Main Coursework Instructions (COMP2207)

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Module:	Distributed Systems and Networks	**Mark Contribution:** 30% of overall module mark
Assignment:	A Distributed Notification Framework using Java RMI	**Effort:** 45 hours
Submission deadline:	15th December 2016, 16:00 (in C-BASS) (Feedback by 13th January 2017)	**Lecturers:** cc2, agw106

About the coursework

The purpose of this assignment is to design and implement a Distributed Notification Framework using Java RMI. You must demonstrate your understanding of Distributed Objects and of Java RMI in particular. Therefore, <u>solutions using other distributed technologies will not be acceptable.</u>

In a distributed system, applications communicate and coordinate with each other by message passing. One approach to integrating distributed applications is using a notification model. A simple distributed notification model is composed of *Notification Sources* and *Notification Sinks (*sources and sinks for simplicity). A sink registers as wishing to receive notifications. A source will notify a registered sink by sending notification messages when a particular event happens. A source can send notifications to any number of sinks, and a sink can receive notifications from any number of sources.

A Distributed Notification Framework can be based on the following classes: **NotificationSource**, **NotificationSink** and **Notification**, whose behaviour is described below:

- A **NotificationSource** is an object that will notify registered **NotificationSink** objects when a particular event happens. The idea is that **NotificationSource** objects are used on remote machines to monitor the behaviour of certain components of a distributed system and to send a Notification on given events.

- A **NotificationSink** object is registered with a **NotificationSource** and receives **Notification** objects every time the monitored event happens.
- A **Notification** object is used to hold information about an event when it happens.

A registry may need to be maintained by a **NotificationSource**. A simple registry could be a Java collection.

A simple example of a **NotificationSource** application could be a specialised clock that promises to notify registered users whenever a certain period of time has elapsed. To use it, other applications create a **NotificationSink** object and register that with the **NotificationSource**. Now whenever the special time event occurs, all registered **NotificationSinks** are informed; in this example, they are sent a Notification with the time information in it.

You are required to implement the above classes, and build an application showing that they work as specified. You must show that your sink application can receive notifications from more than one source, and that your source application can send a **Notification** to more than one sink. You should also devise a scheme so that, if a **NotificationSource** lost connection with a **NotificationSink**, this would not cause a long-term problem.

Your application should run on a single machine, and you must indicate the changes needed to your Java code in order to run the application on several machines.

Submission

Report

You are required to produce a report that describes the design, implementation and testing of your framework and application. The report should focus on your use of the Distributed Objects paradigm and of Java RMI, and should:

1. justify the choices made in the design and implementation of your framework,
2. explain how your framework is used by the application,
3. explain how your framework was tested using the application.

Please note that only the notification aspects of the application will be marked, and your report should focus on these aspects.

The report must include sections describing:

The Notification framework: This section should include an explanation of how the classes for the Notification framework have been designed and implemented. You will demonstrate your understanding of Java RMI succinctly, through a justification of your design/implementation decisions. You are not required to explain the Notification model; instead the focus must be on your use of distributed objects to implement this model.

The application: This section describes your application, with a focus on how it uses the notification model. Relevant implementation details (with regards to the use of classes from the Notification framework) should be given here. You are also required to explain how the application was used to test the notification framework.

Multiple sources and sinks: This section explains and evidences the use of multiple sources and sinks by your application.

Lost connections: This section describes how lost connections are handled by your framework and/or application.

Future work: This section provides an explanation of how the application can be implemented on several machines, and how the framework and application can be improved.

Conclusions: Based on all of the above, this section presents a critical evaluation of the suitability of distributed objects for the implementation of the Notification framework.

The (all inclusive) word count for the report should range between 1350 to 1500 words, in any case not exceeding a total of 6 pages including any figures such as diagrams and screenshots. You must reference in footnotes any resources used. Please note that the University regulations regarding Academic Integrity apply (http://www.calendar.soton.ac.uk/sectionIV/academic-integrity-regs.html).

Additional files

In addition to the report (in PDF format), your submission should include:

- a single TXT file containing a well-documented source code for your Notification framework,
- a ZIP file containing the Java sources of your entire implementation.

Please ensure that your Java sources can be compiled and executed on a lab machine, and that each of the submitted files are of the required extension (PDF, TXT and ZIP respectively).

These three files are to be submitted via the C-BASS handin system (http://handin.ecs.soton.ac.uk), by the submission deadline stated above. Do not submit hard copies. The standard ECS late penalties apply (as per para. 4.1 of the regulations: http://www.calendar.soton.ac.uk/sectionXII/ecs-ug.html): 10% per working day that a piece of work is overdue, up to a maximum of 5 days, after which the mark becomes zero.

Learning Outcomes

Knowledge and Understanding

Having successfully completed this work, you will be able to demonstrate knowledge and understanding of:

A6. Distributed objects, including use through Java RMI

Intellectual Skills

Having successfully completed this work, you will be able to:

B1. Explain the fundamental concepts underlying networks and distributed systems

B4. Choose between alternative paradigms and technologies for solving problems in distributed systems

Practical Skills

Having successfully completed this work, you will be able to:

C1. Build a client-server solution in Java

C2. Build a distributed-objects solution in Java

All the while demonstrating ethical and professional values (i.e. exercising academic integrity and engaging in good software development practices).

Marking Scheme

Your submission will be marked out of 100. The marking criteria below will be used, with descriptors for work at various levels of demonstrated competency:

Criterion	**Descriptors per level**	**L.O.**	**Total**
Framework	Excellent: The framework is described clearly and concisely. The design decisions are well justified, with correct use of remote and serializable interfaces. Good: A clear and concise description of the framework, and a largely correct use of remote and serializable interfaces. Design decisions are justified. Minimal: A working framework is described, though it may lack simplicity or clarity of design decisions.	*A6* *B1*	*40 marks*
Application	Excellent: The application design is clear and succinctly explained (e.g. includes system and class diagrams). The application uses the framework as required, and this is clearly explained. An explanation of how the application has been used to test the framework (e.g. test cases and test reports) is included. Code is well structured and well documented. The application is scalable. A brief step-by-step guide on how to run the application is included. Good: The application design is clear and concise, and the framework is used as expected. Some explanation of how the notification framework is used, and some evidence of testing are included. Minimal: A simple application, which may lack sufficient explanation of its design and its use of the notification framework. Evidence of testing may be limited or inexistent.	*C1* *C2*	*30 marks*
Multiple sources and sinks	Excellent: Clear explanation and evidence of multiple sources and sinks, and how they are distributed. Good: Clear evidence of multiple sources and sinks. Minimal: Limited evidence of multiple sources and sinks.	*A6*	*10 marks*

Criterion	**Descriptors per level**	**L.O.**	**Total**
Handling of lost connections	Excellent: The application can handle lost connections gracefully. Reconnection is attempted. Messages are queued and redelivered. Good: Application can handle lost connections. Reconnection is attempted. Minimal: Application can detect lost connections.	*B1, B4, C1, C2*	*10 marks*
Future work	Excellent: Step-by-step guide to show how the application can run on multiple machines, and clear and succinct explanations of how the framework and application can be improved. Good: Some explanation of how the system can run on multiple machines or of any potential improvements of the framework and/or application. Minimal: Minimum explanation of how the system can run on multiple machines.	*B1, B4*	*5 marks*
Conclusions	Excellent: Critical evaluation of the suitability of distributed objects for the framework and application, including limitations and how these can be overcome. Good: Critical evaluation of the suitability of distributed objects for the framework and application. Minimal: Some evaluation of the use of distributed objects for the framework and application.	*B1, B4*	*5 marks*