CS3307A Course Outline - Fall 2025

1. Course Information

Course Information
COMPSCI 3307A - OBJECT-ORIENTED DESIGN & ANALYSIS

Time/Place
9:30 - 10:30 AM, Tuesdays[redacted]
9:30 - 11:30 AM, Thursdays[redacted]

2. Course Description

This course introduces students to the principles and practices of Object-Oriented Design and Analysis (OODA) with implementation in C++. It covers fundamental object-oriented concepts, UML modeling techniques, and SOLID design principles, followed by an in-depth exploration of design patterns across creational, structural, and behavioral categories. Students will also learn software testing practices, including unit testing and mocking, to validate design and implementation. Through lectures, case studies, and a comprehensive final project, the course emphasizes building scalable, maintainable, and efficient software systems that reflect real-world development challenges.

Prerequisites: Either (Computer Science 2212A/B/Y) or (Computer Science 2210A/B, 2211A/B, Electrical and Computer Engineering 3375A/B, and registration in the fourth year of a BESc program in Computer Engineering or Mechatronic Systems Engineering.)

Antirequisites: Software Engineering 3350A/B

Note: Unless you have either the prerequisites for this course or written special permission from your dean to enroll in it, you may be removed from this course, and it will be deleted from your record. This decision may not be appealed. You will receive no adjustment to your fees if you are dropped from a course for failing to have the necessary prerequisites.

Course Goal
The primary goal of this course is to equip students with the knowledge and skills to design, model, and implement robust object-oriented software systems. By combining theoretical foundations with practical applications, the course prepares students to analyze requirements, apply design principles, and leverage design patterns to create flexible, reusable, and testable software solutions.

Learning Outcomes

By the end of this course, students will be able to:

1. Explain and apply the core concepts of object-oriented programming, including encapsulation, inheritance, polymorphism, and abstraction.
2. Analyze software requirements and translate them into effective UML models (class, sequence, and use case diagrams).
3. Apply the SOLID principles to improve code quality, reduce coupling, and increase maintainability.
4. Identify, implement, and justify appropriate design patterns (creational, structural, and behavioral) to solve recurring software design problems.
5. Develop scalable and maintainable C++ systems that integrate object-oriented design principles.
6. Use unit testing frameworks (GoogleTest and Google Mock) to validate the correctness and robustness of object-oriented systems.
7. Critically evaluate the quality of software designs with respect to scalability, maintainability, and efficiency.
8. Work independently or collaboratively to deliver a complete software project from requirements gathering through implementation and testing.

3. Instructor Information

Instructors	Email	Office	Phone	Office Hours
Dr. Umair Rehman (Course Coordinator)	urehman6@uwo.ca	[redacted]	[redacted]	MS Teams, Tuesday 2:00 PM – 3:00 PM

Course Communication Guidelines

Primary Channels for Communication

- Email: Formal communication, such as course updates and assignment submissions, should be conducted via email. Students are required to use their Western (@uwo.ca) email addresses when contacting instructors. This is essential for maintaining a formal and secure line of communication.
- MS Teams: For real-time interactions, queries about course material, and brief updates, MS Teams will be utilized. This platform facilitates more immediate and interactive communication.

Email Etiquette

- Subject Line: When sending an email, students must include the course number followed by a brief description of the email's purpose in the subject line. For example, "CS3307 – Project Query" or "CS3307 – Request for Meeting".
- Instructor Messaging: The instructor's email is integrated into MS Teams, allowing students to message the instructor directly through the platform. This should be used for quick questions or clarifications.

