Module 7: Object Oriented Development

Stage	1
Semester	2
Module Title	Object-Oriented Development
Module Number/Reference	7
Module Status (Mandatory/Elective)	Mandatory
Module ECTS credit	10
Module NFQ level (only if applicable)	8
Pre-requisite Module Titles	None
Co-requisite Module Titles	None
Is this a capstone module? (Yes or No)	No
List of Module Teaching Personnel	Mr Tony Mullins Mr Eoin Carroll Mr Sean Russell

Contact Hours				**Non-contact Hours**			**Total Effort (Hours)**
Lecture	**Practical**	**Tutorial**	**Seminar**	**Assignment**	**Placement**	**Independent work**	
24	36			60		80	200

Allocation of Marks (Within the Module)

	Continuous Assessment	**Project**	**Practical**	**Final Examination**	**Total**
Percentage contribution	60%			40%	**100%**

Intended Module Learning Outcomes

On successful completion of this module learners will be able to:

1. explain the main reasons behind the development of the object-oriented model of software development;
2. implement classes that encapsulate both simple and complex behaviours;
3. explain the relationship between encapsulation and public interfaces;
4. define both inheritance and composition and the differences between them;
5. design and implement classes that use inheritance and composition;

6. apply abstract concepts in an object oriented manner
7. develop confidence in and awareness of the capabilities of object oriented development
8. develop high quality software that is reliable, reusable and maintainable

Module Objectives

This module builds on the work completed in the first semester Programming module and extends the learners knowledge of programing by giving a comprehensive analysis of object-oriented programming. This paradigm leads to software architectures based on the objects every system or subsystem manipulates. In this view software systems are operational models of real or virtual world activities based around the objects that populate these worlds: people, cars, houses, stacks, sets, queues. As in all programming modules, a key objective is the acquisition, on behalf of the learner, of good software engineering skills and the application of these skills to the design and implementation of software components.

Module Curriculum

Introduction and motivation

- Review of procedural paradigm and its limitations.
- Outline of key reasons for development of object-oriented paradigm

Classes and Objects

- Encapsulation: class definition, private, public modifiers, public methods.
- Examples of class definitions and programs that interact with public class interfaces.

Composition and Inheritance

- Composing new classes from existing classes.
- Protecting encapsulation.
- Inheritance and class hierarchies.
- Access modifier protected.
- Polymorphism, multiple inheritance and interfaces.
- Abstract classes.
- Programming with all of above.

Immutable objects

- Defining classes that have immutable state.
- Examples from Java – String, Integer, Double, Boolean, Character.
- Programming with immutable classes.

Object class

- Need to override methods toString, equals and hashCode.
- The comparable interface and implementing the compareTo method.
- Examples of classes that implement comparable interface and override equals, hashCode and toString.
- Hashing functions.
- Issues around problem of cloning and copy constructors.
- The equals contract in Java and issues that arise around inheritance and satisfying the equals contract.

Static Members and Enumerated Types

- Class static members.
- Singleton classes.
- Enumerated types.
- Programming examples of each one.

Robustness and Exceptions

- Examples of the different types of exception and methods for both throwing and catching exceptions.
- Programming with exceptions. Writing exception handlers.

Collection classes

- Genericity and the Collection classes. Sets, Lists, ArrayLists and Maps.
- Using collection classes and user-defined classes.
- Traversing collections.
- Programming problems that use Collection data structures.

Reading lists

Mullins, T. Object-Oriented Programming and the Story of Encapsulation in Java, Griffith College Dublin, 2011.

Meyer, B. *Object-Oriented Software Construction (2nd Edition)*, Prentice-Hall Professional Technical Reference, 2000

Barnes, David. Object-Oriented Programming with Java, Prentice-Hall, 2000

Niño, J. & Hosch, F. A. *Introduction to Programming and Object-Oriented Design Using Java (3rd Edition)*, Wiley, 2008

Eliens Anton. Principles of Object-Oriented Software Development, Addison-Wesley, 1994

Module Learning Environment

Accommodation

Lectures are carried out in classrooms/lecture halls in the College. Lab tutorials are carried out in computer labs throughout the Campus. All have the language software required to deliver the programme.

Library

All learners have access to an extensive range of physical and electronic (remotely accessible) library resources. The library monitors and updates its resources on an on-going basis, in line with the College's Library Acquisition Policy. Lecturers update reading lists for this course on an annual basis as is the norm with all courses run by Griffith College.

Module Teaching and Learning Strategy

The module is delivered through a combination of lectures and practical lab programming sessions. The learners complete a series of worksheets throughout the module that are directly related to the material covered in lectures. The emphasis is on developing sound software engineering skills in practical programming based on theoretical knowledge.

Module Assessment Strategy

The module assessment consists of a series of continuous assignments and a final examination. Each week learners are required to complete a series of programming tasks that relate to the material covered in lectures. The practical lab sessions are used to enforce concepts covered in the lectures and the worksheets are used to ensure that learners are keeping up with the material as it is delivered. Lab sessions are also used to deal with issues emerging from the worksheets. All work submitted by learners is assessed and comments are given to individual learners. All assessment goes to giving a final grade for the work completed by the learner.

Element No	Weighting	Type	Description	Learning Outcome assessed
1	60%	Weekly Lab Book Submission	A series of weekly lab books designed to teach programming concepts	1-8
2	40%	Closed Book Examination	End of Module Examination	1-7