CSCI-UA 9102. Data Structures
Material for the Final

Augustin Cosse

Summer 2025

1 Material covered

1. You must be able to understand basic programming concepts including Base types, Strings, Wrappers, Arrays, Enum Types
2. You must be able to use and understand all the java modifiers including:
   - Access control modifiers: `public`, `private`, `protected`
   - The `static`, `abstract` and `final` modifiers

   You must be able to write a class, or a method using those modifiers.
3. You must be able to understand and use type conversion (especially between Strings and numbers)
4. You must be able to know how to import packages, classes and methods from the classes (e.g. the `Math` class or the `sqrt` method from the `Math` class)
5. You must be able to use and understand control flow in Java including `if else` and `switch` statements, `while` loops, `do while`, `for` and `for each` loops as well as `break`, `continue` and `return` statements.
6. You must be able to use and understand how to prompt for user input and read command-line arguments (i.e. `System.out.println`, `import java.util.Scanner` and `nextInt`, `nextDouble`,..)
7. You must be able to understand and explain object oriented programming (including the notion of constructor and the keyword `new`) and inheritance (including abstract classes and interfaces, the keywords `extends` and `implements`)
8. You must be able to explain how to catch and throw an exception

9. You must be able to use and manipulate multidimensional arrays.

10. You must be able to describe, compare and provide pseudo code for the implementation of Singly Linked Lists, Circularly Linked Lists and Doubly Linked Lists.

11. You must be able to explain the notions of shallow and deep copies and illustrate the difference between the two using simple examples.

12. You must be able to use and explain the big-Oh, big-Omega and big-Theta notations for the asymptotic analysis of the running time of algorithms.

13. You must be able to use and explain the notion of recursion. You must be able to provide a recursion based pseudo code for the following problems:

    - Compute the value of simple series such as $\sum_{i=1}^{N} \frac{1}{i}$ or $\sum_{i=1}^{N} \frac{i}{i+1}$
    - Compute the integer part of $\log_2(n)$ for some integer n
    - Print the digits from an integer reversely

14. You must be able to understand and discuss the use of generic types in Java.

15. You must be able to understand and explain the notions of stack (Last-in First-out (LIFO)) as well as Queue (First-in First-out (FIFO)). For each of those ADT you must be able to understand and provide both array based and list based implementations

16. You must be able to understand and explain the notion of ArrayList. You must be able to provide and discuss a simple implementation of an ArrayList ADT.

17. You must be able to understand and explain the notion of Position (and in particular its use within Positional Lists).

18. You must be able to understand and discuss the general implementation of a (linked) Positional List ADT. In particular, you must be able to:

    (a) Understand and discuss the private nested `Node` class

    (b) The private `validate()` method which makes it possible to use a Position to access a particular node in the list.

    (c) You must be able to outline the most important methods (`addBetween()`, `set()` and `remove()`) which should appear in the class.

19. You must be able to understand and describe the notion of `Iterator` as well as how it is implemented through the `Iterable` interface. You must be able to describe how to implement iterators on `LinkedPositionalLists` (on `Positions` and `Elements`), `Binary Trees` (see below) as well for separate chaining (`ChainHashMap`) or linear probing (`ProbeHashMap`) based `Maps` (see below).

20. You must be able to understand and discuss the notion of Tree (including the notions of parent and children nodes, internal and external nodes, ancestors and descendants, depth and height, binary trees, proper and improper binary trees)

21. You must be able to outline the implementation of a binary tree through a linked structure. In particular you must be able to list and explain the following key elements:

    (a) The private nested `Node` class (including the connections of each node to the parent, left and right children nodes).

    (b) The `parent()`, `left()` and `right()` methods which are used to respectively return the parent, left or right child of a node.

    (c) The `createNode()`, `addRoot()`, `addLeft()` and `addRight()` methods which can be used to insert new nodes in the tree.

    (d) The `validate()` method which enables access to the nodes through a `Position`

22. You must be able to explain array-based implementation of binary trees through the level numbering function

23. You must be able to describe the following tree traversal algorithms:

    (a) Preorder traversal

    (b) Postorder traversal

    (c) Breadth-First Traversal

    (d) Inorder Traversal

24. You must be able to understand and explain the notion of Priority Queue and in particular the notion of key and the comparable and comparator interfaces.

25. You must be able to outline the main elements of the implementation of a priority queue:

    (a) The nested `Entry` (or `PQEntry`) class.

    (b) The `insert()` and `removeMin()` methods and how those vary from the sorted to the unsorted implementations

    (c) The `comparator` for the class and the difference between the use of the default comparator (associated to the `compareTo` method) and a user specified comparator, as well as how to implement each.

26. You must be able to understand and explain the notion of Heap. In particular, you must be able to explain the following:

    (a) The Heap-Order and Complete Binary Tree properties

(b) The relation between maximum height and number of nodes (I.e. You must be able to explain how to derive the height from the number of nodes)

27. You must be able to explain how to insert and remove a node from a Heap through Up-Heap and Down-Heap Bubbling.

28. You must be able to compare the running times of the methods `min`, `remove` and `insert` for each of the sorted, unsorted and heap-based implementations of a priority queue.

29. You must be able to understand and explain the notion of Map

30. You must be able to outline the implementation of a simple unsorted map based on Java's ArrayList class. More particularly you must be able to explain how to implement the methods `findIndex()`, `put()` and `remove()`.

31. You must be able to understand the notion of Hash table and the various components of such a table. More particularly, you must be able to explain the notions of

    (a) Hash code (and illustrate the idea with polynomial hash codes or cyclic shift hash codes)

    (b) Compression function (and illustrate the idea with the divison method or the MAD method)

    (c) Bucket Array (and describe the implementation through separate chaining and linear probing)

    (d) The notion of load factor

32. You must be able to explain how to retrieve a particular element from a separate chaining or linear probing map (in particular you must be able to discuss the use of a sentinel node in the framework or linear probing)