LEWIS UNIVERSITY
A Catholic and Lasallian University

Object-Oriented Programming -- CPSC 24500

Syllabus
Spring Semester 2026

I. Instructor Information

Instructor's name: ***David Nowak***

Lewis office location: AS 026L

Office hours:

Mondays: Noon through 12:45 pm, and from 3:00 to 4:00 pm
Wednesdays: Noon through 12:45 pm, and from 3:00 to 5:50 pm.

Arranged: Please email me to meet on other days/times and via Zoom.

How to make appointments outside of office hours: Please email me

Lewis office phone number: 815.836.5765

Lewis email address: dnowak2@lewisu.edu

II. Course Information

Course Name, Number, & Section:
Object-Oriented Programming – CPSC 24500 – 001 (11051)

Course Credit Hours: 3

Course description: Students will learn to design and develop software using the object-oriented approach. Topics include encapsulation, inheritance, polymorphism, abstraction, and patterns. Students will learn how to use an SDK to develop desktop and web applications that provide data processing and visualization services. Students will also learn how to manage threads and networking connections in software they write.

Prerequisite: CPSC 21000 – Programming Fundamentals

Professor's course description: Students will deepen their programming skills by applying object-oriented programming concepts in both Java and Python. Topics include encapsulation, inheritance, polymorphism, abstraction, design patterns, GUI programming, event handling, database interaction, serialization, and unit testing. Students will produce software individually and collaboratively, using Git for version control and UML for design documentation.

Course meeting times, days, and location:

- Monday & Wednesday from 2:00 to 2:50 am in person in room AS-104A.
- Friday from 10:00 to 10:50 am synchronously online via Zoom.

Note: Attendance is required for all sessions—both in-person and online.

Student Learning Outcomes:

Course student learning outcomes:

- *Solve problems by writing programs using standard language elements such as data declarations, arithmetic operations, conditional statements, loops, and functions.*
- *List and explain the key concepts of object-oriented development: inheritance, abstraction, information hiding, and polymorphism.*
- *Describe problems that typically plague software: rigidity, fragility, and immobility.*
- *Define the following object-oriented patterns: Factory, Singleton, Delegation, and Model-View-Controller.*
- *Define and provide examples for object-oriented design principles, and be able to describe SOLID: Single responsibility principle, Open-closed principle, Liskov substitution principle, Interface segregation principle, and Dependency inversion principle.*
- *Write class definitions and create objects from them.*
- *Declare and use special types of functions for classes, including constructors, accessors, mutators, and properties.*
- *Create hierarchies of classes that start with abstract base classes and add functionality in descendant classes.*
- *Design an object-oriented program in UML (Unified Modeling Language) that is organized around a set of classes whose objects interact.*
- *Describe what exceptions are and write programs that deal with them.*
- *Perform screen-scraping by retrieving data from a website.*
- *Write programs that use various collections.*
- *Use generic data types in programs.*
- *Work with collections of objects from related classes polymorphically.*
- *Explain the difference between classes and interfaces.*
- *Define interfaces that specify behaviors that particular objects must have.*
- *Perform input and output with text file streams.*
- *Perform input and output with XML file streams and serialization.*
- *Use an API as a reference when writing programs.*
- *Build attractive, intuitive graphical user interfaces.*
- *Write programs that use a graphical interface and manage user events using event-handling.*
- *Describe and use the client-server computing model*
- *Define serialization.*
- *Compare the advantages and disadvantages of various serialization sources and destinations.*
- *Write a program that stores and retrieves data with a relational database.*
- *Describe how Java achieves cross-platform compatibility.*
- *Distinguish between heavyweight and lightweight components.*
- *Define callback functions as they relate to event handling.*
- *Respond to user events in Java and Python.*
- *Describe how layout managers arrange components.*
- *Write unit tests to verify the correctness of software modules.*
- *Manage programming projects using GIT.*

Program student learning outcomes:

- *Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions.*
- *Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.*

Baccalaureate Characteristics/Graduate Student Learning Outcomes:

1. Essential Skills
6. Critical Thinking

III. University Mission Statement

Lewis University, guided by its Catholic and Lasallian heritage, provides to a diverse student population programs for a liberal and professional education grounded in the interaction of knowledge and fidelity in the search for truth.

