cetb
Bord Oideachais agus Oiliúna Chorcaí
Cork Education and Training Board

Cork Education and Training Board

Programme Module for

Fundamentals of Object Oriented Programming

leading to

Level 5 QQI

Fundamentals of Object Oriented Programming 5N0541

Introduction
This programme module may be delivered as a standalone module leading to certification in a QQI minor award. It may also be delivered as part of an overall validated programme leading to a Level 5 QQI Certificate.

The teacher/tutor should familiarise themselves with the information contained in Cork Education and Training Board's programme descriptor for the relevant validated programme prior to delivering this programme module.

The programme module is structured as follows:

1. Title of Programme Module
2. QQI Component Title and Code
3. Duration in hours
4. Credit Value of QQI Component
5. Status
6. Special Requirements
7. Aim of the Programme Module
8. Objectives of the Programme Module
9. Learning Outcomes
10. Indicative Content
11. Assessment a. Assessment Technique(s) b. Mapping of Learning Outcomes to Assessment Technique(s) c. Guidelines for Assessment Activities
12. Grading
13. Learner Marking Sheet(s), including Assessment Criteria

Integrated Delivery and Assessment
The teacher/tutor is encouraged to integrate the delivery of content where an overlap between content of this programme module and one or more other programme modules is identified. This programme module will facilitate the learner to develop the academic and vocational language, literacy and numeracy skills relevant to the themes and content of the module.

Likewise the teacher/tutor is encouraged to integrate assessment where there is an opportunity to facilitate a learner to produce one piece of assessment evidence which demonstrates the learning outcomes from more than one programme module. The integration of the delivery and assessment of level 5 Communications and level 5 Mathematics modules with that of other level 5 modules is specifically encouraged, as appropriate.

Indicative Content
The indicative content in Section 10 does not cover all teaching possibilities. The teacher/tutor is encouraged to be creative in devising and implementing other approaches, as appropriate. The use of examples is there to provide suggestions. The teacher/tutor is free to use other examples, as appropriate. The indicative content ensures all learning outcomes are addressed but it may not

follow the same sequence as that in which the learning outcomes are listed in Section 9. It is the teacher's/tutor's responsibility to ensure that all learning outcomes are included in the delivery of this programme module.

1. Title of Programme Module
Fundamentals of Object Oriented Programming

2. Component Name and Code
Fundamentals of Object Oriented Programming 5N0541

3. Duration in Hours
150 Hours (typical learner effort, to include both directed and self-directed learning)

4. Credit Value
15 Credits

5. Status
This programme module may be compulsory or optional within the context of the validated programme. Please refer to the relevant programme descriptor, Section 9 Programme Structure

6. Special Requirements
NONE

7. Aim of the Programme Module
This programme module aims to equip the learner with the knowledge, skill and competence to construct and test object oriented computer programs

8. Objectives of the Programme Module

- to assist the learner to develop the academic and vocational language, literacy and numeracy skills related to **Fundamentals of Object Oriented Programming** through the medium of the indicative content
- identify and understand the fundamental concepts of object oriented programming
- acquire competency in an object oriented programming language and an integrated development environment (IDE)
- design, build and debug modular object oriented programs using the appropriate language specific data types, constructs, syntax and semantics
- demonstrate an ability to create classes, methods, fields, functions and data structures
- demonstrate an ability to access purpose-built class libraries /API packages and basic Collection classes (**as and if appropriate**) and incorporate them into user defined classes
- document and comment source code, e.g., write an algorithm, use pseudocode and/or design a program flowchart (as appropriate)
- enable the learner to take responsibility for his/her own learning

9. Learning Outcomes of Level 5 Fundamentals of Object Oriented Programming 5N0541

Learners will be able to:

1. Demonstrate an understanding of the different data types used in object oriented programs
2. Understand the fundamental set of instructions involved in computer programs and how to use them to design and construct object programs to solve problems
3. Demonstrate a knowledge of basic OOP constructs
4. Design and construct modular reusable code blocks
5. Demonstrate familiarity with, and work within, a modern integrated development environment
6. Construct larger programs from smaller programs
7. Model real-world objects in order to build object oriented programs that model real-world activities
8. Explain ways to organise and structure data
9. Properly document program code
10. Debug and test programs
11. Deploy programs to the end user via the front end

10. Indicative Content

This section provides suggestions for programme content **but is not intended to be prescriptive**. The programme module can be delivered through classroom based learning activities, group discussions, one-to-one tutorials, field trips, case studies, role play and other suitable activities, as appropriate.

