Module Title **Concurrent Programming** **Keywords** Threads, Synchronisation, Semaphores, Monitors	Reference CM3033 SCQF Level SCQF 9 SCQF Points 15 ECTS Points 7.5 Created May 2002 Approved April 2005 Amended September 2012 Version No. 5

This Version is No Longer Current

The latest version of this module is available here

Prerequisites for Module

CM2015 Object Oriented Software Design or equivalent.

Corequisite Modules

None.

Precluded Modules

None.

Aims of Module

To provide the student with the ability to evaluate concurrent programming and to design appropriate solutions within a concurrent programme environment.

Learning Outcomes for Module

On completion of this module, students are expected to be able

Indicative Student Workload

Contact Hours	Full Time
Assessment	10
Laboratories	12
Lectures	20
Tutorials	12
Directed Study	
Coursework Preperation	20
Directed Reading	24
Private Study	
Private Study	52

Mode of Delivery

Key concepts are introduced and illustrated through the medium of lectures. However the main emphasis of the course is focused on the laboratory sessions in which the student will progress through a series of graded exercises which are intended to test the student's

to:

1. Analyse user requirements and develop a concurrent solution as either a single multi-threaded application or a collection of co-operating distributed applications.
2. Identify concurrent interactions within the overall design and select (an appropriate combination of) synchronisation mechanisms to handle these interactions.
3. Apply analytic rigour to verify correctness of the overall design approach.
4. Implement the design in a concurrent programming environment, making a critical selection of the facilities that provide support for multi-threading, distribution and synchronisation.

Indicative Module Content

Key concepts of multi-threaded programming including: thread attributes, thread life history, scheduling. Indivisible operations, race conditions, safety and liveness, formal approaches to verifying correctness of a concurrent design. Synchronisation primitives based on use of: - shared variables, test-and-set primitives, semaphores, monitors. Generic concurrent programming problems and their understanding of the lecture content and to develop proficiency in the practical application of object oriented programming skills.

Assessment Plan

	Learning Outcomes Assessed
Component 1	2,3
Component 2	1,2,4

Component 2 - Coursework

Component 1 - This is a closed book examination.

Indicative Bibliography

1. LIU Henry H., 2015. Java Concurrent Programming: A Quantitative Approach. CreateSpace Independent Publishing.
2. GONZALEZ Javier F., 2016. Mastering Concurrency Programming with Java 8. Packt Publishing.
3. FREISEN Jeff, 2015. Java Threads and the Concurrency Utilities. Apress.

solution: - mutual exclusion, resource allocation, event ordering, inter-thread communication.