Lewis promotes the development of the complete person through the pursuit of wisdom and justice. Fundamental to its Mission is a spirit of association, which fosters community in all teaching, learning and service.

How this course connects to the University Mission: *The Mission of Lewis University is to prepare lifelong learners who will use their knowledge, faith, wisdom, and talents to improve the lives of others. In our modern, digital world, there is no field that can as profoundly improve the lot of others than Computer Science. By taking this third course in the major, you will be moving much closer to your goal of becoming a world-improving computer scientist, one who can write efficient data-aware applications that inform.*

IV. Required Course Materials

Textbook(s):

For this course, we will use Open Educational Resources (OER) textbooks. These resources are free for you to use, which is important as no single book can provide the coverage we need for Java, Python, and object-oriented programming.

For each of the three books listed below, please note the abbreviation used, as the course schedule references the abbreviation.

- TP: Think Python: How to Think Like a Computer Scientist
- OOSC: Object-Oriented Software Construction
- JAVA: Introduction to Programming with JAVA

Supplemental readings, videos, online materials: In addition to our regular textbook readings, weekly supplemental readings will be assigned. All supplemental materials are found in Blackboard.

Hardware and software requirements:

- *General-purpose laptop/desktop*
- *Python 3.12 or higher (link to download here)*
- *Java JDK 22 (link to download here)*
- *GIT (link to download here)*

You will also need the following IDEs:

- *For Java, Eclipse (link to download here)*
- *For Python Visual Studio Code (link to download here)*

NOTE: Please use the above IDEs to ensure compatibility with everyone in the class.

V. Instructional Methods and Activities

Modality of Instruction: *Hybrid (Mon & Wed face-to-face; Fri synchronously online)*

VI. Course Schedule

Week *Dates* *(Sun to Sat)*	*Topics* *Mon/Wed = Lab & Review,* *Fri = Online Lecture*	Readings (Refer to the abbreviation code for specific book)		
		JAVA Reading	**TP** Reading	**OOSC** Reading
Week 1 *01/18 – 01/24* *Java*	*Syllabus, Git/GitHub workflow, review of Java syntax from fundamentals, intro to classes & objects*	Unit 1: Getting Started	N/A	
Week 2 *01/25 – 01/31* *Java*	*Data types, constructors, accessors, mutators, properties; UML basics for class diagrams*	Unit 2: Using Objects Unit 3: If Statements	N/A	Ch 1: Software Quality
Week 3 *02/01 – 02/07*	*Conditionals, loops, exceptions in OOP; Javadoc* *Mon: Feb. 2 – No Class - Instructor Meeting*	Unit 4: Iteration/Loops Unit 5: Writing Methods	N/A	
Week 4 *02/08 – 02/14* *Java*	*File I/O with text files; collections overview (ArrayList, HashMap)*	Unit 6: Writing Classes	N/A	Ch 2: Criteria of Object-Orientation
Week 5 *02/15 – 02/21* *Java*	*Generics & type safety; designing our own collection class*	Unit 7: Arrays Unit 8: ArrayLists	N/A	
Week 6 *02/22 – 02/28* *Java*	*Inheritance, abstraction, interfaces; Liskov Substitution & Open-Closed Principles*	Unit 9: 2D Arrays Unit 10: Inheritance	N/A	Ch 3: Modularity

