

وزارة التعليم العالي والبحث العلمي
جهاز الاشراف والتقويم العلمي
دائرة ضمان الجودة والاعتماد الأكاديمي

استمارة وصف البرنامج الأكاديمي للعام الدراسي ٢٠٢٥-٢٠٢٦ للكليات والمعاهد

الجامعة : جامعة شط العرب الاهلية

الكلية \المعهد : الكلية التقنية الهندسية

القسم العلمي : قسم هندسة تقنيات الحاسوب

تاريخ ملء الملف : ٢٠٢٥/٩/١١

التوقيع

التوقيع

اسم المعاون العلمي : م. عصام خليل ابراهيم

اسم رئيس القسم : م. مكي جبار عبد الرزاق

التاريخ :

التاريخ : ٢٠٢٥ / ٩ / ١١

م. عصام خليل ابراهيم
معاون العميد للشؤون العلمية

دقق الملف من قبل

شعبة ضمان الجودة والأداء الجامعي

اسم مدير شعبة ضمان الجودة والأداء الجامعي

التوقيع

٢٠٢٥ / ٩ / ١١

التوقيع

مصادقة السيد العميد

أ.م.د. هازن عبداللّه علوان

عميد الكلية التقنية الهندسية

MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

Module Information			
معلومات المادة الدراسية			
Module Title	Object Oriented Programming		Module Delivery
Module Type	S		☒ Theory ☐ Lecture ☒ Lab ☒ Tutorial ☐ Practical ☐ Seminar
Module Code	CET2102		
ECTS Credits	6		
SWL (hr/sem)	150		
Module Level	2	Semester of Delivery	3
Administering Department	CET	College	EETC
Module Leader	Loay Talib Ahmed	e-mail	Loay.alsaffar@mtu.edu.iq
Module Leader's Acad. Title	Lecturer	Module Leader's Qualification	Ph.D.
Module Tutor	Dr. Hisham Raad jafer MERZEH	e-mail	hisham.merzeh@mtu.edu.iq
Peer Reviewer Name	Dr. Osama Abbas Hussein	e-mail	osama.abbas@mtu.edu.iq
Scientific Committee Approval Date	29/10/2025	Version Number	1.0

Relation with other Modules			
العلاقة مع المواد الدراسية الأخرى			
Prerequisite module	Programming Essentials / CET1203	Semester	2
Co-requisites module	None	Semester	

Module Aims, Learning Outcomes and Indicative Contents أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية	
Module Aims أهداف المادة الدراسية	1. Understand and apply object-oriented programming principles. 2. Design and implement object-oriented solutions to programming problems. 3. Utilize C++ libraries and frameworks for application development. 4. Implement data abstraction and encapsulation for secure and efficient code. 5. Plan and execute testing strategies for reliable programs. 6. Debug and optimize program performance for efficient execution.
Module Learning Outcomes مخرجات التعلم للمادة الدراسية	1. Demonstrate a clear understanding of object-oriented programming principles, including inheritance, polymorphism, and encapsulation. 2. Design and implement classes and objects to represent real-world entities, applying appropriate inheritance and encapsulation. 3. Utilize C++ libraries and frameworks effectively to develop robust and scalable applications. 4. Implement data abstraction and encapsulation techniques to ensure secure and efficient code. 5. Plan and execute comprehensive testing strategies to validate the functionality and reliability of object-oriented programs. 6. Identify and debug program errors using appropriate tools and techniques, enhancing program robustness. 7. Evaluate and optimize program performance through code analysis and profiling, improving execution efficiency. 8. Collaborate effectively with peers to develop object-oriented solutions to complex programming challenges. 9. Apply exception handling techniques to handle errors and ensure program stability. 10. Demonstrate proficiency in utilizing debugging tools to identify and fix program errors. 11. Apply object-oriented design patterns and principles to analyze and solve programming problems. 12. Evaluate the efficiency and effectiveness of object-oriented solutions through critical analysis and optimization techniques.
Indicative Contents المحتويات الإرشادية	Indicative content includes the following. <u>Part A: Introduction to Object-Oriented Programming (8 hours)</u> - Overview of object-oriented programming principles and concepts - Classes, objects, and their relationships - Inheritance and polymorphism

