Object Oriented Design & Programming I

Computer Science 221

Instructor:	Erik Krohn
E-mail:	krohne@uwosh.edu
Text Message Only:	608-492-1106
Class Times:	Monday, Wednesday: 11:30am - 12:30pm
Classroom:	Halsey 237
Lab Times:	Friday: 11:30am - 12:30pm
Lab Room:	Halsey 101C
Office Location:	Halsey 216
Office Hours:	Monday: 12:30pm - 1:30pm Wednesday: 10:30am - 11:30am Thursday: 9:30am - 11:30am
Prerequisites:	Math-104 or Math-106 or Math-108 or Math 206 or CompSci-142 with a grade of C or better or Math Placement into Math-171.
Course Website:	`http://www.uwosh.edu/d2l`
Recommended Textbook:	Programming In Java with zyLabs

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, 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.

Course Website

You should check d2l on a regular basis - it will contain lecture notes, handouts, assignments, announcements, and grades. Ill 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.

Mini Assignments

You will have daily mini assignments. Mini assignments are generally short and should take less than an hour to complete. You will be assigned a mini assignment every class period to ensure you are staying current with the material. I will drop your 2 lowest mini assignments. Not all mini assignments will be graded. Since I go over the solutions at the beginning of class, **no late mini assignments will be accepted.**

Assignments & Labs

Most assignments and labs will consist of short programming projects. 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 comment anything that might be confusing to a reader unfamiliar with your program, to name variables intelligently, to use indentation that reflects the codes 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.

One of the goals of this class is to teach you to write functioning programs in Java - thus, your code must compile in order for you to receive any credit. Code that does not compile will not be tested and your score will be a 0. 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 doesnt work at all, even though most of the code is written.

All assignments must be submitted electronically via d2l. It is your responsibility to ensure that your submission was submitted correctly. You must double check to ensure your program was uploaded correctly. Each assignment must be submitted by 11:00pm on the night of the due date. **There are no late submissions**.

Resubmissions

Each assignment must pass **all** of my test cases. A program that fails just one test case will receive at most 70%. That said, you will be allowed to resubmit assignments *once* for regrading within 7 days of the initial program being graded. Complete test cases will be posted after the initial grading so you can go back and test your code to see what went wrong. Regraded programs can receive a maximum score of 85% if the modifications made were very minor. Significant modifications or many additions will receive very little, if any, additional points.

Exams

Exam material will come from the lecture notes, mini assignments, labs, book and assignments. There will be more information about each exam as it approaches. The *tentative* exam dates are listed below. All exams will be taken during the regular class period. These may change, so as the date approaches make sure you've got the most recent information.

- **Exam One** - Wednesday, October 4^{th}, 2017
- **Exam Two** - Wednesday, November 8^{th}, 2017
- **Exam Three** - Wednesday, December 13^{th}, 2017

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, telephone me at 424-7080 or leave a message at the computer science office, 424-2068 or send me a text message. 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.

If allowed, only one make-up exam will be given. It will be a comprehensive exam given at an arranged time during the last week of the semester.

Grading

Course grades will be based on assignments, mini assignments, labs and exams. Your final grade will be computed with the following percentages:

- 40% - assignments
- 15% - mini assignments & labs
- 45% - exams

If you believe anything was graded incorrectly or unfairly and would like to have it regraded, you must let me know about it within *one week* of having the item graded. I will regrade the entire assignment or exam and you may gain or lose points.
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 based on the following:

Percentage	Grade	Percentage	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

Textbook

The textbook is a recommended textbook. If you understand everything that is going on with the program on this page: http://www.uwosh.edu/faculty_staff/krohne/CS221Pretest.java, then you do not need the book. If you did not understand most of it, then you need to get the book. In order to get the book, follow this instructions:

1. Go to http://learn.zybooks.com and sign in or create an account.
2. The zyBook code for this course is: UWOSHCOMPSCI221KrohnFall2017

Academic Dishonesty

Academic dishonesty of any kind will not be tolerated. All assignments, labs, mini assignments 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.

Course Outcomes

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.

Topic Coverage

1. Introduction to Computational Thinking
2. Computer Programming as a Problem-solving and Design Process
3. Introduction to Object-Oriented Programming Concepts
4. Objects, Classes, Members, Methods, and Standard Class Library
5. Abstraction and OO Program Design: Notation and Tools
6. Input/Output in Java
7. Data Types and Persistence
8. Arithmetic Operators and Mathematical Functions
9. Creating and Using Classes in Java
10. Methods: Static, Parameters, Return Values, and Overloading
11. Sequence, Selection, and Decision-Making in Solutions
12. Iteration with Loops
13. One- and Two-Dimensional Arrays
14. Searching
15. Sorting