Section 1 : Programming Fundamentals and Object Oriented Concepts (LO 2, 3)

- Programming Fundamentals
  - define what is meant by the term 'computer program'
  - define what is meant by the term 'software design'
  - state the function of the compiler and the interpreter
  - understand the concept of 'platform independence' and/or architectural neutrality
  - state the purpose of a program algorithm, i.e., understand that a computer program is a set of instructions for a computer to follow
  - differentiate between the syntax and semantics of an object programming language
  - distinguish between user generated source code and object/machine code

- Object Oriented Concepts
  - define what is meant by the term 'object oriented programming'
  - explain the rationale behind object design, i.e., understand the conceptual ideas of modularity and reusability in developing object oriented programs
  - describe objects as having states and behaviours, i.e., explain how objects can be described in terms of their status and inherent characteristics/attributes
  - define a class as a blueprint for an object
  - state the purpose of class libraries and packages
  - state the purpose of methods, members, fields and functions
  - state the purpose of a constructor
  - understand what is meant by the terms *encapsulation, abstraction* and *inheritance*

Section 2: Data Types (LO 1)

- Data Types
  - state what is meant by a data type
  - identify primitive **and/or** fundamental data types in a given object oriented language
  - create and instantiate primitive **and/or** fundamental data types
  - create simple user defined types
  - define what is meant by a logical operator
  - use data types with logical operators
  - state what is meant by a data structure
  - demonstrate an ability to associate data structures with 'real world' problems
  - recognise the use of simple enumerated data types, e.g., *enums*
  - list the common data structures, e.g., *arrays*
  - perform simple type conversions as appropriate
  - write object oriented programs to
    - implement and manipulate data types
    - perform simple type conversions as appropriate

Section 3: Integrated Development Environments (IDEs)
(LO 5)

- Acquire proficiency in an Integrated Development Environment (IDE) **as appropriate**
  - demonstrate an ability to invoke an IDE
  - navigate an IDE competently
  - utilise the IDE to generate and store source code as appropriate
  - correctly compile and execute programs written within an IDE
  - troubleshoot compiler errors
  - use IDE code completion or autocomplete features as appropriate
  - use IDE debugging features as appropriate

Section 4: Modular Programming with Classes, Methods, Fields and Functions
(LO 4, 8, 11)

- Software Design
- demonstrate an ability to break down a problem statement into smaller parts
- use a descriptive problem statement to make coding inferences, i.e., use it to identify and derive possible identifiers/variables, class objects, flow of execution, selection/control statement sequences for the program
- document an algorithm of step-by-step operations/instructions to be performed to solve the problem statement

- Design, build and test object oriented programs using an IDE
- define and implement a single instance of a class
- create interacting base class methods and functions
  - use modular reusable code blocks
- create, instantiate and manipulate fields/variables of different data types
- handle basic input and output
- solve simple arithmetic and logic problems
- simulate basic decision making using a control structure, e.g., use an *if statement*
- perform simple repetition, e.g., use a *while loop, do-while loop* and/or *for loop*
- deploy the program to the end user via a suitable front end

Section 5: Progressive Modular Programming with Inheritance
(LO 4, 5, 6, 7, 8, 10, 11)

- Software Design
  - demonstrate an ability to break down complex problem statements into smaller parts
  - use a descriptive problem statement to make coding inferences, i.e., use it to identify and derive possible identifiers/variables, class objects, flow of execution, selection/control statement sequences for the program
  - document an algorithm of step-by-step operations/instructions to be performed to solve the problem statement

