MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

Module Information معلومات المادة الدراسية					
Module Title	**OOP Programming Fundamentals**			**Module Delivery**	
Module Type	**Core**			☒**Theory** ☒**Lecture** ☒**Lab** ☐**Tutorial** ☐**Practical** ☒**Seminar**	
Module Code	COM-211				
ECTS Credits	4				
SWL (hr/sem)	**100**				
Module Level	1		**Semester of Delivery**	2	
Administering Department	com		**College**	cos	
Module Leader	Ismael Salih Aref		**e-mail**	asmaelsalih@uodiyala.edu.iq	
Module Leader's Acad. Title	Assist.Lect		**Module Leader's Qualification**	MSC	
Module Tutor	Name (if available)		**e-mail**	E-mail	
Peer Reviewer Name	Name		**e-mail**	E-mail	
Scientific Committee Approval Date	01/07/2024		**Version Number**	1.0	

Relation with other Modules العلاقة مع المواد الدراسية الأخرى			
Prerequisite module	**Programming Language c++**	**Semester**	1
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents
أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية

Module Objectives
أهداف المادة الدراسية

The educational objectives of this course are

1- **Understanding Core Concepts of OOP:**

- **Classes and Objects:** Understanding the foundational building blocks of OOP, including how to define classes (blueprints) and create objects (instances of classes).
- **Encapsulation:** Learning how to bundle data (attributes) and methods (functions) that operate on the data into a single unit or class, promoting data hiding and reducing complexity.
- **Inheritance:** Grasping how new classes can be derived from existing ones, allowing for code reuse and the creation of hierarchical class structures.
- **Polymorphism:** Understanding how different classes can be treated as instances of the same class through interfaces, allowing for flexibility in code through method overriding and overloading.
- **Abstraction:** Learning to focus on essential qualities of an object while hiding unnecessary details, making complex systems easier to manage.

2- **Developing Problem-Solving Skills:**

- **Modeling Real-World Systems:** Teaching students to represent real-world entities as objects, helping to develop systems that are intuitive and maintainable.
- **Design Patterns:** Introducing common design patterns that solve recurring problems in OOP, fostering best practices in software development.
- **Code Reusability:** Emphasizing the importance of creating reusable, modular code that can be easily extended and maintained.

3- **Improving Software Design and Architecture:**

- **Software Design Principles:** Educating students on principles like SOLID (Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion) to create well-structured and robust code.
- **Object-Oriented Analysis and Design (OOAD):** Training students to analyze and design software systems using OOP principles, focusing on creating scalable and maintainable architectures.

4- **Enhancing Team Collaboration and Code Maintenance:**

- **Version Control Integration:** Learning to use version control systems (e.g., Git) in the context of OOP projects to manage code changes collaboratively.
- **Code Documentation and Comments:** Understanding the importance of documenting code, especially in large, object-oriented projects, to facilitate collaboration and maintenance.
- **Testing and Debugging:** Gaining skills in writing unit tests for classes and objects, and learning debugging techniques specific to object-oriented codebases.

Module Learning Outcomes مخرجات التعلم للمادة الدراسية	**1. Knowledge and Understanding:** • **MLO 1:** Demonstrate a comprehensive understanding of the fundamental principles of Object-Oriented Programming, including concepts such as classes, objects, inheritance, polymorphism, encapsulation, and abstraction. • **MLO 2:** Understand and apply the principles of software design and architecture, including the use of design patterns and best practices in OOP. • **MLO 3:** Explain the benefits and limitations of the OOP paradigm in software development, including its impact on code reusability, maintainability, and scalability. **2. Cognitive/Intellectual Skills:** • **MLO 4:** Analyze real-world problems and design effective OOP solutions by modeling appropriate classes, objects, and relationships. • **MLO 5:** Critically evaluate and apply design patterns to solve common software design problems. • **MLO 6:** Assess the trade-offs between different object-oriented designs in terms of efficiency, complexity, and scalability. **3. Practical/Professional Skills:** • **MLO 7:** Develop and implement object-oriented software using a relevant programming language (e.g., Java, C++, Python) that adheres to industry standards and best practices. • **MLO 8:** Apply techniques for debugging, testing, and maintaining object-oriented code, including the use of unit tests and version control systems. • **MLO 9:** Work collaboratively in a team environment to design and develop a substantial object-oriented software project, demonstrating effective communication and project management skills. **4. Key Transferable Skills:** • **MLO 10:** Demonstrate problem-solving skills by breaking down complex problems into manageable components using OOP techniques. • **MLO 11:** Communicate technical information effectively, both verbally and in writing, through documentation, code comments, and presentations. • **MLO 12:** Adapt to new and emerging technologies in object-oriented programming, demonstrating lifelong learning and the ability to stay current with industry trends.
Indicative Contents المحتويات الإرشادية	The indicative content of an Object-Oriented Programming (OOP) course includes an introduction to core concepts like classes, objects, inheritance, encapsulation, polymorphism, and abstraction, along with advanced topics such as composition vs. inheritance, design patterns, and SOLID principles. It also covers object-oriented analysis and design (OOAD), practical implementation in a chosen programming language, and testing/debugging techniques. Students will work on hands-on projects, including collaborative team development, integrating OOP with databases, and exploring modern frameworks and libraries. The course concludes with discussions on contemporary OOP languages, emerging trends, and the future direction of software development.

