Object-Oriented Design and Programming I:

(Comp Sci 221, Section 001 – Fall 2014)

Instructor: Scott Summers
Email: summerss@uwosh.edu
Office: Halsey 220
Phone: 920-424-1324

Office hours: MWF 10:20 AM – 11:40 AM
R 9:30 AM – 11:30 AM

Class information: MW: Halsey 237, 9:10am – 10:10am,
F: Halsey 101C, 9:10am – 10:10am.

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. You should check d2l on a regular basis as it will contain any 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.

Recommended textbook: Introduction to Programming with Java – A Problem Solving Approach, J. Dean & R. Dean, 2nd ed, ISBN: 978-0-07-337606-6.

Course description: 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, debugging.

Course grade: Your final course grade will be based on the following components:

120 pts — Homework: There will be five or six programming assignments. At least one programming assignment will be performed in a group (two students per group).

180 pts — Exams: There will be three exams – two midterm exams and a final exam, each worth 60 points.

50 pts — Labs: There will be 11 lab assignments, each worth 5 points. Your lowest lab score will be dropped before your final grade is computed.

50 pts — Quizzes: There will be six quizzes given, each worth 10 points. Your lowest quiz score will be dropped before your final grade is computed. Each quiz will be taken in about 20-25 minutes of a normal class period, usually at the end of class. At least two quizzes will be given before each exam.

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 and ≤ 81	B-	>67 and ≤ 69	D+
>89 and ≤ 91	A-	>77 and ≤ 79	C+	>61 and ≤ 67	D
>87 and ≤ 89	B+	>71 and ≤ 77	C	>55 and ≤ 61	D-
>81 and ≤ 87	B	>69 and ≤ 71	C-	≤55	F

Exams: Exam material will come from the lecture notes, quizzes, programming assignments and labs. There will be more information about each exam as it approaches.

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 920-424-1324 or leave a message at the Computer Science office, 920-424-2068 **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.

Assignments and Labs: Most assignments and labs will consist of short to medium-length 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. For more specific information regarding proper formatting of your programs, refer to the **Programming Ground Rules for Comp Sci 221** document. All programming assignments must strictly adhere to the aforementioned programming ground rules. Labs DO NOT need to follow the programming ground rules.

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 0 credit**. 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 d2l (each lab and assignment will contain specific instructions). It is your responsibility to ensure that your assignment or lab was submitted correctly. You must double check to ensure your assignment or lab was uploaded correctly.

No late submissions will be accepted.

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 (except for those designated as group 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. For group assignments, each group must submit original work.

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

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.