4. Course Schedule

Week	Topics
Week 1: Introduction to Object-Oriented Design and Analysis Sept 4, 2025	In this introductory week, students will gain an overview of the course and establish a foundation in object-oriented design and analysis (OODA). We will contrast the object-oriented approach with functional programming to highlight their differences in thinking and problem-solving. Core concepts such as cohesion and coupling will be introduced as measures of design quality, emphasizing why object-oriented methods provide advantages in building maintainable, scalable, and reusable software systems.
Week 2: Core Object-Oriented Concepts Sept 9 and Sept 11, 2025	This week focuses on the core principles of object-oriented programming: encapsulation, inheritance, polymorphism, and abstraction. Students will explore how these concepts contribute to modularity, flexibility, and reusability, while also recognizing their potential pitfalls if misapplied. Through discussion and examples, abstract classes and interfaces will be compared with encapsulation to clarify their distinct roles. Real-world case studies will demonstrate how these principles shape the quality and maintainability of software systems.
Week 3: Requirements Gathering and Use Case Modeling/ UML Modelling Sept 16 and Sept 18, 2025	This week introduces students to requirements gathering and UML modeling as the bridge between problem understanding and system design. They will learn techniques for eliciting requirements, distinguishing functional from non-functional needs, and translating these into effective use case models. Building on this, students will practice identifying classes, defining relationships, and representing them in UML class, sequence, and use case diagrams. Emphasis will be placed on refining diagrams for clarity, applying best practices, and using case studies to connect modeling techniques to real-world software development.
Week 4: SOLID Principles and Design Practices Sept 23 and Sept 25, 2025	This week covers the SOLID principles, a cornerstone of effective object-oriented design. Students will examine each principle—Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, and Dependency Inversion—understanding how they guide the creation of flexible, maintainable, and scalable systems. Through examples and refactoring exercises, they will see how applying these principles reduces coupling, improves cohesion, and lays the groundwork for clean architecture and the effective use of design patterns.
Sept 30, 2025 (Truth and Reconciliation Day – no class)	
Week 5: Creational Design Patterns I	This week introduces design patterns as proven solutions to recurring software design problems, emphasizing their role in

Oct 2, 2025	improving code reuse, flexibility, and communication among developers. Students will explore the creational category of patterns, beginning with the Singleton pattern for managing single instances, followed by the Factory Method and Abstract Factory patterns for handling complex object creation. This content will continue into the following week since we have only one class this week due to no class on September 30 (Truth and Reconciliation Day).
Week 6: Creational Design Patterns II Oct 7 and Oct 9, 2025	This week continues with creational design patterns, focusing on the Builder and Prototype patterns. Students will learn how the Builder pattern supports step-by-step object construction, contrasting it with factory-based approaches, while the Prototype pattern demonstrates object creation through cloning. The session will also highlight how different creational patterns can be combined effectively, giving students a toolkit of strategies for handling diverse object creation scenarios in real-world systems.
Week 7: Structural Design Patterns I Oct 14 and Oct 16, 2025	This week introduces structural design patterns, which focus on how classes and objects are composed to form larger, more flexible structures. Students will examine the Adapter pattern for reconciling incompatible interfaces, the Composite pattern for treating individual objects and groups uniformly, and the Decorator pattern for dynamically adding responsibilities without altering existing code. Through examples and case studies, they will see how these patterns simplify system architecture while enhancing extensibility and reuse.
Week 8: Structural Design Patterns II Oct 21 and Oct 23, 2025	This week expands on structural design patterns by exploring techniques that simplify and optimize system interactions. Students will study the Façade pattern as a way to provide a unified interface to complex subsystems, and the Proxy pattern as a means of controlling or deferring access to objects. The Bridge pattern will illustrate how to decouple abstractions from implementations for greater flexibility, while the Flyweight pattern demonstrates memory-efficient object sharing. The session will conclude with a discussion on combining structural patterns to address real-world software design challenges.
Week 9: Behavioral Design Patterns I Oct 28 and Oct 30, 2025	This week introduces behavioral design patterns, which focus on how objects interact and distribute responsibilities within a system. Students will begin with the Iterator pattern, learning how it provides a uniform way to traverse collections without exposing their underlying structure. They will then explore the Command pattern, which encapsulates requests as objects to support undo/redo operations, logging, and flexible task execution. Together, these patterns demonstrate how behavioral techniques enhance communication and control in object-oriented design.
Reading Week (Nov 4 and Nov 6, 2025)	

Week 10: Behavioral Design Patterns II Nov 11 and Nov 13, 2025	This week continues with key behavioral design patterns that enable flexible and dynamic object interactions. Students will examine the Strategy pattern for selecting algorithms at runtime, the Observer pattern for implementing publish–subscribe relationships, and the State pattern for managing context-dependent behavior. They will also study the Chain of Responsibility pattern, which delegates requests across a chain of handlers, and the Template Method pattern, which defines the structure of an algorithm while allowing subclasses to refine specific steps. Together, these patterns highlight how behavior can be modularized to create adaptable and maintainable software systems.
Week 11: Software Testing Nov 18 and Nov 20, 2025	This week introduces the fundamentals of software testing with a focus on applying them to object-oriented design. Students will learn how to use GoogleTest in C++ for writing effective unit tests, including declarations, assertions, fixtures, and parametrized tests. They will also explore mocking and matchers as tools for isolating dependencies and verifying interactions. Finally, the discussion will connect testing practices to design patterns, highlighting both the challenges and goals of ensuring correctness, maintainability, and reliability in pattern-based software systems.
Week 12: Course Review Nov 25 and Nov 27, 2025	In the final week, students will review the key concepts, principles, and patterns covered throughout the course. The session will reinforce connections between requirements gathering, UML modeling, SOLID principles, design patterns, and testing, showing how they collectively support robust object-oriented design. Through discussion, recap exercises, and reflective activities, students will consolidate their understanding and prepare to apply these practices to real-world software development challenges.
Week 13: Industry Talk Dec 2 and Dec 4, 2025	

