Lecture 11: Linked Lists

CS 62
Fall 2017
Kim Bruce & Alexandra Papoutsaki

Piazza

- Two students still not enrolled.
- All important communications to the class will be through Piazza.
  - You are responsible for knowing what has been posted there.

Writing Code

- No complex code ever works first time.
  - If I just fix this last thing ...
- Think about testing before you write the code.
  - Never write more than a method or two without testing it.
- Talk about JUnit in lab next week.

FileIO

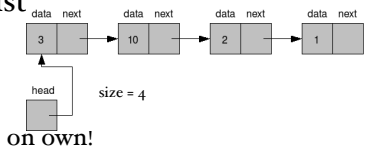
- File class:
  - represents a file or directory
  - doesn't have to exist
  - use the File.separator so that it doesn't matter what system we run on.
- Some methods that may be helpful:
  - delete()
  - exists()
  - createNewFile()
  - isFile()
  - isDirectory()
  - listFiles()
  - mkdir()
  - renameTo(...)

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 snap-lock 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



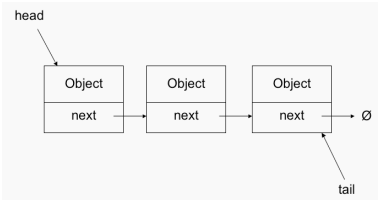
Linked List Algos

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

What is worst-case complexity of each?

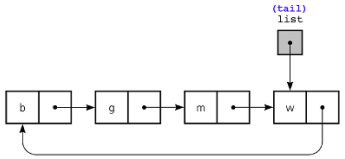
Variants of List

- If add a lot at end, add "tail" pointer
  - Makes adding at end faster
  - But bit harder to delete at end
  - More special cases -- e.g. add first when empty
  - See implementation when look at queues.



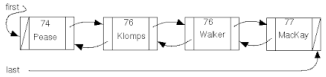
Variants of List

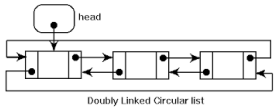
- