Lecture 10: Linked Lists

CS 62
Spring 2015
Kim Bruce & America Chambers

FileIO

- File class:
  - represents a file or directory
  - doesn't have to exist
  - use the File.separator so that it doesn't matter what system we run on.
- Some methods that may be helpful:
  - delete()
  - exists()
  - createNewFile()
  - isFile()
  - isDirectory()
  - listFiles()
  - mkdir()
  - renameTo(...)

More FileIO

- Use the BufferedReader and PrintWriter classes for reading and writing to files.
- Have lots of useful methods
- PrintWriter out =
  new PrintWriter(new FileWriter(...));
- BufferedReader in =
  new BufferedReader(new FileReader(...));

Exceptions

- Many methods/constructors throw exceptions
  - public String readLine() throws IOException
- Handle exceptions by try-catch construct
  - try {
    ... myFile.readLine() ...
    } catch (IOException ex) {
    code to be executed if exception raised
    }

Linked Lists

- Alternate implementation of lists
- Trade-offs in complexity
  - With ArrayList expensive to add at beginning of list
  - Linked lists inexpensive to add early
  - However, slow to access ith element.

Linked List

- Composed of Nodes
  - Think of as pop-beads
  - See code in structure5 library
    - From documentation page!
- See code in SinglyLinkedList
  - *Bailey - not std Java!*
  - keep track of head and size
  - Extends AbstractList -- look at on own!
    - Vector also extends AbstractList
- Also see SinglyLinkedListIterator

Linked List Algos

- Constructor
- addFirst, removeFirst
- get(i)
- indexOf(e)
- add(i,o)
- remove(e), remove(i)
- iterator

What is worst-case complexity of each?

Variants of List

- If add a lot at end, add "tail" pointer
  - Makes adding at end faster
  - But harder to delete at end
  - More special cases -- e.g. add first when empty
  - See implementation when look at queues.
- Circular lists
  - Keep reference/pointer to end rather than beginning
    - What is the difference between adding to end & beginning?
    - getFirst vs getLast?
    - removeLast still hard!
    - How do you know when at end of list if searching?

Doubly-Linked List

- Doubly Linked Lists
  - Previous pointer as well as next
  - Useful if need to traverse in both directions
  - Provided by java.util.LinkedList (but we're using DoublyLinkedList from Bailey)
  - Must change twice as many links when adding or deleting!
  - Our class has head and tail pointers,
    - Doubly-linked lists often represented as circular!

Expectations

- You should be able to write any of these methods in any variant.
- Midterms always include such a question!