5. Course Materials

Course Reference Material

There are no required textbooks for this course. However, the following books are recommended as valuable references:

- **The C++ Programming Language, 4th Edition** by Bjarne Stroustrup
- **Programming Principles and Practice Using C++, 2nd Edition** by Bjarne Stroustrup
- **Design Patterns: Elements of Reusable Object-Oriented Software** by Erich Gamma et al.
- **UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition** by Martin Fowler
- **Object-Oriented Analysis and Design with Applications, 3rd Edition** by Grady Booch et al.

- **Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux** by Derek Molloy

Additional references will be provided as needed. Please check Brightspace for updates.

Lecture Notes and Brightspace

- Lecture materials will be available on Brightspace (Online Web Learning) prior to each class.
- Regular checking of the course site on Brightspace is essential for news and updates. It is the primary method of information dissemination for this course.
- Assistance with the site can be sought on the Brightspace help page or by contacting the Western Technology Services Helpdesk at phone number 519-661-3800 or ext. 83800.

Course Delivery and Format

- The course is planned to be delivered in-person for the entire term, subject to any unforeseen complications.
- Any changes aligning with University guidelines will be communicated through announcements via Brightspace.

6. Methods of Evaluation

Final Project (80%)

The final project requires students to design, model, and implement a non-trivial software system in C++ that demonstrates mastery of Object-Oriented Design and Analysis (OODA). **This project must be completed in pairs (two students only).** Students must perform requirements gathering, create UML models (Class, Sequence, and Use Case diagrams), and apply at least two creational design patterns (e.g., Singleton, Factory, Builder) along with one structural or behavioral pattern (e.g., Adapter, Observer, Strategy). The system must make correct use of core OOD principles—encapsulation, inheritance, polymorphism, and abstraction—and be implemented with clean, modular, and efficient C++ code. Deliverables include a project proposal, an intermediate design with partial implementation, and a final submission with a complete system, final UML diagrams, and a testing suite using GoogleTest and Google Mock. Detailed requirements, milestones, and evaluation criteria are posted on Brightspace.

Final Exam (20%)

The final exam focuses on refactoring C++ code using object-oriented principles, SOLID guidelines, and design patterns. Students will analyze code, apply targeted improvements, and justify their choices, with additional questions testing core C++ design and implementation concepts.

Due Dates of Deliverables

Deliverable	Due Date	Weight
Project Proposal	October 1, 2025	20%
Intermediate Design and Partial Implementation	October 30, 2025	30%
Final Project Submission	December 15, 2025	30%
Final Exam	December 10, 2025	20%

7. Student Absences

For the 2025-2026 academic year, the handling of absences in this course will follow Western University's updated Academic Consideration for Student Absences policy. The approach to academic considerations for each assessment type is outlined below:

Protected Course Components
In accordance with the policy, all group project submissions in this course are designated as *protected components*. This means that undocumented absences (self-reported absences up to 48 hours) cannot be applied to any part of the group project deliverables. Protecting groupwork ensures fairness to all group members and maintains the integrity of collaborative assessment.

Undocumented Absences
Because all major assessments in this course involve group submissions or the final exam, students cannot use any undocumented absences in this course. All documentation required for absences that are not covered by the Self-Reported Absence Policy must be submitted to the Academic Counselling office of a student's Home Faculty.

Project Proposal (20%) – No built-in flexibility
Extensions for this component will only be granted through formal academic consideration. If a student is unable to submit the proposal by the deadline, they must request an academic consideration through the appropriate channels.

Intermediate Design and Partial Implementation (30%) – Limited built-in flexibility
Students are allowed to use one late coupon to extend the deadline by up to 48 hours without needing documentation. Beyond this, academic considerations will be required.

Final Project Submission (30%) – No built-in flexibility
Due to the importance of the final project and the proximity to the course end date, this submission will not have built-in flexibility. Extensions will only be granted under formal academic consideration.

Final Exam (20%) – No built-in flexibility
The final exam must be completed as scheduled unless a formal academic consideration is granted. In such cases, a makeup exam will be arranged.

