TEESSIDE UNIVERSITY
SCHOOL OF COMPUTING

ICA SPECIFICATION

Module Title: Advanced Java Programming	**Module Leader:** Simon Lynch
	Module Code: COM2053-N
Assignment Title: AJP – Part II Middleware	**Deadline Date: Jan 2017 – TBC** vivas will normally be held within 10 working days of submission
	Deadline Time: TBC see module calendar for phase presentations & viva
	Submission Method: Electronic Submission

ONLINE Submission Notes:

- Please follow carefully the instructions given on the Assignment Specification
- When an extension has been granted, a fully completed and signed Extension form must be submitted as soon as it had been signed.

FULL DETAILS OF THE ASSIGNMENT ARE ATTACHED
INCLUDING MARKING & GRADING CRITERIA

S.C.Lynch@tees.ac.uk

AJP Assessment – middleware

brief

The aim of this work is to build a thin middleware layer in Java based on the concept of actors/meta-agents.

learning objectives

1. Demonstrate an understanding of concurrency and code synchronisation.
2. Design, implement and evaluate a simple agent based application.
3. Work effectively as part of an agile development team.

assessment groups

For this assessment you will be organised into development teams, each with 3 or 4 members. The style of working your team uses is for the team to choose but, where possible, we expect you operate as an agile team rather than a collection of individuals.

Work submitted by the team will be marked as a team effort. Marks for individual team members will be calculated from the team mark and the XP-points awarded by the team to each individual member. See the module web-page for details about the XP-points system used on this module and details of the approach used to calculate individual marks (for further discussions/advice about agile programming methods and extreme programming please talk to your module tutors).

phases

Work will be submitted and feedback given in the following stages (all work will be marked for its technical content)…

- pitch: discuss your design, message flow details, the main classes and contracts;
- code review: present your Java code for review & feedback;
- final submission: groups will be required to submit Java code (see later notes) and attend a 20-30 minute viva. Testing strategies will be discussed as part of the viva.

S.C.Lynch@tees.ac.uk

detail

The primary reference materials for this assignment may be drawn from articles (provided) which describe (i) meta-agents (ii) meta-actors (iii) the design brief of Boris. These materials give a useful background but they are theoretical in nature and focus on meta-agents/actors so do not always illustrate the most appropriate designs or design patterns. You should, therefore, use the reference materials as a background for your work but not use them as design documents.

Your aim is to build a thin middleware layer in Java based on the concepts described in the reference materials. This middleware will be constructed from meta-agents which you will specify and build in the early stages of this work. You will then use these meta-agents to build portals which will manage agent to agent communication. Meta-agents (and their extensions/wrappers) should operate concurrently using synchronisation where appropriate to the design of your system.

You are expected to make good use of design patterns as appropriate.

objectives

- the middleware will support mixed networks of portals and user defined agents (also allowing for future extensibility);
- networks will be structured as interlaced trees with uni- and bi-directional links;
- where appropriate the middleware will allow nodes to be scoped (up to and below named nodes in their hierarchies;
- the software will include hooks for node monitors and at least one implementation of NodeMonitor;
- the software will support (at least) two message types (i) UsrMsg to encapsulate messages between user agents (ii) SystemMsg to wrap messages between meta-agents detailing agent registration requests and other admin functions;
- all aspects of software design/construction will (i) make best use of appropriate design patterns (ii) support scalability and (iii) use designs which support future extensibility.

Note also... part of this work is to understand the difficulties, trade-offs and compromises inherent in building dynamic middleware. You will be asked about these during the viva.

build stages

The following table outlines a recommended order for constructing the required software components for the assessment. The points column are to provide a rough guide to the effort/time required.

Note:

1. before you start this code construction you should draw your high-level design, identify message flows, consider the application of design patterns and identify any concurrency issues;
2. the build stages suggest testing episodes, you should carry out unit tests, etc. in addition to these;
3. you should not consider code to be finished until it is Javadoc'd.

S.C.Lynch@tees.ac.uk

	tasks	points
	infrastructure	
1	build a simple meta-agent based on a blocking queue	3
2	design structures & classes for internal message passing	2
3	build a portal (some white/yellow-page device)	3
3	build a "user agent" wrapper for a meta-agent	1
4	TEST – multiple user agents connected to a single portal exchanging UsrMsg's	1
5	add functionality for uni-directional & bi-directional portal-portal links (to allow message routing)	1-2
6	TEST – portal-portal link	1
7	implement robust agent registration	3
8	TEST – agent registration across multiple, pre-established multi-portal networks based on single hierarchies	1
9	TEST – agent registration across multiple, post-established multi-portal networks based on single hierarchies	1
10	TEST – agent registration across interlaced hierarchies	2
	scoping	
11	refactor agent registration to allow scoping	2-3
12	TEST – modify & repeat earlier tests to validate agent scoping	2
	monitors	
13	build the NodeMonitor infrastructure (as required: interfaces, observer classes, adapter) and default NodeMonitor implementation	3+
15	TEST – NodeMonitor infrastructure & default NodeMonitor	2

submission

The following table identifies the elements which should be presented and/or submitted for this phase of the assessment – see also the notes that follow. The mode indicates whether deliverables are to be documentation or code based or to be defended during viva/presentation.

	deliverables	mode	notes	marks
1	pitch	pres		10
2	code review	pres		20
3	submission & viva (see below)			70
3a	a guide to the material submitted	doc	1	-
3b	infrastructure code	code	3,4	20
3c	infrastructure testing	viva	7	10
3d	scoping code	code	3, 4	5
3e	scoping tests	viva	7	5
3f	monitor code	code	3, 4	10
3g	monitor testing	viva	7	5
10	Javadoc	code		5
11	conceptual diagram of meta-agent system structure	doc	2	10
12	feedback reviews & actions	doc		20
13	summary of design & critique	viva	6	20

S.C.Lynch@tees.ac.uk

notes

documentation

1. a guide which should (briefly & accurately) identify all that has been submitted & where it is located; how programs can be loaded & run, etc.

2. the conceptual diagram should show the structure of the system and how its various components are underpinned by a meta-agent substrate. As part of this you may also choose to provide some object diagram (UML, etc) but a UML diagram is not a conceptual diagram. UML diagrams will only gain marks if they enhance conceptual diagrams.

code

3. code should be consistent, well formed, properly (& usefully) commented, ie: code should...
   - clearly & efficiently demonstrate the chosen design;
   - follow a consistent style in terms of indentation, use of braces, variable naming etc.;
   - use variables which are scoped correctly and declared with appropriate access modifiers;
   - include Javadoc comments which are meaningful and complete at the field, method, constructor, class and package level.

4. if code and or comments are difficult to read/understand they will not be marked

~~5. void (this is a legacy place-holder)~~

presentation

6. a concise summary of the work (achievements, limitations, compromises, etc) including a summary of key design features & patterns (eg: when & why have recognised patterns been used; when & why have they been avoided; what would you improve if you rebuilt the work, etc)
7. test harnesses and results – briefly but conclusively presented

S.C.Lynch@tees.ac.uk