University of Glasgow | School of Computing Science

Assessed Coursework

Course Name	**CBSE exercise 2**			
Coursework Number				
Deadline	**Time:**	**4.30pm**	**Date:**	**20th Feb 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 2

This exercise builds on your experience of exercise 1, and is based on a new version of mCom (version1.0.1). This version is streamlined specifically for manual execution of remote invocations. It can also be used as the basis for your exercise 3.

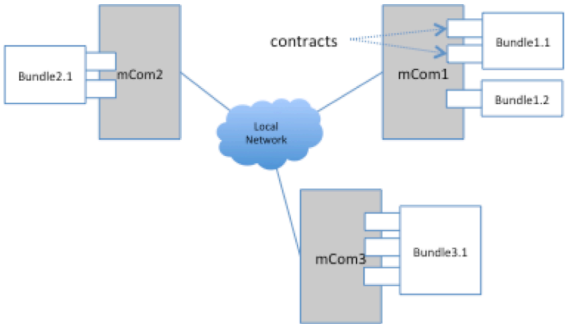

Figure 1. Structural diagram of mCom component platform

A structural representation of a mCom instance is as shown in Figure 1.

The following terms are used in mCom component model:

1. **Bundle** – A bundle is a unit of deployment on mCom platform.
2. **Contract** – A contract is an agreed service that a bundle provides to other bundles in the network. A bundle can consist of one or more contracts.
3. **Bundle descriptor** - A bundle deployment is described by a bundle descriptor and stored as .xml file in the local directory of a mCom instance. An xml tree diagram of a bundle descriptor is as shown in Figure 2.

xml	version="1.0" encoding="UTF-8" sta…
▼ BundleDescriptor	
BundleName	gbpBundle.jar
BundleId	37
HostAddress	192.168.1.74
HostPort	64551
BundleController	gbp.converter.PoundBundle
BundleControllerInit	convertPound
▼ Contracts	
▶ Contract	
▼ Contract	
BundleEntity	gbp.converter.PoundBundle
BundleEntityContract	convertPound
ContractType	GET
Description	Supported Currency: UAH,AUD,AZM…
▼ Parameters	
▼ Parameter	
Name	java.lang.String
Value	null
▶ Parameter	
▶ Parameter	
ReturnType	java.lang.String

Figure 2. xml tree diagram of a bundle descriptor

A set of mCom annotations stored in `mbundle-annotations.jar` is used to specify the behavior of a bundle, and also influence the automatic generation of bundle descriptors during deployment. There are four types of mcom annotations:

1. **@mController:** This is a ElementType.TYPE annotation that is used to identify the class that controls the bundle. A bundle has can have at most one @mController annotation.
2. **@mControllerInit:** This is an a ElementType.METHOD annotation used to identify a method that initializes parameters and calls methods required for the bundle to run appropriately during invocation. A @mControllerInit can only exist within a @mController class.
3. **@mEntity:** This is a ElementType.TYPE annotation that is used to identify the class that contains a bundle contract.
4. **@mEntityContract:** This is a ElementType.METHOD annotation that is used to identify contracts specified in a @mEntity. This annotation takes the description of the contract and contractType as parameters. The current version of contract specifies two main types of contracts: GET or POST (Although they are not directly used in this exercise, you may choose to leverage on contract types for session management for your exercise 3. You can also define new ones).

Annotation code can be found in the package `mcom.bundle.annotations`. The file `gbpBundle.jar` is an example illustrating the usage of mCom annotations

Each time a mCom instance is executed using the command-line, the following actions are automatically carried out:

- The instance automatically creates a local bundle directory /LocalBundleDir where all bundles that can be deployed and advertised by that instance is kept.
- The instance automatically creates a local bundle description directory /LocalBundleDescDir. This the directory where BundleDescriptors that describe deployed bundles are kept.

Once mCom is running, the following command features can be executed:

Command	Action
isReg%<bool>	Enables the instance to switch on/off registry service.
drs	The instance executes Dynamic Registrar Discovery. This involves the automatic discovery of instances on the platform (including itself) that also assume the role of a Registrar.
reg	show a list of all available Registrars that the instance can advertise or lookup contracts.
deploy	deploys all bundles in LocalBundleDir directory. For each deployed Bundle, a BundleDescriptor is generated. The descriptor is stored as .xml file in /LocalBundleDescDir. Only deployed bundles can be advertised, looked up and invocked.
llookup	The instance executes a local lookup all bundle contracts in LocalBundleDir. This provides information on each bundle and the associated contracts that it provides.
rlookup	The instance executes a remote lookup of all advertised contracts with known Registrars
adv%<bundleId>	The instance advertises all the contracts in a bundle descriptor referenced by bundleId. The advert is sent to all known Registers.
sadv	Shows all the adverts that has been made on the instance. This command is only active if the instance also assumes the rolw of a Registrar.
invoke	A remote invocation of a specified bundle contract. This function requires bundleId, contract name and contract parameters as input. (This command is to be completed as part of task 1)

The java class files and dependencies for compiling and running `mCom`, `mbundle-annotation` and `gbpBundle` is contained in the bundle `mComCompendium.zip` (available for download from Moodle). The shell script `mcom.sh` is also included. This script is to enable you start mcom.jar from command line, you can amend to suit your environment.

Task 1 (2.5 marks):
Using the component communication setup in mCom and the reflective capabilities of java, complete the invoke command.

This task should be achieved by completing the code for `executeRemoteCall` method in `mcom.kernel.impl.RemoteMComInvocation` class. You can also refactor or extend other methods or classes as you see necessary. The following are guidelines on how to carry out this task:

1. Parse the result from rlookup command to instantiate the address, port of a remote mCom instance, as well as a deployed contract.
2. Execute a request-response communication protocol with the remote object. The request should encode parsed variables as text, and the response should contain the result of invoked contract.

Hint: Some code required to execute this task is already implemented in other commands. You may find it easier to understand and reuse the code rather than writing completely new ones.

Task 2 (2.5 marks):
Study the code and usage of mCom annotation in gbpBundle. Based on that, convert the code in the folder wbundle to a deployable mCom bundle.

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 21 and 22. That is: **Friday 13th Feb 2015** and **Friday 20th Feb 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.