The new academic considerations policy allows students to self-report absences for up to 48 hours without documentation. For more details on how to request academic considerations, please refer to the full policy here:
https://www.uwo.ca/univsec/pdf/academic_policies/appeals/academic_consideration_Sep24.pdf.

8. Accommodation and Accessibility

Religious Accommodation

When a course requirement conflicts with a religious holiday that requires an absence from the University or prohibits certain activities, students should request accommodation for their absence in writing at least two weeks prior to the holiday to the course instructor and/or the Academic Counselling office of their Faculty of Registration. Please consult University's list of recognized religious holidays (updated annually) at

https://www.edi.uwo.ca/img/3754-2024-Diversity-Calendar-PDF.pdf

Accommodation Policies
Students with disabilities are encouraged to contact Accessible Education, which provides recommendations for accommodation based on medical documentation or psychological and cognitive testing. The policy on Academic Accommodation for Students with Disabilities can be found at

https://www.uwo.ca/univsec/pdf/academic_policies/appeals/Academic Accommodation_disabilities.pdf.

9. Academic Policies

The website for Registrarial Services is http://www.registrar.uwo.ca.

In accordance with policy,

https://www.uwo.ca/univsec/pdf/policies_procedures/section1/mapp113.pdf,

the centrally administered e-mail account provided to students will be considered the individual's official university e-mail address. It is the responsibility of the account holder to ensure that e-mail received from the University at their official university address is attended to in a timely manner.

Scholastic offences are taken seriously and students are directed to read the appropriate policy, specifically, the definition of what constitutes a Scholastic Offence, at the following Web site:

http://www.uwo.ca/univsec/pdf/academic_policies/appeals/scholastic_discipline_undergrad.pdf.

All required submissions may be subject to submission for textual similarity review to the commercial plagiarism detection software under license to the University for the detection of plagiarism. All papers submitted for such checking will be included as source documents in the reference database for the purpose of detecting plagiarism of papers subsequently submitted to the system. Use of the service is subject to the licensing agreement, currently between The University of Western Ontario and Turnitin.com (http://www.turnitin.com).

10. Support Services

Please visit the Science & Basic Medical Sciences Academic Counselling webpage for information on adding/dropping courses, academic considerations for absences, appeals, exam conflicts, and many other academic related matters: https://www.uwo.ca/sci/counselling/.

Students who are in emotional/mental distress should refer to Mental Health@Western (https://uwo.ca/health/) for a complete list of options about how to obtain help.

Western is committed to reducing incidents of gender-based and sexual violence and providing compassionate support to anyone who has gone through these traumatic events. If you have experienced sexual or gender-based violence (either recently or in the past), you will find information about support services for survivors, including emergency contacts at

https://www.uwo.ca/health/student_support/survivor_support/get-help.html.

To connect with a case manager or set up an appointment, please contact support@uwo.ca.

Please contact the course instructor if you require lecture or printed material in an alternate format or if any other arrangements can make this course more accessible to you. You may also wish to contact Accessible Education at

http://academicsupport.uwo.ca/accessible_education/index.html

if you have any questions regarding accommodations.

Learning-skills counsellors at the Student Development Centre (https://learning.uwo.ca) are ready to help you improve your learning skills. They offer presentations on strategies for improving time management, multiple-choice exam preparation/writing, textbook reading, and more. Individual support is offered throughout the Fall/Winter terms in the drop-in Learning Help Centre, and year-round through individual counselling.

Western University is committed to a thriving campus as we deliver our courses in the mixed model of both virtual and face-to-face formats. We encourage you to check out the Digital Student Experience website to manage your academics and well-being: https://www.uwo.ca/se/digital/.

Additional student-run support services are offered by the USC, https://westernusc.ca/services/.

11. Use of Generative AI Tools

This course embraces the use of generative AI tools (e.g., ChatGPT, Microsoft Copilot, Google Gemini, or similar platforms) as part of its AI-first approach to technology. Students are encouraged to thoughtfully and responsibly leverage such tools for learning, exploration, and course assessments (e.g., assignments, project work, coding support, summarization, and design assistance), provided that their use is clearly documented and acknowledged in submissions. Refer to individual project instructions for details.

Exception – Final Exam: The use of generative AI is strictly prohibited on the final exam. Exam performance must reflect each student's own independent understanding and ability to apply course concepts.

Academic Integrity Reminder:

- Any use of generative AI must be transparent, with clear attribution (e.g., noting prompts or tools used).

- Misrepresenting AI-generated content as entirely one's own work without disclosure will be treated as a scholastic offence under university policy.
- If uncertain about the appropriate use of AI for a specific assessment, students are responsible for seeking clarification in advance.