- Design, build and test progressive object oriented programs using an IDE
  - define and implement a class that is based on another object, i.e., inherits functionality from another class/library (single/multiple inheritance)
  - create, instantiate and manipulate class fields/variables of different data types
  - create interacting class methods, members, functions, etc.,

- use modular reusable code blocks

- perform simple method overloading (as appropriate)
- define and implement interacting objects, e.g., method passing, function calls
- handle basic input and output
- solve arithmetic and logic problems
- simulate basic decision making using control structures, e.g., use *if statements, switch statements etc*
- perform simple repetition, e.g., use a *while loop, do-while loop* and/or *for loop*
- deploy the program to the end user via a suitable front end

Section 6: Test and Debug Programs
(LO 9, 10)

- Establish competency and technique in documenting source code and programs
  - state the purpose of pseudocode and provide examples
  - write and revise program algorithms as appropriate
  - demonstrate an ability to place meaningful comments in source code
  - document programs

- Software testing and debugging
  - distinguish between syntax errors and semantic errors in source code
  - demonstrate an ability to interpret compiler errors and debug source code correctly
  - manipulate source code in response to compiler errors and logic errors
  - deconstruct and rearrange modular code blocks to facilitate software revisions and/or incorporate additional source code
  - perform simple stress testing of a program
  - acquire problem solving skills

Assessment

11a. Assessment Techniques

Skills Demonstration 70% (Practical)

Examination (Theory) 30%

11b. Mapping of Learning Outcomes to Assessment Techniques

In order to ensure that the learner is facilitated to demonstrate the achievement of all learning outcomes from the component specification; each learning outcome is mapped to an assessment technique(s). This mapping should **not** restrict an assessor from taking an integrated approach to assessment.

Learning Outcome		Assessment Technique
1.	Demonstrate an understanding of the different data types used in object oriented programs	Skills Demo
2.	Understand the fundamental set of instructions involved in all programs and how to use them to construct programs to solve problems	Examination
3.	Demonstrate a knowledge of basic OOP constructs	Examination
4.	Design and construct modular reusable code blocks	Skills Demo
5.	Demonstrate familiarity with and work within, a modern integrated development environment	Skills Demo
6.	Construct larger programs from smaller programs	Skills Demo
7.	Model real-world objects in order to build object oriented programs that model real-world activities	Skills Demo
8.	Explain ways to organise and structure data	Examination
9.	Properly document program code	Skills Demo
10.	Debug and test programs	Skills Demo
11.	Deploy programs to the end user via the front-end	Skills Demo

11c. Guidelines for Assessment Activities

The component may be delivered using an object oriented language of the assessor's choice, e.g., Java, C#, C++, Visual Basic .NET, Python, Ruby, etc. Where deemed appropriate and/or necessary, an integrated development environment should be used to build and test programs.

The assessor is required to devise assessment briefs and marking schemes for the skills demonstration and an examination paper, marking scheme and outline solution for the examination. In devising the assessment briefs and examination paper, care should be taken to ensure that the learner is given the opportunity to show evidence of achievement of ALL the learning outcomes. Assessment briefs may be designed to allow the learner to make use of a wide range of appropriate media in presenting assessment evidence. Quality assured procedures must be in place to ensure the reliability of learner evidence.

Skills Demonstration	**70%**
Time Allocation: 12 hours per skills demonstration	

The assessor should devise two skills demonstrations that allow the learner to demonstrate their competence in the specified learning outcomes.

