CompSci 271 Data Structures Spring-17

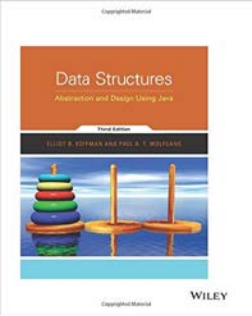
Instructor: George Georgiev
Office: HS, Room 217

Phone: 424 - 11 80 **E-mail:** georgiev@uwosh.edu

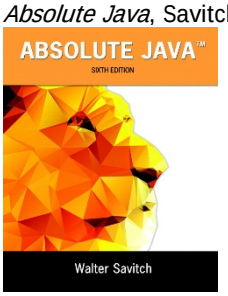
Office Hours: M Tu W F: 9:10 – 10:10

Lectures: MTuW: 8:00 am - 9:00 am, HS 212
Labs: F: 8:00 am – 9:00 am, HS 101 C

Required Text:
Data Structures: Abstraction and Design Using Java, Koffman & Wolfgang, John Wiley & Sons, 3d ed.



Recommended Text:
Absolute Java, Savitch, Pearson, 6th ed., 2016.



All the materials and other information for the course will be posted on D2L.

Class Details

Status		Career	Undergraduate
Class Number	70019	Dates	1/30/2017 - 5/12/2017
Session	Fourteen Week	Grading	Student Option
Units	4 units	Location	Oshkosh
Instruction Mode	In Person	Campus	Main
Class Components	Lecture Required		

Current Catalog Description -
A course surveying the fundamental methods of representing data and the algorithms that implement and use those data representation techniques. Data structures and algorithms include; linked lists, stacks, queues, trees, heaps, priority queues, hashing, searching, sorting, data compression, graphs, recursion. Analysis topics include: elementary big-O analysis, empirical measurements of performance, time/space trade-offs, and identifying differences among best, average, and worst case behaviors. Prerequisites: Computer Science 262 with a grade of C or better AND either Completion of or concurrent enrollment in Computer Science 251 with a grade of C or better. (Fall, Spring)

Prerequisite: Prerequisites: Computer Science 262 with a grade of C or better AND either Completion of or concurrent enrollment in Computer Science 251 with a grade of C or better.

Course Outcomes
1. Given a non-recursive algorithm, the student will be able to examine its loop structures and infer its asymptotic runtime using big-O notation.
2. Given a recursive algorithm, the student will be able to examine its recursive structure, determine the corresponding recurrence relation (from a small collection of commonly occurring recurrence relations), and use the recurrence relation in determining the asymptotic runtime of the algorithm using big-O notation.
3. Given the description of a computational problem requiring a mixture of search, insertion, and/or deletion operations on collections of data, the student will be able to compare the relative advantages of using arrays and linked lists in solving the problem efficiently.
4. Given a classical computational problem (e.g., infix-to-postfix conversion, postfix-expression evaluation, path planning, minimum-spanning tree computation), the student will be able to trace a solution to the problem using appropriate data structures (e.g., stacks, queues, binary trees, binary search trees, red-black trees, graphs) and to predict the asymptotic runtime of the solution based on the selected data structures.
5. Given a collection of unordered data, the student will be able to trace the execution of an advanced sorting algorithm (such as quick sort and heap sort) on this data set.
6. Given a set of data keys, the student will be able to trace through a sequence of key insertions, searches and deletions on a balanced tree structure. The student will also be able to discuss the relationship between the number of keys and the execution time of these operations.
7. Given a set of data keys, a hash function, a table size, and a collision-handling strategy, the student will be able to trace through a sequence of key insertions and searches, and to discuss how varying the table size, hash function or collision-handling would affect the execution time of these operations.
8. Given a graph data structure, the student will be able to implement it using either adjacency lists or an adjacency matrix, to traverse it using either a depth-first or breadth-first strategy, to identify its structural properties (whether it is

directed, cyclic, connected, complete), and to trace the execution of one or more classical graph algorithms (e.g., Dijkstra's shortest path, topological sort or minimum-spanning tree computation).
9. Given a problem requiring the efficient use of a variety of data structures, the student will be able to apply object-oriented design principles in implementing and testing a solution to that problem in an appropriate object-oriented language.

Course Requirements:

There will be three exams, unannounced quizzes, programming assignments, and laboratory works. The material for all exams will come from either a material covered in class, lab work, and/or programming assignments.

Complete all required work on time. In the event that an exam must be missed, or required work can't be completed on time, due to illness or other serious and unavoidable circumstance, notify the professor in advance by phone or e-mail.

The programming assignments are due by 8:00 on the due date (electronic copy in a drop box is due by 8:00 am, and a paper (hard) copy of the assignment is due at the beginning of class). Programs will be accepted up to three days late subject to the following penalties:

Turned in	Penalty
After 9:00 am on the due date	10%
1 day late	25%
2 days late	50%
3 days late	75%

Saturdays, Sundays, and holidays count when computing penalties.

If you work with partners, you will submit one electronic copy and one paper copy of the assignment with names on it. Partners will earn equal scores on the assignment. You may work alone on some assignments and with a partners on others. You may change partners during the semester.

You are encouraged to discuss assigned problems with other people but you must individually (group) design and write your own solutions/code for all assignments. Submitting modified versions of other people's work as your own is considered cheating.

There will be no make up for unannounced quizzes.

There will be one make up for the exams, which will cover all topics. It will be at the end of the semester.

Make up exam will be given if you call before the exam, make arrangements, have a medical certificate signed by the physician, and have a note from the Dean of Students Office.

The three exams will be announced at least a week before taking place.

Laboratory assignments will be in the teaching lab. The materials will be placed on D2L. You are encouraged to discuss the lab assignment with other students before and during the lab hours, but each student must demonstrate her or his own solution. If you do not finish a lab assignment during lab session, you have to demonstrate your solution to the instructor during the instructor's office hours before next lab.

Evaluation:

Three Exams: ~60% (20% each)

Programming assignments:	~25%	(equal points for each assignment)
Unannounced quizzes:	~5%	(equal points for each quiz)
Laboratory Assignments	~10%	(equal points for each lab)

Grading:

Score	Grade
>= 92	A
90-92	A-
88-90	B+
82-88	B
80-82	B-
78-80	C+
72-78	C
70-72	C-
68-70	D+
62-68	D
60-62	D-
< 60	F

Feedback:

Your comments and questions about all aspects of the course (content, grading, teaching methods, pace, textbook, etc) are welcome. You can use e-mail or talk to me during office hours.