Week 7 *03/01 – 03/07* *Java*	*Polymorphism & dependency inversion; working with collections of related objects polymorphically* *Mon: Mar. 2 – No Class – Instructor Meeting*	Unit 11: Recursion	N/A	
Week 8 *03/08 – 03/14* *Java*	*Object-oriented design patterns (Factory, Singleton, Delegation, MVC)*	N/A	N/A	Ch 4: Approaches to Reusability
03/15 – 03/21	*No Classes this Week - Spring Break*			
Week 9 *03/22 – 03/28* *Python*	*Switch to Python: Python OOP syntax, differences from Java, properties, decorators; reimplement a Java example in Python*	N/A	Ch. 14: Classes and Functions Ch. 15: Classes and Methods	
Week 10 *03/29 – 04/04* *Python*	*Exceptions in Python; working with APIs* • *Mon: Mar 30 – No Class – Instructor Meeting* • *Wed: Apr 1 Class Online* • *Fri: Apr 3 – No Class (Easter Break)*	N/A	Ch. 9 Lists Ch. 16: Classes and Objects	Ch 5: Towards Object Technology
Week 11 *04/05 – 04/11* *Java & Python*	*Serialization in Java & Python (XML, JSON, Pickle); compare approaches* *Mon Apr 6 – No Class (Easter Break)*	N/A	Ch. 17: Inheritance	
Week 12 *04/12 – 04/18* *Java & Python*	*Gang of Four Design Patterns*	N/A		Ch 6: Abstract Data Types
Week 13 *04/19 – 04/25* *Java & Python*	*GUI development basics (Java Swing vs Python Tkinter)* *Composition over Inheritance*	N/A	Supplement with online Tkinter docs	
Week 14 *04/26 – 05/02* *Java & Python*	*Databases in Java (JDBC) & Python (sqlite3); ORM intro* *Refactoring*	N/A	Python DB API docs (supplemental)	
Week 15 *05/03 – 05/09*	*Unit testing in Java (JUnit) & Python (pytest/unittest); project presentations*	N/A		

Java & Python	*Scheduled Oral Exams* *Mon: May. 4 – No Class – Instructor Meeting*			
Week 16 *05/10 – 05/16*	*Final Project, Reflection + Scheduled Oral Exams* *No Classes this Week – Finals Week*			

Schedule Changes: the above schedule is subject to change; I reserve the right to make modifications based on class progress and student feedback. Any modifications to the above schedule will be discussed and communicated verbally, via a Blackboard announcement, and email.

VII. Grading Criteria and Course Policies

Assignments and Course Requirements:

Quizzes: Your knowledge will be tested throughout the semester, based on the materials covered in the books, lectures, and programming problems you have worked on. These quizzes will be on Blackboard and taken during the week assigned outside of the classroom.

Programming Assignments: The purpose of programming assignments is for students to practice writing programs (both short code snippets and longer programs) that are efficient and easy to read. To facilitate grading, all code must be submitted as source files via GitHub--ONLY. I will not accept zip files or screen captures in lieu of the actual code.

Strive for excellence, but don't let the pursuit of perfection constrain your ability to submit material on time. I am looking for well-designed and well-written code: code that conforms to typical conventions for the language you are using, that is easy to read, and that demonstrates the use of best programming features for the job. Use classes and exploit inheritance when you can. Use appropriate data structures for the work to be performed, and use meaningful variable names. All submissions must be made using Blackboard; email submissions will not be accepted. Furthermore, our email system typically strips code files out of sent emails.

Students will work individually and in groups.

Discussion Boards: Regularly throughout the semester we will have discussion board assignments. These assignments might be related to a current technology news event; they may relate to the most recent programming assignment. I often incorporate reflection writing as part of the work we do in the classroom. All discussion board assignments count as homework.

Surveys and Polls: I regularly request you complete short surveys. These allow me to get a pulse in the class and how people are feeling about the work and their learning. Because these are important to me, they factor into homework assignments for you. Some will allow you to remain

anonymous. In such cases, you will submit a screen capture of the final page demonstrating you completed the survey or poll.

Presentations and Group Work: There will be times when you are required to contribute to a group project. One of the deliverables from each person in the group will be a sheet assessing the contributions of fellow teammates. If multiple assessments indicate a lack of contribution by a particular member, that person will receive a zero for the project. You are responsible to one another in a group setting, and every person must contribute.

Important: Throughout the semester, I will ask students to present their work. These usually will be in the form of code reviews, which is a common occurrence with software development teams. These will consist of a short presentation of specific code, a possible demonstration of the running program, and a short Q&A session from the class. Presentations and group work all count as homework assignments.

