Object-Oriented Programming
Course Syllabus

Course & Instructor Information

Course:	Lewis University, Object-Oriented Programming, CPSC-24500-LT1, Online, Spring 2018
Dates:	Tuesday, 16 January 2018 through 12 May 2018 per Academic Calendar [link]
Times:	TTh 2:00 to 3:15pm CST
Location:	AS 104A
Final:	Tuesday, 8 May 2018 from 1:30 to 3:30pm in our normal location (AS104A)
Instructor:	Eric Pogue
Email:	epogue@lewisu.edu …you can expect a response within 24 hours weekdays
Phone:	563-209-7280 (personal mobile)

Object-Oriented Programming Course Description

Students will learn to design and develop software using the object-oriented programming (OOP) techniques. Topics include encapsulation, inheritance, polymorphism, abstraction, and patterns. Students will learn how to use and libraries and software development kits (SDKs) to develop applications that provide data processing and visualization services. Students will also learn how to manage threads and networking connections in software they write.

In addition, students will learn the basics of developing software utilizing Agile processes, terminology, and development techniques.

Prerequisites:	CPSC 21000 Programming Fundamentals or consent of instructor
Credits:	3

Course Materials

Reference materials for this course will focus on class slides, notes, and online sources.

Textbook:	No textbook is required for this course.
Reference Materials:	Class slides, notes, and online sources will serve as your primary reference materials. These will be posted online and linked via Blackboard. New materials will be made available at one and two-week intervals.
Recordings:	Lecture sessions and important topics will be recorded and posted online… as long as this practice does not have a negative impact on class attendance or participation.

Student Learning Outcomes

On the successful completion of this course you will be able to:

- Understand the basics of various software delivery process and their implications for utilizing object-oriented (OOP) programming techniques
- Utilize Agile software delivery techniques and terminology to deliver assignments
- Effectively utilize key developer tools with an initial focus on Git and GitHub
- 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
- Write class definitions and create objects from them
- Declare and use special types of functions for classes, including constructors, accessors, and properties
- Define the following object-oriented patterns: Factory, Singleton, Delegation, and Model-View-Controller
- Describe problems that typically plague software: rigidity, fragility, immobility
- Define and provide examples for object-oriented design principles: Liskov Substitution Principle, Dependency Inversion Principle, Interface Segregation Principle, Open-Close Principle, Single-Responsibility Principle
- Create hierarchies of classes that start with abstract base classes and add functionality in descendant classes.
- Design an object-oriented program utilizing the Unified Modeling Language (UML) to describe a set of classes whose objects interact
- Describe what exceptions are and write programs that deal with them
- 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 certain objects must have
- Perform input and output with text file streams
- Perform input and output with JSON (or possibly 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
- Write a program that stores and retrieves data with a relational database
- Create a user interface in Java and C# (possibly C++)
- Describe how Java achieves cross-platform compatibility
- Distinguish among heavyweight and lightweight components
- Define callback function as it relates to event handling
- Respond to user events in Java and C# (possible C++)
- Describe how layout managers arrange components
- Understand how to write unit tests to verify the correctness of software modules

Course Outline with Weekly Student Learning Outcomes:

Weeks 1 & 2

- Define object-oriented programming
- Distinguish between an object and a class
- Describe how object-oriented programming is fundamentally different from the programming you have done before
- Justify the choice an object-oriented approach to developing software
- Identify and define six object-oriented concepts: abstraction, encapsulation, information hiding, ownership, inheritance, and polymorphism
- Distinguish between aggregation and composition as two different kinds of ownership
- Identify the superclass and the subclass in an inheritance relationship
- Demonstrate inheritance, ownership, and polymorphism in snippets of Java code
- Depict classes and their relationships using UML class diagrams
- Define and demonstrate the software patterns delegation, singleton, factory, and model-view-controller
- Explain through examples the following software design principles: Open-Closed, Dependency-Inversion, Interface Segregation, Liskov Substitution, and Single Responsibility
- Identify and describe characteristics of bad software: rigidity, immobility, fragility

Week 3 & 4

- Write definitions for classes that comprise the model of a model-view-controller application
- Use inheritance to build classes that expand upon the capabilities and features of classes that came before
- Declare an abstract class and explain why creating abstract classes is useful
- Implement information-hiding by adding private data members to your model classes and including public get and set functions for accessing their values
- Write default and non-default constructors for a model class
- Redefine the toString function a class inherits from the base java.lang.Object class to represent the object as a String
- Distinguish between function overloading and function overriding
- Create objects that are instances of a model class
- Store multiple objects in a Java collection called an ArrayList
- Define what generic means as applied to a data type
- Distinguish between using an array and an ArrayList to store a collection of objects
- Use Javadocs to provide attractive and comprehensive documentation for your code
- Work with a collection of related objects polymorphically
- Explain how polymorphism is implemented behind the scenes

