CMSC 132: OBJECT-ORIENTED PROGRAMMING II

Miscellaneous/Review

Department of Computer Science
University of Maryland, College Park
IMPORTANT

• Make sure you check your e-mails every day and the messages we post on the class announcements. It is your responsibility to check them so you are aware of important information/deadlines.
• Final exam information is available on the class web page.
• Please complete course evaluations 😊
• Save your projects for future reference. CVS repositories will be deleted after the semester is over.
• FYI: For future advising sessions
  • http://www.cs.umd.edu/~nelson/advising/
FYI: BitSet Class

- Implements a set of bits where the bits of the set are indexed by nonnegative integers.
- We could have used it for Sudoku
- Methods
  - `BitSet()` – New bit set
  - `BitSet(int nbits)` – Bit set large enough to represent bits with indices from 0 through nbits – 1
  - `and(BitSet set)` – Performs logical `and` between the current object and the set parameter (current object is updated with the result)
  - `or(BitSet set)` – Performs logical `or` between the current object and the set parameter (current object is updated with the result)
  - `cardinality()` – Returns number of bits set to 1
  - `flip(int bitIndex)` – Sets the bit at the specified index
  - `get(int bitIndex)` – Returns true if the bit at bitIndex is set; false otherwise
  - `length()` – Index of the highest set bit + 1. It returns zero if the BitSet contains no bits set.
  - `size()` – Number of bits space used by the BitSet to represent bit values
  - `toString()` – For every bit set, the decimal representation of that index is included in the result.
Review

• Note: this is NOT a complete list of the topics for the final. See the final exam information posted on the class web page for complete information.

• Object-Oriented Principles
  • Abstraction
  • How to design system based on provided descriptions

• Algorithmic Complexity
  • Why we use it?
  • What is the alternative?
  • Examples

• Linear Data Structures
  • Traditional linked list (head)
  • Project linked list (head and tail)
  • Example: A method returning a list
Review

- **Note:** this is NOT a complete list of the topics for the final. See the final exam information posted on the class web page for complete information.

- **Sets/Map**
  - Different types
  - Examples

- **Trees**
  - Traditional vs. Polymorphic
  - Examples

- **Software Development**
  - Kinds of testing
  - Software Process Models
  - Software Lifecycle
  - Architectures

- **Multithreading**
  - How to define threaded solutions
  - How to avoid data races
Review

- Note: this is NOT a complete list of the topics for the final. See the final exam information posted on the class web page for complete information.

- **Graphs**
  - BFS/DFS
  - Dijkstra’s

- **Sorting**
  - Performance of each algorithm

- **Algorithm Strategies/Design Patterns**

- **You don’t need to know UML**
  - But you can use it to provide answers if you want.
  - Practice material may have UML exercises. Don’t write the UML, but write the solution to any design problem.

- **Java**
  - Abstract Classes vs. Interfaces
  - Comparable, Comparator
  - Etc.
Questions about Final Exam Material

• Any questions?