

**CS 262 - Object-Oriented Design and Programming II**  
**Spring 2009**  
**Credits: 4 hours**

**Instructor:** David Furcy  
**Email:** [furcyd@uwosh.edu](mailto:furcyd@uwosh.edu)

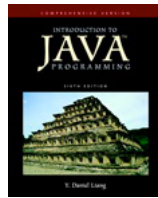
**Office:** Halsey 220  
**Phone:** 424-1182

**Class Meetings:** 11:30-12:30PM on **Mondays, Wednesdays, and Thursdays** in **HS 212**  
11:30-12:30PM on **Fridays** in **HS 101C**

**Office Hours:** **MWThF** 9:10-10:10AM and **Tu** 8:00-10:00AM in HS 220  
Feel free to drop in at any other time when my door is open.  
Alternatively, to ensure that I have enough time to answer your questions, I strongly encourage you to make an appointment.

**Prerequisites:** Math 108 or equivalent with a grade of C or better, or qualifying for a higher level mathematics course via the Mathematics Placement Exam, and CS 221 or equivalent with a grade of C or better.

**Required Textbook:**



*Introduction to Java Programming - Comprehensive Version, SIXTH Edition*  
by Y. Daniel Liang  
Prentice Hall

**References:**

- Class handouts. **TAKE NOTES** on and about them. Handouts that are not liberally saturated with your own notes will likely prove useless when you need them most!
- Course web page at: [http://www.uwosh.edu/faculty\\_staff/furcyd/cs262/](http://www.uwosh.edu/faculty_staff/furcyd/cs262/)

**Tests:**

Exam 1: Thursday, March 12<sup>th</sup>, 5:00PM  
Exam 2: Thursday, April 16<sup>th</sup>, 5:00PM  
Exam 3: Wednesday, May 13<sup>th</sup>, 5:00PM

Exams are scheduled at night to circumvent the time limit imposed by in-class exams. Please block these time slots at the beginning of the semester. The procedure and criteria of eligibility for making up a missed exam are described below.

**If you have special needs (or ABSOLUTELY cannot make it to one of the above night exams), please come and talk to me RIGHT AWAY so I can accommodate your needs.**

## Topics and Learning Outcomes

1. Finite State Machine – You will be expected to
  - a. apply the finite state machine concept to design systems by drawing a state-transition diagram
  - b. model states of a system as objects
  - c. apply the State Pattern to accommodate changes
2. Debugging with BlueJ – You will be expected to
  - a. analyze a program for correctness
  - b. identify software bugs with the BlueJ debugger
3. Objects and Classes – You will be expected to
  - a. specify a class with the UML graphical notation
  - b. use the UML graphical notation to describe classes
  - c. distinguish between object reference and primitive data type variables
  - d. apply classes in the Java API (Application Programming Interface)
  - e. differentiate between instance and static variables
  - f. develop methods in classes
  - g. store and process objects in arrays
  - h. apply class abstraction to develop software
4. Inheritance and Polymorphism – You will be expected to
  - a. develop a subclass from a superclass through inheritance
  - b. apply the polymorphism concept to handle different data types using a uniform interface
5. Abstract Classes and Interface – You will be expected to
  - a. identify the similarities and differences between an abstract class and an interface
  - b. model weak inheritance relationships with interfaces
  - c. specify a natural order using the Comparable interface
  - d. wrap primitive data values into objects
  - e. create a generic sort method
  - f. simplify programming using JDK 1.5 automatic conversion between primitive types and wrapper class types
6. Design Patterns – You will be expected to
  - a. identify the weaknesses of inheritance in software maintenance when requirements change
  - b. identify the aspects of your application that vary and separate them from what stays the same
  - c. program to a supertype, not an implementation
  - d. favor composition (HAS-A) relationships over (IS-A) relationships in design of classes
  - e. apply the Strategy Pattern that defines a family of algorithms, encapsulates each one, and makes them interchangeable so that strategy lets the algorithm vary independently from clients that use it
7. Exceptions and Assertions – You will be expected to
  - a. distinguish exception types: Error versus Exception in Java
  - b. throw exceptions in a method

- c. write an exception handler using a try-catch-finally block
  - d. explain the propagation of an exception
  - e. apply assertions to help ensure program correctness
8. Text I/O – You will be expected to
- a. read and write characters using the InputStreamReader, FileReader, BufferedReader, OutputStreamWriter, FileWriter, PrintWriter, BufferedWriter classes
  - b. apply the appropriate class in text I/O operations based on the requirements and performance needs
  - c. apply the Decorator Pattern in the use of text I/O Reader and Writer classes
9. Binary I/O – You will be expected to
- a. distinguish between text I/O and binary I/O
  - b. read and write bytes using FileInputStream and FileOutputStream
  - c. read and write primitive values and strings using DataInputStream and DataOutputStream
  - d. use the Serializable interface to enable objects to be serializable
  - e. store and restore objects using ObjectOutputStream and ObjectInputStream
  - f. apply the Decorator Pattern in the use of binary I/O Stream classes
10. Object-Oriented Design – You will be expected to
- a. become familiar with the software development process
  - b. model a system with the appropriate relationships: association, aggregation, composition, dependency, strong inheritance, and weak inheritance
  - c. specify classes and the relationships among them
  - d. design systems by identifying the classes and discovering the relationships among these classes
11. Unit testing with JUnit – You will be expected to
- a. create test classes, test methods, and run tests with JUnit
  - b. create and use test fixtures in JUnit
  - c. interpret test results with JUnit
  - d. correlate the test fixtures with assertions
  - e. verify that a software project performs as specified
12. GUI and Graphics – You will be expected to
- a. describe the Java GUI hierarchy
  - b. create user interfaces using frames, panels, and simple GUI components
  - c. apply layout managers
  - d. use JPanel as subcontainers
  - e. draw figures using the methods in the Graphics class
  - f. override the paintComponent method to draw figures on a GUI component
13. Event-Driven Programming
- a. declare listener classes and write event handlers to handle events
  - b. apply the Observer Pattern to decouple classes
  - c. register listener objects in the source object
  - d. create inner classes and anonymous inner classes
  - e. write programs to handle ActionEvent, MouseEvent, KeyEvent, and Timer event

