Object-Oriented Design and Programming I

CS 221 - Spring 2019

Instructor:	Hannah Hillberg
Email:	hillbergh@uwosh.edu
Office:	Halsey 217
Office Hours:	3:00-4:30pm MW
	3:00-5:00pm Th

Section:	001	
Lecture:	MW	10:20-11:20am Halsey 208
Lab:	F	10:20-11:20am Halsey 101C

Course Information
A first course in problem solving, software design, and computer programming using an object-oriented language. Problem solving/software design techniques include: flow charts, pseudo code, structure charts, and UML class diagrams. Data structures and algorithms include: arrays, characters strings, Linear search. Programming topics include: data types, assignment statements, standard input/output, selection, repetition, functions/methods, parameters, scope of identifiers, debugging.

Credits: This is a 3 credit course.
Prerequisites: A grade of C or better in Math 104 or Math 108 or Math 206 or Computer Science 142, or qualifying for Math 171 via the Mathematics Placement Exam.

Course Website: UWO D2L (http://www.uwosh.edu/D2L)
You should check D2L on a regular basis - it will contain lecture notes, handouts, 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 web site frequently for information that you might not get otherwise.

Required Textbook:
COMP SCI 221: Object-Oriented Design and Programming I, Online book by Zybooks.
Subscription Instructions:
1. 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: UWOSHCOMPSCI221HillbergSpring2019
3. Subscribe. A subscription is $58 and will last until May 30, 2019. Students will be able to subscribe until May 4, 2019.

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	25%
Labs	20%
Participation and Challenge Activities	10%

Your letter grade for the course 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. Do not wait until the end of the semester to bring up grading issues.

***Tentative* Exam Dates:**
• Exam 1 – Wednesday, March 6
• Exam 2 – Wednesday, April 17
• Exam 3 – Wednesday, May 15

Deadlines and Late Days

Each lab, participation activity and challenge activity will come with a deadline (day and time) by which it must be submitted. Late lab, participation activity or challenge activity 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.

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 projects and labs 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.

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 Services for Students with Disabilities or Project Success and receive an Accommodation Recommendation form to receive accommodations. Services for Students with Disabilities is located in 125 Dempsey Hall.

It is also the policy and practice of UW Oshkosh to create inclusive learning environments. 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 Services for Students with Disabilities at 920-424-3100 or dean1@uwosh.edu. For more information, visit the Services for Students with Disabilities website at http://www.uwosh.edu/deanofstudents/disability-services.

Disclosure Statement

Students are advised to see the following URL for disclosures about essential consumer protection items required by the Students Right to Know Act of 1990: https://uwosh.edu/financialaid/consumer-information/

Course Outcomes

Students should be able to do the following upon successful completion of the course:

1. Given a description of a problem, apply the problem-solving steps used in computer programming to create a solution design.
2. Working from a solution design, implement a solution to a problem using the Java programming language.
3. Use incremental development to construct a working Java program.
4. Identify and apply appropriate data types within a Java solution.
5. Describe and identify key object-oriented programming concepts.
6. Differentiate between the memory allocation approach for primitive and reference data types in Java.
7. Examine the code available in the Java standard class libraries, and incorporate relevant Java standard classes into object-oriented design and program construction.
8. Create and document program design solutions for simple Java programs.
9. Given a solution design, create programmer-defined classes and incorporate these classes into Java program solutions.
10. Distinguish among the options for input and output using Java, and select appropriate approaches for a given Java solution.
11. Describe scope and persistence of objects and variables in object-oriented programming.
12. Identify and correctly apply sequence, selection, and iteration/repetition patterns in object-oriented Java solutions and program designs.
13. Identify and apply advanced class and object features, including: overloading methods and constructors, argument passing, object return from methods, and organizing classes into packages.
14. Manipulate collections of data using arrays and objects to solve a given problem using Java.
15. Describe the different sorting options available and select the best basic sort for use in a Java solution.
16. Apply test-first development to the construction of an object-oriented computer program.
17. Read and interpret UML 2.0 diagrams that document a problem, and implement the proposed solution using Java.
18. Implement professional standards and guidelines for designing and coding Java computer programs.
19. Present and justify, to a group of peers, the design and implementation of a problem solution.
20. Plan for and schedule adequate time to complete labs and projects no later than the required due date.
21. Consult various online and independent resources to independently attempt to resolve problems BEFORE requesting assistance from co-workers/co-learners or supervisor/instructor.
22. Determine when it is appropriate to seek assistance, from co-workers/co-learners or supervisor/instructor to resolve problems that could not be resolved independently.