Object-Oriented Design and Programming II
CS 262 - Fall 2013

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

Office Hours:	MTWR	1:45-3:00	
	Or by appointment		
Lectures:	MTR	12:40-1:40,	HS 309
Lab:	W	12:40-1:40,	HS 101C

Prerequisites: Math-108 or equivalent with a grade of C of 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.

Recommended Textbook: Introduction to Java Programming, by Y. Daniel Liang, *Eighth Edition, Prentice Hall.*

Course Website: UWO D2L

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

Tutoring: The Computer Science department provides tutoring for the introductory CS courses. Jacob Colstrom and Sam Micka are the tutors this semester and their tutoring hours are: *Jacob – Sundays and Thursdays (6:00-7:00PM)*, and *Sam – Mondays, Tuesdays and Wednesdays (6:00-7:00PM)* in the *Halsey Computer Lab (Room 101)*. They can help you with general programming and Java questions but may not be able to answer specific questions on assignments or course administrative details.

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

Topic areas and corresponding Learning Outcomes:

1. Debugging Java programs with BlueJ – You will be expected to:
    - analyze a program on its correctness
    - identify software bugs with the debugger in BlueJ
2. Objects and Classes – You 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 – You 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 – You 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 – You 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 – You 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 – You 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 – You 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 – You 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 – You 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 – You will be expected to:
    - solve problems with recursion
    - write program using recursion
    - explain the difference between iteration and recursion
12. Generic Types – You will be expected to:
    - improve reliability and readability of Java programs by using generic types
13. Java Collections Framework – You 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 suing 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 four components, namely exams, programming assignments, labs, and unannounced quizzes. 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)	48%
Assignments	25%
Weekly labs	18%
Quizzes	9%

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.

Assignment and Lab Deadlines: Each assignment and lab will come with a deadline (day and time) by which they must be submitted. Late submissions will NOT be accepted. Extensions on assignments and labs 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: Unless otherwise stated in the assignment or lab, all 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).

Tentative Calendar

Week	Date	Topics	Labs & Exams
1	9/4-9/6	Review – Java Basics	
2	9/9-9/13	Searching and Sorting	Lab 1
3	9/16-9/20	Recursion	Lab 2
4	9/23-9/27	Debugging , Software Process and Testing, OOD – Encapsulation and Abstraction	Lab 3
5	9/30-10/4	OOD - Inheritance and Polymorphism	Lab 4
6	10/7-10/11	OOD - Abstract Classes and Interfaces	**Exam 1 (10/9) in Lab**
7	10/14-10/18	Exceptions	Lab 5
8	10/21-10/25	File I/O, Regular Expressions	Lab 6
9	10/28-11/1	Wrapper Classes	Lab 7
10	11/4-11/8	Generics	Lab 8
11	11/11-11/15	Collections	**Exam 2 (11/13) in Lab**
12	11/18-11/22	Collections	Lab 9
13	11/25-11/26	Events	
	11/27-11/29	**THANKSGIVING BREAK**	
14	12/2-12/6	Events and Threads	Lab 10
15	12/9-12/13	Review	**Exam 3 (12/12) in lecture room**