University of Glasgow | School of Computing Science

Assessed Coursework

Course Name	**CBSE exercise 3**			
Coursework Number				
Deadline	**Time:**	**4.30pm**	**Date:**	**30th January 2015.**
% Contribution to final course mark	**5%**		**This should take this many hours:**	**2hrs/week**
Solo or Group ✓	**Solo**		**Group**	**X**
Submission Instructions	**Submission is via Moodle**			
Marking Criteria	**Task based**			
Please Note: This Coursework cannot be Re-Done				

Code of Assessment Rules for Coursework Submission

Deadlines for the submission of coursework which is to be formally assessed will be published in course documentation, and work which is submitted later than the deadline will be subject to penalty as set out below. The primary grade and secondary band awarded for coursework which is submitted after the published deadline will be calculated as follows:

(i) in respect of work submitted not more than five working days after the deadline
   a. the work will be assessed in the usual way;
   b. the primary grade and secondary band so determined will then be reduced by two secondary bands for each working day (or part of a working day) the work was submitted late.

(ii) work submitted more than five working days after the deadline will be awarded Grade H.

Penalties for late submission of coursework will not be imposed if good cause is established for the late submission. You should submit documents supporting good cause via MyCampus.

Penalty for non-adherence to Submission Instructions is 2 bands

You must complete an "Own Work" form via

https://webapps.dcs.gla.ac.uk/ETHICS for all coursework

UNLESS submitted via Moodle

CBSE 2014/2015

Exercise 1

The objectives of this exercise are:

1. To acquaint you with implementing communication protocols for component based systems.
2. To learn how to build components that communicate using TCP sockets.

Preamble:
`mCom` is a lightweight component (a set of objects) for sending and receiving text messages over a network. It is lightweight because of its network agnostic properties, and its ability to assume the role of a server (by receiving messages) and/or a client (by sending message) at the same time. Currently, a `mCom` instance can send a message to only one other instance, and receive from multiple instances at any particular moment.

The java class files and dependencies for compiling and running `mCom` is contained in the bundle `mCom.zip`. (available for download from Moodle). Description of core files in `mCom.zip` is as shown in Table 1.

Table 1: description of mCom.zip content

`SenderImpl.java`	- Implements `Sender` interface. - Responsible for making connection request and sending messages to a connected recipient.
`ReceiverImpl.java`	- Implements `Receiver` interface. - Starts a background thread that listens to incoming messages and responds to connection request.
`Initialiser.java`	Contains the main method and executes the following: (1) Initialise local host address (2) Start a Receiver to listen to incoming messages (3) Creates a display for console interaction
`Display.java`	UI console
`IPResolver.java`	Utility class for IP and port configuration

You are to carry out the following tasks:

Task 1(1 mark):

(1) Configuration and deployment **(1/2):**
   a. Create and configure an eclipse project for `mCom` by using the files and dependency lib in `mCom.zip`.
   b. Build the project and generate a jar file. This should be done by exporting project as a runnable jar named `mCom.jar`

(2) Demonstrate interaction between instances of `mCom` by running `mCom.jar` on multiple consoles (where each console represents an instance). Recommend 3-4 consoles, each running on a single or multiple computers (for convenience demonstration with course tutor should be done on a single computer in the lab). **(1/2)**

Task 2 (3 marks):

Assuming the set $M = \{m_1, m_2, m_3, \ldots m_n\}$ are instances of `mCom`.

(3) Add a new class `RegistryImpl.java` that implements the `Registry` interface. This class should act as a broker with the following two features**:**

a. The `register` and `deregister methods` enables m_i to act as a Registrar. A Registrar accepts request from `mCom` instances to advertise or unadvertised their availability to receive text messages, and keeps record of their IP address and listening port. **(1/2)**

b. The `lookup` method enables an instance m_j to lookup instances that are available to receive messages from m_i. **(1/2)**

(4) Carry out the following refactoring activities**:**

a. Refactor `ReceiverImpl.java` and/or `Display.java` to enable an instance m_i at startup to also run as the Registrar for m_j that is available to receive text message. It should be possible for m_i to register more than one instance. Furthermore, m_i should inform all its registered instances when it ceases to act as a Registrar **(1/2)**.

b. Refactor `ReceiverImpl.java` and/or `Display.java` to enable a user initialize or update m_j with the IP address and port of a set of instances $M' \subset M$, where M' is a set of available registrars. **(1/2)**.

c. Refactor `ReceiverImpl.java` so that m_j can advertises its availability with M' at start up. Furthermore, m_j should be updated accordingly when $m_k \in M'$ ceases to act as a Registrar. Similarly at shutdown, m_j should request for deregistration from M' **(1/2)**

d. Refactor `SenderImpl.java` by adding the ability for m_j to lookup available instances from M'. This function should enable m_j to lookup and send text message to all instances that have registered their availability to receive messages with M'. **(1/2)**

Task 3(1 mark):

(5) Demonstrate the following:

a. A `mCom` instance running as a registrar and satisfying advertisement and lookup request from other instances. **(1/2)**

b. A `mCom` instance sending text message to a set of available instances. **(1/2)**

Deliverables:

1. Group demonstration of each tasks to the course tutor. This will be done during scheduled lab session.
2. Submit your refactored code and jar as a compressed zip file on Moodle.

Datelines:

- Exercise will be assessed during labs sessions of week 18 and 19. That is: **Friday 23th January 2015** and **Friday 30th January 2015**.

Process:

This is a group exercise and every member of the group is expected to take an active role. At the end of the exercise, every individual in a group will submit an objective peer review of effort and contributions of group member to the tutor. Submission shall be made via Moodle. For the review, you will benchmark each member of the team on a Likert scale of 1-5. Where 1 means no contribution and 5 is excellent contribution.