Object-Oriented Design and Programming II

CS 262 - Spring 2020

Instructor: Hannah Hillberg
Email: hillbergh@uwosh.edu
Office: Halsey 217
Office Hours: 2:00-3:30pm MW
4:30-5:30pm Tu
9:15-10:15am F

Section:	001C		
Lecture:	MTW	11:30am-12:30pm	Halsey 208
Lab:	F	11:30am-12:30pm	Halsey 101C

Course Information
A second course in problem solving, software design, and computer programming using an object-oriented language. Problem solving/software design topics include: abstract data types, universal modeling language (UML), simple recursion, unit testing, event-handling, simple concurrency. Data structures and algorithms include: binary search, simple sorting algorithms, use of collection classes and their iteration protocols, sequential file processing. Additional topics include: inheritance, polymorphism, graphical user interfaces, simple use of threads. **Credits:** This is a 4 credit course.
Prerequisites: Math-108 or equivalent with a grade of C of better, or qualifying for a higher level mathematics via the Mathematics Placement Test, and CS 221 or equivalent with a grade of C or better.

Course Website: UWO Canvas
You should check Canvas on a regular basis - it will contain lecture notes, assignments, announcements, and grades. I'll do my best to let you know when something new and important comes up, but it is your responsibility to check the website frequently for information that you might not get otherwise.

Required Textbook:
COMP SCI 262: Object Oriented Design and Programming II, Online book by Zybooks.
Subscription Instructions:
1. Sign in or create an account at learn.zybooks.com. You are required to use your uwosh email as your login. You may need to also provide your ID number.
2. Enter zyBook code: UWOSHCOMPSCI262HillbergSpring2020
3. Subscribe. A subscription is $58 and will last until May 29, 2020. Students will be able to subscribe until May 3, 2020. **Note:** If you had a previous subscription (for CS 221), you should automatically get a 50% discount, so your subscription should be $29. Be sure to let me know if you do not receive the discount but I believe you should.

Course Grading Policy
Your final grade for this course will be based on four components, namely exams, programming projects, labs, and participation and challenge activities. Your overall numerical grade for the course will be computed as the weighted sum of the component grades using the following weights:

Component	**Weight**
Exams (3)	45%
Programming Projects	30%
Labs	20%
Participation and Challenge Activities	5%

Grading will be on a plus/minus system. Grading may be done on a curve depending on the overall performance of the class. If no curve is used, your grade will be computed as follows:

Numerical Score	Grade	Numerical Score	Grade
>=92	A	72-78	C
90-92	A-	70-72	C-
88-90	B+	68-70	D+
82-88	B	62-68	D
80-82	B-	60-62	D-
78-80	C+	<60	F

While this overall grading scheme is fixed, I will be happy to discuss any issue you may have with individual grades. If you notice a mistake or have a question regarding a specific grade, please come and talk to me as soon as possible (within one week of the received grade). Do not wait until the end of the semester to bring up grading issues.

***Tentative* Exam Dates:**
- Exam 1 – Wednesday, March 4
- Exam 2 – Wednesday, April 15
- Exam 3 – Wednesday, May 13

If you are unable to take a scheduled exam, it may be possible to take a make-up exam provided that you do both of the following, which are then subject to my approval:

1. Make arrangements prior to the scheduled exam. For last minute emergencies, email me immediately. No after-the-fact notifications will be accepted.
2. Have a written medical excuse signed by the attending physician OR have a note of justification from the Dean of Students Office.

Deadlines and Late Days
Each lab and participation activity will come with a deadline by which it must be submitted. Late lab or participation activity (zyBook assignment) submissions will NOT be accepted.

Each project will also come with a deadline (day and time) by which it must be submitted. You are allotted **three *project credit days*** that you can use to submit a project after its deadline without penalty. A credit day is exactly 24 hours or less. Please keep in mind that three days is not much, so plan to hit the deadlines and save the late days for real emergencies. Any project submitted after the deadline, plus any credit days you have unused, will receive a zero.

Coding Standards
In this class, you will write several short to medium-length Java programs. One of your goals (during this class and beyond, in Java or any programming language) should be to write understandable, readable code. You should be making every effort to document anything that might be confusing to a reader unfamiliar with your program (using correct spelling and grammar), to name variables intelligently, to use indentation that reflects the code's organization, and so on. All of this will be taken into account during grading: poorly organized or written code may have a negative impact on your grade, even if the resulting program works fine. Briefly, your code must be formatted in a consistent and easily-readable manner. At a minimum, I will require that you utilize the auto-format feature of whatever development environment we use (e.g., “Auto-layout” in BlueJ).

One of the goals of this class is to teach you to write functioning programs in Java. Thus, your code must compile and run correctly in order for you to receive full credit. Code that does not compile will receive at most 50% credit (or less depending on completion). Keep this in mind when writing programs: write your code in small pieces, making sure each piece works before moving on to the next one. It is much better to turn in a project that is not finished but has many working pieces than to turn in one that doesn't work at all, even though most of the code is written.

All assignments and labs must be submitted electronically via Canvas. It is your responsibility to ensure that your assignment or lab was submitted correctly (double check!).

