

Study Topics for CS165 Final Exam (Comprehensive)

1) Testing

- a. Software Testing
- b. Test Driven Development
- c. Black Box Testing
- d. Unit Testing
- e. JUnit assertions
- f. White Box Testing
- g. Coverage Testing

2) Recursion

- a. Activation Stack model for recursion
- b. Counting recursive calls, Drawing recursive calls trees
- c. Helper methods
- d. Memoization
- e. Examples: Fibonacci, Hanoi, Parking Lot, Spock's dilemma

3) Classes & Objects

- a. Classes versus Objects
- b. Class variables (static)
- c. Instance variable (non-static)
- d. Dot operator for data and method access
- e. Class and Instance methods
- f. Scoping of variables
- g. Instantiating an object from a class
- h. Class constructors
- i. Object references
- j. public versus private data and methods
- k. Getter and Setter methods
- l. Arrays / ArrayLists of objects
- m. **this** keyword (reference to current object and constructor)

4) Inheritance

- a. Terminology
 - ✓ Superclass versus Subclass
 - ✓ Base class versus Derived class

✓ Parent class versus Child class

- b. **extends** keyword
- c. default constructor
- d. no-arg constructor
- e. **super** keyword (for constructor and superclass methods)
- f. Constructor chaining
- g. **Overriding** methods
- h. **Overloading** methods
- i. Polymorphism and casting
- j. Dynamic binding
- k. **instanceof** operator
- l. **equals** and **toString** methods
- m. Access modifiers: **private**, default, **protected**, **public**
- n. **final** keyword

5) Abstract Classes

- a. Abstract Methods vs. Concrete Methods (differences)
- b. Sharing code using concrete methods in abstract class
- c. Differentiating code in concrete subclasses by overriding abstract methods
- d. **extends** keyword
- e. **super** keyword

6) Interfaces

- a. Only contains abstract methods
- b. Cannot contain concrete methods
- c. **implements** keyword
- d. **extends** keyword
- e. Methods only, no class or instance data
- f. Constants sometimes included
- g. Method return type and Signature (method name and parameters with types)
- h. Arrays and ArrayLists of interface types can store any class that implements the interface
- i. Comparable

7) Generics

- a. Generic types
- b. Raw types and compile time warnings
- c. Runtime errors when using raw types
- d. Compile time error messages using Generics
- e. Declaring Generic classes
- f. Declaring Generic methods
- g. Bounded generic (E Extends Type)

8) Data Structures

- a. Collections hierarchy
- b. List interface
- c. Queue interface
- d. Stack interface
- e. ArrayList class
- f. LinkedList class
- g. Comparable interface
- h. Iterable and Iterators
- i. ArrayList implementation of Stack/Queue
- j. LinkedList implementation of Stack/Queue
- k. Array implementation of Stack/Queue
- l. Circular implementation of queue of Queue
- m. Infix, Prefix, Postfix Expression
- n. Postfix Expression Evaluation using Stack

9) Priority Queues, Heaps

- a. Offer/add
- b. Complete binary tree
- c. Array storage
- d. maxHeap, minHeap
 - Insert/delete
 - HeapSort

10) Grammars, Expressions

- a. Grammars / Production Rules
- b. Regular Expressions

11) Trees

- a. Binary Trees
- b. Binary Search Trees (BST)
 - Searching
 - Inserting / Delete
 - Inorder, Preorder, Postorder, LevelOrder Traversal
- c. B+ trees
 - Searching
 - Insert
 - Delete
 - Rotation
 - Bulk Loading

12) Hashing

- a. Hashing functions
- b. Collisions
- c. Linear and Quadratic Probing, Double Hashing
- d. Separate Chaining (Hash Buckets)
- e. Dynamic Hashing – extendible

13) Unweighted Graphs

- a. Data Structures
- b. Adjacency Matrices
- c. Adjacency Lists
- d. Depth-First Traversal (DFS)
- e. Breadth-First Traversal (BFS)
- f. Topological Sort

14) Expression Trees

- a. Infix expression to expression tree
- b. Pre-order, in-order, post-order traversal of expression tree
- c. Expression tree evaluation