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	
Strategies	• Lectures • Tutorials • Problem solving • Lab • Case study • Small project

Student Workload (SWL) الحمل الدراسي للطالب محسوب لـ ١٥ اسبوعا			
Structured SWL (h/sem) الحمل الدراسي المنتظم للطالب خلال الفصل	60	**Structured SWL (h/w)** الحمل الدراسي المنتظم للطالب أسبوعيا	4
Unstructured SWL (h/sem) الحمل الدراسي غير المنتظم للطالب خلال الفصل	40	**Unstructured SWL (h/w)** الحمل الدراسي غير المنتظم للطالب أسبوعيا	2.66
Total SWL (h/sem) الحمل الدراسي الكلي للطالب خلال الفصل	**100**		

Module Evaluation تقييم المادة الدراسية					
		Time/Number	**Weight (Marks)**	**Week Due**	**Relevant Learning Outcome**
Formative assessment	**Quizzes**	3	10% (10)	4 and 9	LO #1, #2 and #10, #11
	Assignments	3	5% (5)	5 and 12	LO #3, #4 and #6, #7
	Projects / Lab.	2	10% (10)	Continuous	All
	Report	1	10% (10)	13	LO #5, #8 and #10
Summative assessment	**Midterm Exam**	2hr	10% (10)	7	LO #1 - #7
	Final Exam	2hr	50% (50)	16	All
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus) المنهاج الاسبوعي النظري	
	Material Covered
Week 1	Introduction to inheritance
Weeks 2,3	Type of inheritance

Weeks 4,5,6	Constructor inheritance , friend function , virtual function
Week 7	Mid-term Exam
Weeks 8,9	UML Diagrams
Weeks 10 and 11	SOLID Principles
Weeks 12 and 13	Interfaces and abstract classes , Abstract Methods
Week 14	polymorphism
Week 15	template
Week 16	**Preparatory week before the final Exam**

Delivery Plan (Weekly Lab. Syllabus) المنهاج الاسبوعي للمختبر	
	Material Covered
Weeks 1 and 2	Apply inheritance
Weeks 3 and 4	Methods and data member inheritance
Weeks 5,6 and 7	Constructor inheritance
Weeks 8 and 9	friend function, virtual function
Weeks 10,11 and 12	Interfaces and abstract classes , Abstract Methods
Weeks 13 and 14	Polymorphism examples.
Week 15	Template examples.

Learning and Teaching Resources مصادر التعلم والتدريس		
	Text	**Available in the Library?**

Required Texts	• Programming in C++ Frank Vahid and Roman Lysecky Available through the zyBooks website directly • A C++ compiler and/or IDE. There are many out there, but the only two that are officially supported: - CLion (on Windows and macOS) - Visual Studio (Windows only)	Yes
Recommended Texts	▪ Think Like a Programmer, An Introduction to Creative Problem Solving V. Anton Spraul ISBN: 978-1593274245 ▪ A good text editor, such as: Notepad++ (This is my personal favorite) Sublime Text Atom, or Vim, or anything else you might prefer	No
Websites	1-http://www.cplusplus.com/ 2-https://www.w3schools.com/cpp/	

Grading Scheme
مخطط الدرجات

Group	Grade	التقدير	Marks %	Definition
Success Group (50 - 100)	**A -** Excellent	امتياز	90 - 100	Outstanding Performance
	B - Very Good	جيد جدا	80 - 89	Above average with some errors
	C - Good	جيد	70 - 79	Sound work with notable errors
	D - Satisfactory	متوسط	60 - 69	Fair but with major shortcomings
	E - Sufficient	مقبول	50 - 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX –** Fail	راسب (قيد المعالجة)	(45-49)	More work required but credit awarded
	F – Fail	راسب	(0-44)	Considerable amount of work required

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.