
**DBLN CPSI 3394**

This course is being offered at Griffith College, CAPA's academic partner in Dublin. The Irish academic system differs from the US, particularly with grading. Griffith College professors expect students to undertake a good deal of independent study to achieve a high mark in their classes. For additional information about this class, please contact the Boston Program Advising Team at 1-800-793-0334.

## Object-Oriented Programming

Continuous Assessment: 60%
Exam: 40%

**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 year 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, queus. 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 definiton, 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.

**Immutable objects**

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

**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.