# MODULE DESCRIPTION FORM

# نموذج وصف المادة الدراسية

| Module Information<br>معلومات المادة الدراسية | | | | | |
|---|---|---|---|---|---|
| **Module Title** | **Introduction to Object Oriented Language** | | | **Module Delivery** | |
| **Module Type** | **Core** | | | ☒**Theory**<br>☒**Lecture**<br>☒**Lab**<br>☐**Tutorial**<br>☐**Practical**<br>☒**Seminar** | |
| **Module Code** | **COM-211** | | | | |
| **ECTS Credits** | 7 | | | | |
| **SWL (hr/sem)** | **175** | | | | |
| **Module Level** | 2 | | **Semester of Delivery** | 3 | |
| **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** | 01/08/2024 | | **Version Number** | 1.0 | |

| Relation with other Modules<br>العلاقة مع المواد الدراسية الأخرى | | | |
|---|---|---|---|
| **Prerequisite module** | **Programming Language1** | **Semester** | 1 |
| **Co-requisites module** | None | **Semester** | |

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

| | |
|---|---|
| **Module Objectives**<br>أهداف المادة الدراسية | The educational objectives of this course are<br>**1- Understanding Core Concepts of OOP:**<br>• **Classes and Objects:** Understanding the foundational building blocks of OOP, including how to define classes (blueprints) and create objects (instances of classes).<br>• **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.<br>• **Inheritance:** Grasping how new classes can be derived from existing ones, allowing for code reuse and the creation of hierarchical class structures.<br>• **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.<br>• **Abstraction:** Learning to focus on essential qualities of an object while hiding unnecessary details, making complex systems easier to manage.<br>**2- Developing Problem-Solving Skills:**<br>• **Modeling Real-World Systems:** Teaching students to represent real-world entities as objects, helping to develop systems that are intuitive and maintainable.<br>• **Design Patterns:** Introducing common design patterns that solve recurring problems in OOP, fostering best practices in software development.<br>• **Code Reusability:** Emphasizing the importance of creating reusable, modular code that can be easily extended and maintained.<br>**3- Improving Software Design and Architecture:**<br>• **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.<br>• **Object-Oriented Analysis and Design (OOAD):** Training students to analyze and design software systems using OOP principles, focusing on creating scalable and maintainable architectures. |
| **Module Learning Outcomes**<br>مخرجات التعلم للمادة الدراسية | **1. Knowledge and Understanding:**<br>• **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.<br>• **MLO 2:** Understand and apply the principles of software design and architecture, including the use of design patterns and best practices in OOP.<br>• **MLO 3:** Explain the benefits and limitations of the OOP paradigm in software development, including its impact on code reusability, maintainability, and scalability.<br>**2. Cognitive/Intellectual Skills:** |

| | |
|---|---|
| | • **MLO 4:** Analyze real-world problems and design effective OOP solutions by modeling appropriate classes, objects, and relationships.<br>• **MLO 5:** Critically evaluate and apply design patterns to solve common software design problems.<br>• **MLO 6:** Assess the trade-offs between different object-oriented designs in terms of efficiency, complexity, and scalability.<br>**3. Practical/Professional Skills:**<br>• **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.<br>• **MLO 8:** Apply techniques for debugging, testing, and maintaining object-oriented code, including the use of unit tests and version control systems.<br>• **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.<br>**4. Key Transferable Skills:**<br>• **MLO 10:** Demonstrate problem-solving skills by breaking down complex problems into manageable components using OOP techniques.<br>• **MLO 11:** Communicate technical information effectively, both verbally and in writing, through documentation, code comments, and presentations.<br>• **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**<br>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**<br>استراتيجيات التعلم والتعليم | |
|---|---|
| **Strategies** | • Lectures<br>• Tutorials<br>• Problem solving<br>• Lab<br>• Case study<br>• Small project |

| Student Workload (SWL)<br>الحمل الدراسي للطالب محسوب لـ ١٥ اسبوعا | | | |
|---|---|---|---|
| **Structured SWL (h/sem)**<br>الحمل الدراسي المنتظم للطالب خلال الفصل | 99 | **Structured SWL (h/w)**<br>الحمل الدراسي المنتظم للطالب أسبوعيا | 6.6 |
| **Unstructured SWL (h/sem)**<br>الحمل الدراسي غير المنتظم للطالب خلال الفصل | 76 | **Unstructured SWL (h/w)**<br>الحمل الدراسي غير المنتظم للطالب أسبوعيا | 5 |
| **Total SWL (h/sem)**<br>الحمل الدراسي الكلي للطالب خلال الفصل | **175** | | |

| Module Evaluation<br>تقييم المادة الدراسية | | | | | |
|---|---|---|---|---|---|
| | | **Time/Number** | **Weight (Marks)** | **Week Due** | **Relevant Learning Outcome** |
| **Formative assessment** | **Quizzes** | 2 | 10% (10) | 4 and 9 | LO #1, #2 and #10, #11 |
| | **Assignments** | 2 | 10% (10) | 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** | 3hr | 50% (50) | 16 | All |
| **Total assessment** | | | 100% (100 Marks) | | |

| Delivery Plan (Weekly Syllabus)<br>المنهاج الاسبوعي النظري | |
|---|---|
| | **Material Covered** |
| **Week 1** | Introduction to OOP |
| **Weeks 2** | Classes and Objects: Explain Structure of Simple Class |
| **Week 3** | Access Specifiers: public, private and protected |
| **Weeks 4** | Encapsulation principle (data hiding) |
| **Week 5** | Constructor (usage and advantage) |
| **Week 6** | Destructor (Purpose, syntax, advantages) |
| **Week 7** | Mid-term Exam |
| **Week 8** | Inheritance Basics |
| **Weeks 9** | Types of Inheritance |

| | |
|---|---|
| **Weeks 10** | Function Overriding in class |
| **Weeks 11** | constructor overloading |
| **Weeks 12** | Polymorphism |
| **Week 13** | Virtual Functions |
| **Week 14** | Operator Overloading |
| **Week 15** | Homework Sheets solving |
| **Week 16** | **Preparatory week before the final Exam** |

| **Delivery Plan (Weekly Lab. Syllabus)**<br>المنهاج الاسبوعي للمختبر | |
|---|---|
| | **Material Covered** |
| **Week 1** | Simple Cass Structure |
| **Week 2** | Create Classes and objects |
| **Week 3** | Public and private example (control data access) |
| **Week** 4 | Protected access specifier and private difference by examples. |
| **Week 5** | Structures of contractor |
| **Week 6** | Pointer and destructor roles |
| **Week 7** | Mid-term Exam |
| **Week 8** | Explain Inheritance structure by examples |
| **Week 9** | Many examples to explain inheritance levels and types. |
| **Week 10** | Apply Function Overriding in class |
| **Week 11** | Apply constructor overloading with different arguments |
| **Week 12** | Explain Polymorphism concepts by examples. |
| **Week 13** | Examples to show usage of Virtual Functions |
| **Week 14** | Operator Overloading (unary and binary) |
| **Week 15** | Solving Examples Sheet |
| **Week 16** | Exam |

| **Learning and Teaching Resources**<br>مصادر التعلم والتدريس | | |
|---|---|---|
| | **Text** | **Available in the Library?** |
| **Required Texts** | • Programming in C++<br>Frank Vahid and Roman Lysecky | Yes |

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

| **Grading Scheme**<br>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.