Academic Dishonesty
Academic dishonesty of any kind will not be tolerated. All assignments, labs, and exams are to be completed individually. While discussion of ideas and problems with fellow students is encouraged, all work must be done individually. In certain circumstances, code fragments from the instructor may be provided to eliminate tedious coding or to provide a common

framework for all students. All other code must be original. Online resources may be used to help you understand the material, but you may not copy online code nor can you "borrow'" code from other students, past or present. If you use a book, website or any reference to help you solve a problem, you must cite the reference in your assignment.

Any suspected academic dishonesty will be dealt with on a case-by-case basis. Any clarification of what does or does not constitute academic dishonesty must take place *before* you turn in questionable work. For clarification on what constitutes academic dishonesty, contact me or consult the printed policy in the UWO Student Discipline Code, Chapter UWS 14.

Accessibility
It is the University's policy to provide, on a flexible and individual basis, reasonable accommodations to students who have documented disabilities that may affect their ability to participate in course activities or to meet course requirements. Students with disabilities must be registered with the Accessibility Center and receive an Accommodation Recommendation form to receive accommodations. The Accessibility Center is located in Dempsey Hall 125.

It is also the policy and practice of UW Oshkosh to create an inclusive learning environment. If there are aspects of the instruction or design of this course that result in barriers to your inclusion, please notify me as soon as possible. You are also welcome to contact the Accessibility Center at (920) 424-3100 or accessibilitycenter@uwosh.edu. For more information, visit the Accessibility website at http://www.uwosh.edu/deanofstudents/accessibilitycenter.

Course Outcomes
At conclusion of the course, students will be able to:

1. Debugging Java programs with BlueJ -- You will be expected to:
    a. Analyze a program on its correctness
    b. Identify software bugs with the debugger in BlueJ
2. Objects and Classes -- You will be expected to:
    a. Specify a class with the UML graphical notation
    b. Use the UML graphical notation to describe classes
    c. Distinguish between object reference and primitive data type variables
    d. Apply classes in the Java API (Application Programming Interface)
    e. Differentiate between instance and static variables
    f. Develop methods in classes
    g. Store and process objects in arrays
    h. Apply class abstraction to develop software
3. Inheritance and Polymorphism -- You will be expected to:
    a. Develop a subclass from a superclass through inheritance
    b. Apply the polymorphism concept to handle different data types using a uniform interface
4. Abstract Classes and Interfaces -- You will be expected to:

    a. Identify the similarities and differences between an abstract class and an interface
    b. Model weak inheritance relationships with interfaces
    c. Specify a natural order using the Comparable interface
    d. To wrap primitive data values into objects
    e. Create a generic sort method
    f. Simplify programming using JDK 1.5 automatic conversion between types and wrapper class types
5. Exceptions and Assertions -- You will be expected to:
    a. Distinguish exception types: Error version Exception in Java.
    b. Throw an exception in a method
    c. Write an exception handler using a try-catch-finally block
    d. Explain the propagation of an exception
    e. Apply assertions to help ensure program correctness
6. Text I/O -- You will be expected to:
    a. Read and write characters using the InputStreamReader, FileReader, BufferedReader, OutputStreamWriter, FileWriter, PrintWriter and BufferedWriter classes.
    b. Be able to apply the appropriate class in text I/O operations based on the requirements and performance needs
    c. Distinguish between text I/O and binary I/O
7. Object-Oriented Design -- You will be expected to:
    a. Become familiar with the software development process
    b. Model a system with the appropriate relationships: association, aggregation, composition, dependency, strong inheritance and weak inheritance
    c. Declare classes to represent the relationships among them
    d. Design systems by identifying the classes and discovering the relationships among these classes
8. Unit testing with JUnit -- You will be expected to:
    a. Create test classes, test methods and run tests with JUnit
    b. Create and use test fixtures in JUnit
    c. Interpret test results with JUnit
    d. Correlate the test fixtures with assertions
    e. Verify that a software unit performs as specified
9. GUI and Graphics -- You will be expected to:
    a. Describe the Java GUI hierarchy
    b. Create user interfaces using frames, panels and simple GUI components
    c. Apply layout managers
    d. Use JPanel as sub-containers.
    e. Draw figures using the methods in the Graphics class.
    f. Override the paintComponent method to draw figures on a GUI component
    g. Introduction to Threads -- creation, simple usage
10. Event Driven Programming -- You will be expected to:
    a. Declare listener classes and write event handlers to handle events

    b. Apply the Observer pattern to decoupled programs
    c. Register listener objects in the source object
    d. Create inner classes and anonymous inner classes
    e. Write programs to handle ActionEvent, MouseEvent, KeyEvent and Timer events.
11. Recursion -- You will be expected to:
    a. Solve problems with recursion
    b. Write programs using recursion
    c. Explain the difference between iteration and recursion
12. Generic Types -- You will be expected to: Improve reliability and readability of Java programs by using generic types
13. Java Collections Framework -- You will be expected to:
    a. Describe the Java Collections framework hierarchy
    b. Utilize the common methods in the Collection interface for operation sets and lists
    c. Utilize the Iterator interface to traverse a collection
    d. Examine the Set interface and be capable of deciding when to use HashSet, LinkedHashSet or TreeSet to store elements
    e. Compare elements using the Comparator interface
    f. Examine the List interface and be capable of deciding how and when to use ArrayList or LinkedList to store elements
    g. Examine the Collection and Map and be capable of deciding how and when to use HashMap, LinkedHashMap and TreeMap to store values associated with keys.