BROAD OVERVIEW OF TOPICS

Module 1: Key Concepts in Object-oriented Programming
Traditional programming paradigms, the OOP paradigm, Encapsulation, Inheritance, Polymorphism, Generics

Module 2: Algorithm Time Complexity
What is algorithm complexity?, Big O notation, Deriving Big O given a growth function, Problem solving with Big O, Examples of algorithms with various Big Os., Best case, worst case and average case complexity.

Module 3: Unordered Lists
Definition and examples, Generic set of operations and the big picture of implementation, Linked List Implementation, Complexities of operations Best case, worst case and average case search; smart search methods

Module 4: Ordered Lists
Definition and examples, Binary search algorithm and its complexity, Complexities of operations, Merging ordered lists.

Module 5: Recursion
Concept, recursive definition, writing recursive programs, examples

Module 6: Binary Trees
Definition, Binary Tree Terminology, Strictly binary and complete binary trees, Traversals, Recursive definitions for a binary tree, Binary Tree Class, Application: Huffman coding

Module 7: Binary Search Trees
Definition and Examples, Operations – Search, Insert and Delete, Binary Search Tree Class, Complexity of Binary Search

Module 8: AVL Trees
Motivation and Definition, AVL tree examples, AVL tree rotations with examples

Module 9: Heaps
Motivation, definition and properties, Operations – Search, Insert and Delete, Heap Class, Heap Applications – Sorting and CPU Process scheduling, Updatable Heaps

Module 10: Hashing
Hashing concept, hash tables and hash functions, Hash Collision, Separate chaining, Linear and Quadratic probing, Hash Functions, Strings as keys, Load Factor, Rehashing , HashMaps, TreeMaps and LinkedHashMaps

Module 11: Sorting Algorithms
Bubble Sort, Insertion Sort, Selection Sort, Quick Sort, Merge Sort, Bucket Sort, Complexities of each sorting algorithm

Module 12: Graphs
Graph Definitions and Terminology, Applications of graphs, Graph Types, Representation of an edge, Degree of a vertex, Indegree vs. Outdegree Representation of graphs (Adjacency Matrix Representation, Linked List Representation), Graph Traversals – breadth first, depth first, Topological Sort, Shortest Path Algorithms

Text/References
1. Data Structures Outside In with Java by Sesh Venugopal, Pearson Prentice Hall Publishers ISBN 0-13-198619-8 (this will provide the main outline of topics for the course).

PLEASE NOTE: Lecture Notes and Workbook given in class is important. Please bring it to every lecture.
LEARNING OUTCOMES

• Select and use appropriate abstract data types, data structures, and algorithms to solve moderately complex problems.
• Select the appropriate data structure to implement a given ADT under a given set of constraints.
• Determine the number of primitive operations of an algorithm in terms of instance size.
• Determine the asymptotic complexity (Big-O) of simple functions such as polynomial, polylogarithmic, and exponential functions.
• Implement a tree data structure and implement depth-first and breadth-first traversals.
• Implement a binary tree data structure and implement pre-order, in-order, and post-order traversals.
• Understand, compare, and implement various sorting algorithms, including: selection sort, insertion sort, quicksort, merge sort, heap sort, and radix sort.
• Implement simple iterative and recursive algorithms to solve moderately simple tasks.
• Implement the map, dictionary, and set ADTs using lists, binary search trees, and hash tables.
• Implement the priority-queue ADT using lists and heaps.
• Determine the asymptotic complexity of some simple iterative and recursive algorithms.
• Implement mechanisms to deal with collisions in a hash-table.
• Describe basic concepts in Graph Theory.
• Describe the graph ADT, as well as depth-first and breadth first traversals.
• Implement recursive search and state-space exploration algorithms.
• Implement various graph algorithms.