Object-Oriented Design and Programming I
(Comp Sci 221, Section 001)

Instructor: Erik Krohn
E-mail: krohne@uwosh.edu
Text Message Only: 920-644-3745
Teaching Assistant: Erich Brungraber
Teaching Assistant's E-mail: brunge63@uwosh.edu
Class Information: Monday, Friday: 208 Halsey, 10:20am – 11:20am
Wednesday: 101C Halsey, 10:20am – 11:20am
Office Location: 216 Halsey
Office Hours: Monday, Wednesday, Friday: 2:00pm – 3:00pm
Tuesday: 2:00pm – 4:00pm
Tutor Hours: Sunday to Thursday: Halsey 101C, 6:00pm – 7:00pm
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: Object Oriented Programming, J. Dean & R. Dean, ISBN 9781121752627
Other book: Introduction to Programming with Java, J. Dean & R. Dean, ISBN 9780073047027

Course Information
A first course in problem solving, software design, and computer programming using the Java language. Problem solving/software design techniques will be drawn from: flow charts, pseudo code, structure charts, and class diagrams. Data structures and algorithms include: Arrays, character strings, searching, and sorting. Programming topics include: data types, assignment statements, standard input/output, selection, repetition, functions/methods, parameters, scope of identifiers, data file input/output, recursion, and simple GUIs.

Course Website
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.

Grading
Course grades will be based on programming assignments, labs, quizzes and three exams. Your final grade will be computed as follows:

40% - programming projects
15% - weekly labs/quizzes
45% - exams

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	Percentage	Grade
>91	A	79 - 81	B-	67 - 69	D+
89 - 91	A-	77 - 79	C+	61 - 67	D
87 - 89	B+	71 - 77	C	55 - 61	D-
81 - 87	B	69 - 71	C-	<55	F

Exams
Exam material will come from the lecture notes, quizzes, book, programming assignments and labs. 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.

Midterm 1: Friday, March 1st, 2013
Midterm 2: Friday, April 12th, 2013
Final: Wednesday, May 8th, 2013

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 at the number on the first page). **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.

Quizzes
You will have quizzes throughout the semester. Quizzes are generally short and should only take a few minutes to complete. You will be given a quiz every 3-5 class periods to ensure you are staying current with the material. Your lowest quiz will be dropped. There are no make-up quizzes.

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 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.

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 <u>at most</u> 50% credit, and often substantially less.** 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.

Each project *must* include a README text file with any information I need to know regarding this project. The information in this file will be taken into account during grading, so it will be beneficial for you to make sure that everything I need to know about your work is written in this file.

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. **Late assignments are penalized at 10% per day up to 7 days.** If you believe an assignment or exam was graded incorrectly or unfairly and would like to have it re-graded, please let me know about it *within one week* of having the assignment or exam graded. I will re-grade the entire assignment and you may gain or lose points.

Academic Dishonesty

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

- Topic areas and corresponding Learning Outcomes
    - Given a description of a problem, apply the problem-solving steps used in computer programming to create a solution design.
    - Working from a solution design, implement a solution to a problem using the Java programming language.
    - Use incremental development to construct a working Java program.
    - Identify and apply appropriate data types within a Java solution.
    - Describe and identify key object-oriented programming concepts.
    - Differentiate between the memory allocation approach for primitive and reference data types in Java.
    - Examine the code available in the Java standard class libraries, and incorporate relevant Java standard classes into object-oriented design and program construction.
    - Create and document program design solutions for simple Java programs.
    - Given a solution design, create programmer-defined classes and incorporate these classes into Java program solutions.
    - Distinguish among the options for input and output using Java, and select appropriate approaches for a given Java solution.
    - Describe scope and persistence of objects and variables in object-oriented programming.
    - Identify and correctly apply sequence, selection, and iteration/repetition patterns in object-oriented Java solutions and program designs.
    - Identify and apply advanced class and object features, including: overloading methods and constructors, argument passing, object return from methods, and organizing classes into packages.
    - Manipulate collections of data using arrays and objects to solve a given problem using Java.
    - Describe the different sorting options available and select the best basic sort for use in a Java solution.
    - Apply test-first development to the construction of an object-oriented computer program.
    - Read and interpret UML 2.0 diagrams that document a problem, and implement the proposed solution using Java.
    - Implement professional standards and guidelines for designing and coding Java computer programs.
    - Present and justify, to a group of peers, the design and implementation of a problem solution.
    - Plan for and schedule adequate time to complete labs and projects no later than the required due date.