the list after 2s.*

✎ **Quiz.** Answer and save **Questions 6 and 7**, which are related to this exercise.

d) *Change the code further so that waiting for a response is performed by a new thread, one thread for each response.*

✎ **Quiz.** Answer and save **Question 8** and **optional Question 9**, which are related to this exercise.