14. Recursion – You will be expected to
  - a. analyze and solve problems with recursion
  - b. write recursive program
  - c. explain the difference between iteration and recursion
15. Data Structures: Lists, Stacks, Queues, Binary Search Trees – You will be expected to
  - a. debug a dynamic data structure with jGRASP
  - b. describe what a data structure is
  - c. explain the limitations of arrays
  - d. design and implement a dynamic list using an array
  - e. design and implement a dynamic list using a linked structure of nodes
  - f. design and implement a stack using an array list
  - g. design and implement a queue using a linked list
  - h. design and implement a binary search tree
  - i. traverse a binary search tree
  - j. insert nodes into a binary search tree
  - k. delete a node from a binary search tree
  - l. develop API (Application Programming Interface)
16. Generics – You will be expected to
  - a. improve reliability and readability of Java programs by using generic types
17. Java Collections Framework – You will be expected to
  - a. describe the Java Collections Framework hierarchy
  - b. use the common methods in the Collection interface for operating on sets and lists
  - c. use the Iterator interface to traverse a collection
  - d. examine the Set interface and be capable of deciding when to use HashSet, LinkedHashSet, or TreeSet to store elements
  - e. compare elements using the Comparator interface
  - f. examine the List interface, and be capable of deciding how and when to use ArrayList or LinkedList to store elements
  - g. examine the Map interface 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 four components, namely exams, programming assignments, labs, and quizzes. Within each component (for example, the *labs* component), all items (for example, all labs) will have equal weight. In other words, each component grade will be computed as the average of all items in the component (however, see important note below). 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)	40%
Assignments (6-7)	40%
Labs (12-13)	10%
Quizzes (10-15)	10%

**Important note:** If you receive a zero on 3 or more of your lab grades, your **overall lab grade** will automatically be reduced to 0. The same rule applies to your **overall quiz grade**.

Finally, your letter grade for the course will be computed using the following mapping:

Numerical Score	Course Grade	Numerical Score	Course Grade
$\geq 92$	A	$\geq 72$	C
$\geq 89$	AB	$\geq 69$	CD
$\geq 82$	B	$\geq 60$	D
$\geq 79$	BC	$< 60$	F

I will be glad to discuss any questions you may have about grades. However, make sure to bring them up right away, upon return of each graded assignment or exam. Last minute requests after the final exam will not be entertained.

**Attendance and Participation:** You are expected to not only attend **every** class meeting 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 outside of class to discuss any questions you may have, and to have completed the programming assignments on time.

**It is hard to imagine how a student could do well in this course while missing classes, attending them unprepared, or not participating.**

On the positive side, I have high expectations for my students and will always support and encourage you. 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. Send me email or give me a call to make an appointment. While I will meet with you as soon as my schedule permits, do not expect me to be widely available before an assignment is due.

**Late Submissions:** I will describe the submission procedure for your labs and assignments when the time comes. However, let me point out right away that each lab and assignment will come with a deadline (day and time) after which any submission is considered late, **with no exception**. The penalty for late submissions is computed as follows:

Turned in	Penalty
on due date, after deadline	10%
one day late	20%
two days late	40%
three days late	60%
four days late	80%

Note that submissions that are more than 4 days late will receive zero points. Late submissions can easily be avoided by starting to work on the assignment right away and asking me questions early if you get stuck.

The penalty for late submissions can be waived in **only one** scenario, namely if you give me a signed note from the attending physician or a written justification for the extension from the Dean of Students Office. If you miss a scheduled exam, you **may** be able to take a make-up exam provided you give me 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.

**Bonus points:** Since late submissions are penalized, it is only fair to reward early ones as well. Therefore, if you submit a **CORRECT** assignment at least one full day early (and up to four days early), you will earn 3 bonus points for every full 24-hour interval between your submission time and the deadline time. Thus a maximum of 12 bonus points may be earned for each assignment. Furthermore, there will be other opportunities for students to earn bonus points throughout the semester.

**Collaboration versus Cheating:** All submissions must be the work of a single student (the one whose name appears on the submission, that is). While it is acceptable to discuss the assignments with others, you must submit your own work. You may not look at or “borrow” any piece of code of any length from anyone else, unless you can live with a zero and the other potential academic sanctions of cheating. Check out the UWO Student Discipline Code (UWS 14) at <http://www.uwosh.edu/dean/conduct.htm> for details.

**Final note:** Be aware that computer science classes require a lot of work in addition to active participation in class. It takes considerable practice to develop the technical and analytical skills targeted by this course. You will need to spend **at least (and typically more than) three hours of effort outside of class for each in-class hour**. Having said this, I do expect every committed and hardworking student to do well in this course. I am looking forward to a fun and rewarding semester together. Given my high standards, I could not be more satisfied than if everyone earned a good grade in this course.