Skills Demonstration #1 – 20%:
Design, code and test an object oriented program that performs simple type conversions
It is envisaged the assessor will design a skills demonstration to test the learner's competence and ability in learning outcome 1. The assessor should present the learner with a suitable problem statement and provide clear guidelines/instructions as to what is to be done and what is being assessed. Specifically, the skills demonstration should test the learner's ability to:

- devise and document a simple algorithm based on the problem statement
- make simple coding inferences from the problem statement in order to identify all appropriate data types required
- code and test an object oriented program that provides a solution to the problem statement that:
    - contains a single class
    - features appropriate data types to simulate and/or model the problem
    - uses logical program blocks
    - correctly performs type conversions

The learner will produce:

- a working program (soft and hard copy)
- appropriate program documentation to include an algorithm, relevant screen captures, digital/visual evidence of the development cycle (including evidence of debugging/testing, etc) and indicative critical reasoning

Skills Demonstration #2 – 20%:
Design, code and test a single class object oriented program with
It is envisaged the assessor will design a skills demonstration to test the learner's competence and ability in learning outcomes 4, 5 and 7. The assessor should present the learner with a suitable problem statement and provide clear guidelines/instructions as to what is to be done and what is being assessed. Specifically, the skills demonstration should test the learner's ability to:

- devise and document a simple algorithm based on the problem statement
- make correct coding inferences from the problem statement to identify appropriate fields/variables, objects, control/repetition statements, data sequences, etc, as required
- utilise an integrated development environment (IDE) to code and test an object oriented program that provides a solution to the problem statement that:
    - contains a single class
    - features appropriate data types to simulate and/or model the problem
    - uses logical program blocks
    - correctly implements a control structure, e.g., *if statement*
    - uses a repetition statement, e.g., *while loop, do-while loop, for loop*
    - contains meaningful comments and suitably indented code

The learner will produce:

- a working program (soft and hard copy)
- appropriate program documentation to include the algorithm, relevant screen captures, digital/visual evidence of the development cycle (including evidence of debugging/testing, etc) and indicative critical reasoning

Skills Demonstration #3 - 30%:
Design, code and test an object oriented program that uses inheritance
It is envisaged the assessor will design a skills demonstration to test the learner's competence and ability in learning outcomes 6, 9, 10 and 11. The assessor should present the learner with a suitable problem statement and provide clear guidelines/instructions as to what is to be done and what is being assessed. Specifically, the skills demonstration should test the learner's ability to:

- devise and document an algorithm based on the problem statement
- make correct coding inferences from the problem statement to identify appropriate fields/variables, objects, control/repetition statements, data sequences, etc, as required
- utilise an integrated development environment (IDE) to code and test an object oriented program that provides a solution to the problem statement that:
    - contains a class object that is based on another class
    - makes use of separate class libraries or packages (**as appropriate**[1])
    - features appropriate data types to simulate and/or model the problem
    - using modularised and logical program blocks
    - contains a number of appropriately named methods/functions
    - correctly implements control structures
    - uses repetition statements appropriately
    - contains meaningful comments and suitably indented code
    - is presented via a suitable front-end

The learner will produce:

- a working program (soft and hard copy)
- appropriate program documentation to include the algorithm, relevant screen captures, digital/visual evidence of the development cycle (including evidence of debugging/testing, etc) and indicative critical reasoning

[1] Language dependent

Examination	**30%**

The examination will be theory based and will last 1.5 hours in duration. The assessor should devise an examination that allows the learner to demonstrate their competence and knowledge of the relevant learning outcomes and the theory behind them. The exam will conform to the following structure:

Section A –
10 short questions, answer all questions

- Questions should be based on LO 3

Section B –
Answer 1 long question out of a possible 2. Each question is worth 10 marks

- Both questions to be based on LO 2

Section C –
Answer 1 long question out of a possible 2. Each question is worth 10 marks

- Both questions to be based on LO 8

All instructions for the learner must be clearly outlined in the examination paper. Evidence for this assessment technique may be in a written or digital format (audio/video).

12. Grading

Distinction:	80% - 100%
Merit:	65% - 79%
Pass:	50% - 64%
Unsuccessful:	0% - 49%

At levels 4, 5 and 6 major and minor awards will be graded. The grade achieved for the major award will be determined by the grades achieved in the minor awards.

