COMP 1020
Final exam topics

Lauren Himbeault

Exam topics

- Major area 1: Basic objects/classes
  - From Weeks 3-4
  - A class defines a new object types
  - All object types are stored as a references
  - Actual objects are create by new
  - Instance variables are stored in individual objects
  - Instance methods are always applied to the this object
    - Don't write "this." inside the class unless there's a duplicate name.
  - Constructors are special methods that create objects (no return type, same name as class)
  - Class variables/methods are defined using static
    - One for the entire class, not one per object

Exam topics

- Major area 1: Basic objects/classes
  - Visibility:
    - private – only this class file
    - public – visible everywhere
  - Standard Object methods: toString, equals, compareTo
  - Shallow/deep copies
  - System.arraycopy

Exam topics

- Major topic 2: Files and Exceptions
  - Exceptions are Objects
  - Objects have methods:
    - getMessage(), printStacktrace(), toString()
  - Exceptions can be intentionally created and thrown:
    - throw new ExceptionType("message")
    - Like a return statement – the method is terminated
      - Or just part of it, if you catch it.

Exam topics

- Major topic 2: Files and Exceptions
  - You can catch exceptions with try/catch:
    - try {...code...} catch(Exception e){...}
    - If an Exception is thrown in ...code... then it will be caught, and the catch block {...} is executed.
    - Only the remainder of the ...code... is terminated, not the whole method.
  - You can also add "throws" to the method header.
    - Then any Exception of that type anywhere in the method will be thrown back to the caller. and become the result of this method.
    - The caller then has to catch it (try/catch) or use "throws" itself.

Exam topics

- Major Topic 2: Files and Exceptions
  - File Input:
    - new FileReader(String) or new FileReader(File)
      - opens a low-level input file. Only used to...
    - new BufferedReader(FileReader)
      - "wrap" a FileReader in a BufferedReader.
    - BufferedReader methods:
      - String readLine()
        - returns null for end-of-file
      - int read()
        - reads one character, gives -1 for end-of-file
      - void close()

Exam topics

- Major Topic 2: Files and Exceptions
  - File Output:
    - new FileWriter(String) or new FileWriter(File)
      - opens a low-level output file and erases it.
    - new FileWriter(String,true) or new FileWriter(File,true)
      - opens a low-level output file and appends to it.
    - new PrintWriter(FileWriter)
      - "wrap" a FileWriter in a PrintWriter.
    - PrintWriter methods:
      - The usual println and print methods
      - void close()
    - Choosing a file: File, JFileChooser
    - Not examinable: printf, binary files.

Final exam topics

- Major Topic 3: ArrayLists
  - Hold only Object types
  - ArrayList x; -or- ArrayList<Type> x;
  - For ArrayList x; need instanceof and casting often.
  - Basic methods:
    - add(Object), add(index,Object)
    - remove(Object), remove(index)
    - size()
    - clear()
    - indexOf(Object), lastIndexOf(Object), contains(Object)
    - get(index), set(index,Object)
  - Wrapper classes: Integer, Double, Character, etc.
    - Automatic conversion to int, double, char, etc.

Final exam topics

- Major Topic 4: Strings
  - Write code to perform a range of manipulations on textual data, including:
    - splitting a String according to a very simple expression (e.g., blank space)
    - accessing individual characters
    - accessing substrings.
  - Manipulating strings directly, using scanner etc.

Final exam topics

- Major topic 5: Searching and Sorting
  - Searching: Linear and Binary searches
  - Basic concept of algorithm order
    - O(n) vs. O(log n) vs. $O(n^2)$
  - Slow $O(n^2)$ sorts:
    - Insertion sort:
      - Build up a sorted list one item at a time with ordered insertion.
    - Selection sort:
      - Find the smallest, 2^{nd} smallest, etc.

Final exam topics

- Major topic 5: Searching and Sorting
  - Merge sort O(nlog n):
    - Based on the merge operation
      - Combining two smaller sorted lists into one larger one.
    - Split in half, sort the halves, merge them.
  - Quicksort average O(nlog n) (technically worst case is n^2):
    - Chooses one element to be the pivot
    - Split into ones smaller and larger than the pivot
    - Sort the two pieces

Final exam topics

- Major topic 6: Recursion
  - Call a smaller case of the same method
  - Must be an easy non-recursive base case
  - Works because of the execution stack (stack frames)
  - To write a recursive method:
    - Assume/trust that any smaller case will work
    - Think of how that can be used to solve the larger case
    - Add a base/easy case
  - Tail recursion is equivalent to a loop
  - Recursion is not automatically correct to use (fib(n)!)

Exam topics

- Major area 7: Multidimensional arrays
  - It's just an array of other arrays
  - But the subscripts are backwards in the declaration.
    - int[4][3] means an array of 4 int[3] objects.
  - "Ragged" arrays: Rows can be different sizes.
    - int[4][ ] an array of 4 int[] objects, all null initially.
    - The first size must be specified.
    - Once one is blank, the rest must be blank.

Final exam topics

- Major Topic 8: Linked Lists
  - Node and LinkedList standard classes
    - links and top pointers
  - Easy to add a node at the start (1 or 2 lines)
  - Sequencing through a list (next = next.getLink();)
  - Frequently need the *previous* node (e.g. deletion)
    - Move a *pair* of prev/next references through the list
  - Draw a picture!

Final exam topics

- Major Topic 9: Interfaces/Abstract Data Types
  - **Interface** = a *contract* (method signatures, no implementation)
  - **Abstract Data Type (ADT)** = a *conceptual model* for organizing data and operations
  - Interfaces can represent **ADTs** like Stack, Queue, or List
  - Implementing an interface = *defining how* the ADT works (e.g., with arrays or linked lists)
  - Use or implement **Comparable** or **user-defined interfaces**
  - Understand how **Stacks and Queues** are *used through interfaces*, regardless of how they are built

Final exam topics

- Major Topic 9: Interfaces/Abstract Data Types
  - **Stack**
    - push(), pop(), isEmpty(), size(), peek()
  - **Queue**
    - enqueue(), dequeue(), isEmpty(), size(), peek()

Final exam topics

- Major Topic 10: Analysis of Algorithms
  - **Why Steps Matter**
    - CPU time varies across machines; *step counts* give a consistent way to compare algorithms
    - Focus is on **growth rate** as input size increases
  - **Big-O Basics**
    - Describes **worst-case** runtime in terms of input size n
    - Common examples:
      - Constant time: O(1)
      - Linear time: O(n)
      - Logarithmic time: O(log n)
      - Quadratic time: $O(n^2)$

Final exam topics

Algorithm	Requirements	Big-O Time (Worst)	Notes / Strategy
Linear Search	Unsorted list	$O(n)$	Step-by-step scan
Binary Search	Sorted list	$O(\log n)$	Halves the list each step
Selection Sort	None	$O(n^2)$	Finds smallest, swaps into place
Insertion Sort	Usually small/mostly sorted data	$O(n^2)$	Builds sorted list one item at a time
Merge Sort	None	$O(n \log n)$	Divide and conquer, uses extra space
Quick Sort	None (but best with random/shuffled data)	$O(n^2)$ worst, $O(n \log n)$ avg	Divide and conquer, in-place