Week 5 & 6

- Explain the difference between lightweight and heavyweight components
- Identify lightweight and heavyweight components
- Describe how lightweight and heavyweight components render themselves
- Explain what a layout manager does and identify and describe three of them
- Specify layout managers for heavyweight and lightweight components

- Use paint and paintComponent's Graphics object to draw a variety of shapes
- Explain how to achieve true Model-View-Controller architecture by removing any reference from the view to the model and from the model to the view
- Distinguish between extending a class and implementing an interface
- Create an event handler that implements the ActionListener interface to respond to the user clicking on a button
- Explain multiple ways to implement an event handler for a particular object and event (anonymous inner classes vs. named classes vs. having the frame itself serve as the handler)

Weeks 7 & 8

- Create interactive applications that adhere to the Model-View-Controller pattern
- Write code that responds to a variety of events, including clicking the mouse on a frame or panel, moving or dragging the mouse, and typing a key on the keyboard
- Implement animation using a timer and a corresponding event handler.
- Build a menu system that enables the user to trigger a variety of actions in a familiar way.
- Create multiple, intuitive ways for a user to perform a particular task.
- Design and implement a controller class that outputs data to a text file.
- Design and implement a controller class that inputs data from a text file and builds a collection of objects from the read data.
- Describe how Java achieves speedier input and output through its hierarchy of input and output classes.

Week 9 & 10

- Explain why software testing is important for modern applications.
- Define various terms related to the process of software testing.
- Distinguish between the various kinds of testing: unit testing, integration testing, and performance testing.
- Compare testing state with testing behavior.
- Explain the purpose of annotations in Java source code.
- Explain the purpose and syntax of the various assert statements junit supports.
- Install junit onto your machine and make it available in a Java project you write.
- Write and execute a junit test for a simple application you write.
- Write javadoc comments that describe the purpose, inputs, and outputs of classes and their components.
- Build javadoc documentation using Java's javadoc command-line tool.
- Identify reasons to use a Java package.
- Explain what a Java package is in terms of both how it is used and how it is stored.
- Compile a class from the command line so that it belongs to a particular package.
- Import a class you write that belongs to a particular package.
- Identify reasons for using jar files to group together related java classes.
- Create a jar file that stores the contents of a particular package.
- Provide access to the classes you've added to a jar file in an application you write.

Week 11 & 12

- Identify characteristics of C# (possibly Python) that distinguish it from other languages.

- Work with C# lists.
- Write non-object-oriented programs that use sequence, selection, and repetition to accomplish tasks.
- Write C# functions and place those functions in separate libraries.
- Define a C# class, complete with data members, functions / methods, and an initializer (constructor).
- Use C# built-in text file objects to create and read text files.
- Use inheritance to create a hierarchy of classes that are related to each other.
- Create objects of classes and use them to carry out the work of your program.
- Deal with a list of related objects polymorphically.

Week 13 & 14

- Design and implement a Model-View-Controller application in C#.
- Perform basic drawing operations using a popular C# drawing package.
- Implement input and output with plain text files using a controller class designed for that purpose.
- Separate code among various libraries that you can reuse in other applications by importing them.
- Implement information hiding in C# using both accessor and mutator functions and properties.
- Download documents from remote servers from your application.
- Parse data expressed in XML format.
- Separate an application's functionality among classes having well-defined function sets and purposes.

Week 15 & 16

- Design, build, and query a simple database using SQL
- Write a class to query a SQL database in C#.
- Encapsulate the data returned by an SQL query in lists of objects so that your program can process them.
- Program the interaction of a client and server at the socket level.
- Describe what a thread is and why it can be useful to distribute tasks among multiple threads
- Write a multi-threaded application in C#.
- Explain why it is important to synchronize threads that need to share access to data sources

Class Assignments

Assignments for this course will take the form of quizzes, labs, programming projects, and participation.

Tests:	There will be no midterm or final tests for this course.
Quizzes:	Quizzes will be assigned at the end of one or two-week intervals.
Programming Projects:	Programming projects are the most important element of this class and will be assigned to be delivered in one or two-week intervals.
Participation:	Participation and engagement can be demonstrated in class, via Blackboard discussions, and through your class presentations/demos.

Programming projects are the most essential element of this course and will account for approximately 60% of the final grade. Leaving quizzes and participation to account for approximately 30% and 10% respectively.

Late Assignments & Partially Completed Assignments

Late assignments will not be excepted except under extreme circumstances. It is vastly preferable to turn in a partially complete assignment than to turn in a late one.