	- Encapsulation and data abstraction <u>Part B: Designing Object-Oriented Solutions (12 hours)</u> - Problem analysis and requirements gathering - Identifying classes and objects - Object-oriented design principles and patterns - Designing class hierarchies and relationships - UML diagrams for visualizing designs <u>Part C: Implementing Object-Oriented Solutions in C++ (20 hours)</u> - C++ language essentials for object-oriented programming - Implementing classes and objects in C++ - Inheritance and polymorphism in C++ - Handling exceptions in C++ - Utilizing C++ libraries and frameworks <u>Part D: Testing and Debugging Object-Oriented Programs (12 hours)</u> - Testing methodologies and strategies - Unit testing and test-driven development - Integration testing and system testing - Debugging techniques and tools - Error handling and exception management <u>Part E: Optimization and Performance Analysis (8 hours)</u> - Profiling and performance analysis tools - Identifying performance bottlenecks - Optimization techniques for object-oriented programs - Memory management and resource optimization <u>Part F: Collaborative Object-Oriented Programming (8 hours)</u> - Collaborative development environments and version control systems - Code reviews and best practices - Pair programming and team collaboration - Communication and coordination in object-oriented projects <u>Part G: Project Work and Application Development (20 hours)</u> - Applying object-oriented principles and techniques in a practical project - Developing a complete application using C++ and object-oriented design - Project planning, implementation, and documentation - Integration of various modules and testing the application
--	--

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	
Strategies	The learning and teaching strategies for the Object-Oriented Programming Course include lectures to introduce concepts, practical exercises for hands-on programming, group discussions for collaboration, case studies for real-world application, code reviews for feedback, practical projects to apply knowledge, guest lectures for industry insights, online resources for self-study, assessments to evaluate understanding, and presentations to enhance communication skills. These strategies aim to actively engage students, develop their programming abilities, and foster a deep understanding of object-oriented programming principles.

Student Workload (SWL) الحمل الدراسي للطالب موزع على (15) اسبوع			
Structured SWL (h/sem) الحمل الدراسي المنتظم للطالب خلال الفصل	79	Structured SWL (h/w) الحمل الدراسي المنتظم للطالب أسبوعياً	5.26
Unstructured SWL (h/sem) الحمل الدراسي غير المنتظم للطالب خلال الفصل	71	Unstructured SWL (h/w) الحمل الدراسي غير المنتظم للطالب أسبوعياً	4.73
Total SWL (h/sem) الحمل الدراسي الكلي للطالب خلال الفصل	150		

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

Delivery Plan (Weekly Syllabus) المنهاج الاسبوعي النظري	
	Material Covered
Week 1	Introduction to Object-Oriented Programming
Week 2	Classes, Objects, and Relationships
Week 3	Inheritance and Polymorphism principles
Week 4	Encapsulation and Data Abstraction
Week 5	Problem Analysis and Requirements Gathering
Week 6	Object-Oriented Design Principles and Patterns
Week 7	Mid-term Exam
Week 8	C++ Language Essentials and Advanced Topics
Week 9	Implementing Classes and Objects in C++
Week 10	Implementing Inheritance and Polymorphism in C++
Week 11	Handling Exceptions in C++
Week 12	Utilizing C++ Libraries and Frameworks
Week 13	Testing Methodologies and Strategies in C++
Week 14	Debugging Techniques and Tools in C++
Week 15	Optimization and Performance Analysis in C++
Week 16	Preparatory week before the final Exam

Delivery Plan (Weekly Lab. Syllabus) المنهاج الاسبوعي للمختبر	
	Material Covered
Week 1	Introduction to C++ programming environment and basic syntax.
Week 2	Implementing simple classes and objects.
Week 3	Experimenting with inheritance and polymorphism in C++.
Week 4	Implementing data abstraction and encapsulation.
Week 5	Problem-solving exercise using object-oriented design principles and patterns.
Week 6	Utilizing C++ libraries and frameworks for application development.
Week 7	Midterm Exam (No lab session).
Week 8	Implementing exception handling techniques in C++.
Week 9	Testing and debugging strategies for object-oriented programs.
Week 10	Profiling and performance analysis of C++ programs.
Week 11	Code optimization techniques for object-oriented programming.
Week 12	Collaborative programming exercise utilizing version control systems.
Week 13	Implementing advanced data structures using object-oriented techniques.
Week 14	Project work and application development using object-oriented concepts.
Week 15	review and practice exercises, Preparatory for Final Exam.
Week 16	Final Exam (No lab session).

Learning and Teaching Resources مصادر التعلم والتدريس		
	Text	Available in the Library?
Required Texts	"Object-Oriented Programming in C++" by Robert Lafore	
Recommended Texts	"Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides	
Websites	https://www.w3schools.com/cpp/cpp_oop.asp	

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.				