

Object-Oriented Design and Programming II CS 262 - Spring 2018

Instructor: George Thomas **Office:** Halsey 218
Email: thomasg@uwosh.edu **Phone:** 424-2069

Office Hours: MW 11:30-1:00
 T 11:30-12:30
 R 8:30-9:30
 Or by appointment

Lectures: MTR 10:20-11:20, Halsey 208
Lab: W 10:20-11:20, Halsey 101C

Prerequisites: Math-108 or equivalent with a grade of C or better, or qualifying for a higher level mathematics via the Mathematics Placement Test, and CompSci-221 or equivalent with a grade of C or better.

Required Textbook: *COMP SCI 262: Object Oriented Design and Programming II*, 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: **UWOSHCOMPSCI262ThomasSpring2018**
3. Subscribe. A subscription is \$48 and will last until Jun 02, 2018. Students will be able to subscribe until May 05, 2018. If you purchased this book for CS 221 last semester, you should automatically receive a 50% discount on the price.

Course Website: UWO D2L

Note: If you have special needs, please come and talk to me at the end of the first class.

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.

Course Outcomes

1. Debugging Java programs with BlueJ – Students will be expected to:
 - analyze a program on its correctness
 - identify software bugs with the debugger in BlueJ
2. Objects and Classes – Students will be expected to:
 - specify a class with the UML graphical notation
 - use the UML graphical notation to describe classes
 - distinguish between object reference and primitive data type variables
 - apply classes in the Java API (Application Programming Interface)
 - differentiate between instance and static variables
 - develop methods in classes
 - store and process objects in arrays
 - apply class abstraction to develop software
3. Inheritance and Polymorphism – Students will be expected to:
 - develop a subclass from a superclass through inheritance
 - apply the polymorphism concept to handle different data types using a uniform interface
4. Abstract Classes and Interfaces – Students will be expected to:
 - identify the similarities and differences between an abstract class and an interface
 - model weak inheritance relationships with interfaces
 - specify a natural order using the Comparable interface
 - to wrap primitive data values into objects
 - create a generic sort method
 - simplify programming using JDK 1.5 automatic conversion between primitive types and wrapper class types
5. Exceptions and Assertions – Students will be expected to:
 - distinguish exception types: Error versus Exception in Java
 - throw an exception in a method
 - write an exception handler using a try-catch-finally block
 - explain the propagation of an exception
 - apply assertions to help ensure program correctness
6. Text I/O – Students will be expected to:
 - Read and write characters using the InputStreamReader, FileReader, BufferedReader, OutputStreamWriter, FileWriter, PrintWriter, BufferedWriter classes
 - Be able to apply the appropriate class in text I/O operations based on the requirements and performance needs
 - Distinguish between text I/O and binary I/O
7. Object-Oriented Design – Students will be expected to:
 - become familiar with the software development process
 - model a system with the appropriate relationships: association, aggregation, composition, dependency, strong inheritance, and weak inheritance
 - declare classes to represent the relationships among them.
 - design systems by identifying the classes and discovering the relationships among these classes

8. Unit testing with JUnit – Students will be expected to:
 - Create test classes, test methods, and run tests with JUnit
 - Create and use test fixtures in JUnit
 - Interpret test results with JUnit
 - Correlate the test fixtures with assertions
 - Verify that a software unit performs as specified
9. GUI and Graphics – Students will be expected to:
 - Describe the Java GUI hierarchy
 - Create user interfaces using frames, panels, and simple GUI components
 - Apply layout managers
 - Use JPanel as subcontainers
 - Draw figures using the methods in the Graphics class
 - Override the paintComponent method to draw figures on a GUI component
 - Introduction to Threads – creation, simple usage
10. Event Driven Programming – Students will be expected to:
 - declare listener classes and write event handlers to handle events
 - apply the Observer Pattern to decoupled programs
 - register listener objects in the source object
 - create inner classes and anonymous inner classes
 - write programs to handle(ActionEvent, MouseEvent, KeyEvent, and Timer event
11. Recursion – Students will be expected to:
 - solve problems with recursion
 - write programs using recursion
 - explain the difference between iteration and recursion
12. Generic Types – Students will be expected to:
 - improve reliability and readability of Java programs by using generic types
13. Java Collections Framework – Students will be expected to:
 - describe the Java Collections Framework hierarchy
 - utilize the common methods in the Collection interface for operating sets and lists
 - use the Iterator interface to traverse a collection
 - examine the Set interface and be capable of deciding when to use HashSet, LinkedHashSet, or TreeSet to store elements
 - compare elements using the Comparator interface
 - examine the List interface, and be capable of deciding how and when to use ArrayList or LinkedList to store elements
 - 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.

Course Grading Policy: Your final grade for this course will be based on five components, namely exams, programming assignments, labs, unannounced quizzes 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% |
| Assignments | 23% |
| Labs | 20% |
| Quizzes | 4% |
| Participation and Challenge Activities | 8% |

Tentative Exam Dates are as follows:

- **Exam 1 - Wednesday, 2/21 in lab**
- **Exam 2 - Wednesday, 4/04 in lab**
- **Exam 3 - Wednesday, 5/09 in lab**

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.

Attendance and Participation: You are expected to not only attend **every** class meeting on time but also to come **prepared** for and **participate** actively in it. Necessary preparation requires you to have studied and assimilated the material covered in previous sessions, to have met with me or the tutors outside of class to discuss any questions you may have, to have done the assigned reading (when applicable), and to have completed the programming assignments on time. I **strongly encourage you to ask any question** or raise any issue you have with the course either during or at the end of class, or during my office hours. I will also gladly meet with you by appointment.

Deadlines: 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 assignment will also come with a deadline (day and time) by which it must be submitted. You are allotted *three* assignment credit days you can use through the semester. A credit day is exactly 24 hours or less. You can use unused credit days to submit an assignment after its deadline, without penalty. Any assignment submitted after the deadline, plus any credit days you have unused, will receive a zero.

For example, if you have 2 unused credit days available and an assignment is due on Tuesday at 5:00PM, you can submit it anytime by exactly Thursday at 5:00PM without penalty. Do note that if you submit your assignment on Thursday at 5:01PM, you will be penalized 100% of the score of the assignment and thus receive a zero! Note also that if you submit your assignment on Wednesday at 5:01PM, you will be charged two credit days (but no penalty, obviously).

Absences and Extensions: Extensions on deadlines may be granted at the discretion of the instructor if you provide a valid justification (in the form of a written excuse from a medical doctor or the Dean of Students Office) **before** the due date.

If you miss a scheduled exam (dates are given in the calendar), you **may** be able to take a make-up exam provided you give the instructor a valid justification (see above) ahead of time if possible. Only one make-up exam will be given. It will be a comprehensive exam scheduled at the end of the semester. Similarly, there will be no make-up quizzes unless the instructor is provided with a valid justification (see above) for your absence on the day of the quiz.

Collaborating versus Cheating: It should be clear that any collaboration during exams or quizzes is considered cheating. Unless otherwise stated in the assignment or lab, all assignment and lab submissions must be entirely your own work. While it is acceptable to discuss the assignments at a high level (for example, at the design level) with others, you must submit your own work. You may not “borrow” any piece of code or design of any length from someone else, unless you can live with a zero and the other potential academic sanctions of cheating (see [UWO Student Discipline Code 2007](#), Chapter UWS 14).