Similarly, it is vastly more beneficially to turn in an assignment that has 70% of the features working 100% than to turn in an assignment that has 100% of the features working 70%.

Grading Policies

Final course letter grade will be determined using the following approximate scale:

A	>= 90	C-	70-72.99
B+	87-89.99	D+	67-69.99
B	83-86.99	D	63-66.99
B-	80-82.99	D-	60-62.99
C+	77-79.99	F	< 60
C	73-76.99		

Drop and Withdrawal Deadlines

Information regarding important drop and withdrawal dates and policies can be found on the Bursar's webpage [link].

Plagiarism:

Copying work from each other or from the Internet will be punished harshly and appropriately. This includes viewing test, quiz, or assignments from current or previous class sessions. Measure of Software Similarity (MOSS) or similar software may be utilized to detect copied work. If it is determined that you have copied/plagiarized your work, you will fail the assignment and may have addition points deducted from your grade equal to the assignment maximum. If you do this twice, you will fail the course and potentially face additional disciplinary action. Integrity is expected and required.

Also see "Academic Honesty" below.

Closing Comments:

Nearly all modern, sophisticated software is created utilizing object-oriented programming techniques. I hope that this course will allow you to dramatically improve your programming skills and capabilities in

the coming weeks. Please make sure you let me know when you're struggling so that I can help you be successful. Finally, be sure to take a moment and enjoy the journey.

Connection to 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. The Lewis University Mission Statement can be found online [link].

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.

Class Attendance Policy

Students are expected to attend all classes as part of the normal learning process. Students bear the ultimate responsibility for all missed class material as the result of an absence and can be required to make up any work missed.

Students must be especially consistent in attendance, both on-ground and online, during the first two weeks of the class to confirm registration and to be listed on the official course roster. Students who fail to attend the first two weeks and who have not received prior approval from the instructor for absences will be withdrawn from the courses in question by certification of the instructor on the official class lists.

Additional information relating to Class Attendance Policy is available online [link].

Academic Honesty

Scholastic integrity lies at the heart of Lewis University. Plagiarism, collusion and other forms of cheating or scholastic dishonesty are incompatible with the principles of the University. Students engaging in such activities are subject to loss of credit and expulsion from the University. Cases involving academic dishonesty are initially considered and determined at the instructor level. If the student is not satisfied with the instructor's explanation, the student may appeal at the department/program level. Appeal of the department /program decision must be made to the Dean of the college/school. The Dean reviews the appeal and makes the final decision in all cases except those in which suspension or expulsion is recommended, and in these cases the Provost makes the final decision.

- Policies regarding make-up examinations and late submission of assignments
- Drop and withdrawal deadlines (see semester Course Schedule)
- Classroom behavior expectations (consistent with "Classroom Decorum" statement from Student Handbook on page 14 [link]

Classroom Decorum

In order to maintain an environment conducive to learning and student development, it is expected that classroom discourse is respectful and non-disruptive. The primary responsibility for managing the classroom environment rests with the faculty. Students who engage in any prohibited or unlawful acts that result in disruption of a class may be directed by the faculty member to leave class for the

remainder of the class period. Students considered to be a disruption or who present a threat of potential harm to self or others may be referred for action to the Dean of Student Services.

Additional information relating to Classroom Decorum is available online in the Student Handbook [link].

Sanctified Zone

This learning space is an extension of Lewis University's Sanctified Zone, a place where people are committed to working to end racism, bias and prejudice by valuing diversity in a safe and nurturing environment. This active promotion of diversity and the opposition to all forms of prejudice and bias are a powerful and healing expression of our desire to be Signum Fidei, "Signs of Faith," in accordance with the Lewis Mission Statement. To learn more about the Sanctified Zone, please visit online [link].

Students Requiring Special Accommodations

Lewis University is committed to providing equal access and opportunity for participation in all programs, services and activities. If you are a student with a disability who would like to request a reasonable accommodation, please speak with the Learning Access Coordinator, Angelia Martinez, at the **Center for Academic Success and Enrichment (CASE)**. Please make an appointment by calling **815-836-5593** or emailing learningaccess@lewisu.edu. For more information about academic support services, visit the website at: www.lewisu.edu/CASE. Since accommodations require early planning and are not provided retroactively, it is recommended that you make your request prior to or during the first week of class. It is not necessary to disclose the nature of your disability to your instructor.

Additional Policy & Guideline Resources

Additional policy and resources can be found at the links provided below.

University Student Complaint Policy [link]

University Grade Appeal Policy [link]

University Copyright and Intellectual Property Guidelines [link]