Important: Presentation of code (code reviews) is a standard practice in my class—because it's a standard practice in the professional software development community. Like other activities in my class, these code reviews count as assignments.

Course Grade:
In this course I use contract grading: instead of chasing points, you earn your grade by meeting clearly defined requirements. The baseline contract (base grade) guarantees a minimum of a B for students who complete all assigned homework on time and at an acceptable level, as defined by each assignment's specifications and rubric. Work is evaluated as "meets spec" or "revise and resubmit" until it meets the stated standard (within the limits of the late and revision policies). Quizzes, short tests, programming assignments, surveys, and discussion posts occur throughout the term to support your learning; you are expected to complete them as scheduled, and they factor into eligibility to claim the contract you aim to fulfill.

Concerning attendance, you are allowed six unexcused absences (equivalent to two weeks) from my class and maintain your B grade (these are in addition to the regularly scheduled days off this semester). Seven through ten absences will drop your base grade to a C. Eleven or more absences will cause your base grade to drop to a D.

To earn a higher grade than the base grade requires one or both of the following final assignments:

1. A final project demonstrating skills acquired through the semester. You will have several choices of final project.
2. A scheduled 30-minute conversation (oral exam) with your professor.

Each of the above two (optional) final assignments will raise your base grade by a half-letter grade. Successfully passing both final assignments above will raise your base grade by one full letter grade. Hence, if you earned the contract B grade, successfully completing the project and the oral exam will raise your grade to an A.

If, because you did not submit all work or there are excessive absences, you earn a C base grade, successfully completing both final assignments will raise your grade to a B.

Students who do not complete all homework at the acceptable standard, who miss required activities, or who opt out of the final activities receive a grade below A and possibly below B, depending on how much of the baseline contract is unmet. Your current standing will be tracked transparently, so you always know what remains to earn the grade you're targeting.

Total Assignments: Because this course uses a contract-grading scheme—and because some concepts may come quickly while others require more time—I will not post a complete assignment list in advance. Expect a mix of short, in-class tasks and longer projects that span several class sessions or weeks; the sequence may adjust as we progress, and your final grade will be determined by the contract you fulfill.

Grading Policies: *All assignments will be submitted on time unless prior arrangements for severe circumstances (before the due date) are made.*

Attendance is required, and excessive absences will negatively impact your grade (see requirements under *Course Grade* above).

Late and Revision Policies:
Due to my contract grading policy, you must submit work on time. Furthermore, we discuss the assignments in our class, and it would be unfair to students who turn in work on time to allow others to submit after we have thoroughly examined it. If you must be late submitting your work, it is your responsibility to inform me before the work is due. In no case will assignments submitted later than 48 hours after the due date be accepted.

I may elect not to accept work that does not "meet spec." In such cases, you have one week from the date of my grading decision to resubmit work that meets the specifications for that assignment. If you choose not to revise and resubmit, the work will be treated as missing for contract grading purposes.

Course Policies: Let us remember that we are in the holy presence of God. May we accept ourselves and one another with grace, kindness, humility, and patience.

Changes to Course Assignments or Grades: Any changes to the grading or course policies will be discussed and communicated verbally, via a Blackboard announcement, and email.

Plagiarism: Copying from another student, or uncredited reuse from another student (current or former) or from any online source—will result in your submission being rejected with no opportunity to revise. If you use ChatGPT or any generative-AI tool, you must clearly identify the AI-generated portions and provide a detailed explanation of what that code does and how it works. Failure to meet both requirements will result in the assignment being recorded as missing. These standards will be enforced strictly and without exception.

VIII. Academic Information for Students

Coursera Career Academy:
Students at Lewis University have open access to industry-recognized courses and certificates from leading employer. Through this collaboration, our students have access to over 50 certification programs developed in collaboration with industry leaders, including Microsoft, Google, Meta, and IBM. The certifications are an opportunity to supplement your current curricula and learn additional job-relevant skills that today's top employers are looking for. Click on this link for more information and to start your Coursera Career Academy journey.