Fundamentals of Object Oriented Programming 5N0541	Learner Marking Sheet 1 Skills Demonstration 1 (Practical) 20%

Learner's Name: _____ Learner's PPSN: _____

Assessment Criteria	Maximum Mark	Learner Mark
Clearly Documented Source Code • algorithm provided • intelligent use of comments	4	
Program Functionality • working program • prudent use of print formatting	4	
Accurate Programming (Syntax and Semantics) • appropriately named identifiers (class, method, fields/variables) • correctly implemented data types • appropriate data type conversion performed • no syntax or semantic errors	8	
Software Testing/Debugging • evidence of software testing, e.g., documentation of problems/bugs • screen captures, visual/digital evidence provided	4	
Total Mark	20	

Assessor's Signature: _____ Date: _____

External Authenticator's Signature: _____ Date: _____

Fundamentals of Object Oriented Programming 5N0541	Learner Marking Sheet 2 Skills Demonstration 2 (Practical) 20%

Learner's Name: _____ Learner's PPSN: _____

Assessment Criteria	Maximum Mark	Learner Mark
Clearly Documented Source Code • algorithm/pseudocode provided • intelligent use of comments • source code correctly indented	3	
Program Functionality • working program • prudent use of print formatting • appropriate layout or screen ratio applied (front-end)	3	
Accurate Programming (Syntax and Semantics) • source code generated within IDE • appropriately named identifiers (class, method/function, fields/variables) • working selection statements, e.g., *if-else* • working control structures, e.g., *while-loop, for-loop* • no syntax or semantic errors	10	
Software Testing/Debugging • evidence of software testing, e.g., documentation of problems/bugs • screen captures, visual/digital evidence provided	4	
Total Mark	20	

Assessor's Signature: _____ Date: _____

External Authenticator's Signature: _____ Date: _____

Fundamentals of Object Oriented Programming 5N0541	Learner Marking Sheet 3 Skills Demonstration 3 (Practical) 30%

Learner's Name: _____ Learner's PPSN: _____

Assessment Criteria	Maximum Mark	Learner Mark
Clearly Documented Source Code • algorithm/pseudocode provided • intelligent use of comments • source code correctly indented	3	
Program Functionality • working program • appropriate layout or screen ratio applied (front-end) • prudent use of print formatting	3	
Accurate Programming (Syntax and Semantics) • source code generated within IDE • correctly accessed packages/library classes (as appropriate) • appropriately named base class(es) • appropriately named identifiers (fields/variables) • appropriately named methods/functions • appropriate use of methods, e.g., correct method calls, return values • working selection statements, e.g., *if-else, switch* • working control structures, e.g., *while, do-while, for loops* • correctly implemented data structure(s) • no syntax or semantic errors	20	
Software Testing/Debugging • evidence of software testing, e.g., documentation of problems/bugs • screen captures, written evidence, etc	4	
Total Mark	30	

Assessor's Signature: _____ Date: _____

External Authenticator's Signature: _____ Date: _____

Fundamentals of Object Oriented Programming 5N0541	Learner Marking Sheet 4 Examination Theory 30%

Learner's Name: _____ Learner's PPSN: _____

Assessment Criteria	Maximum Mark	Learner Mark
Section A: [Short Questions] **10 Questions, answer all questions (1 mark each)** Question No.:* _____ _____ _____ _____ _____ _____ _____ _____ _____ _____	1 1 1 1 1 1 1 1 1 1	
Subtotal	10	
Section B: [Long Questions] **2 Questions, answer <u>ONE</u> (10 marks)** (Indicate question answered) Question No.:* _____	10	
Subtotal	10	
Section C: [Long Questions] **2 Questions, answer <u>ONE</u> (10 marks)** (Indicate question answered) Question No.:* _____	10	
Subtotal	10	
Total Mark	30	

Assessor's Signature: _____ Date: _____

External Authenticator's Signature: _____ Date: _____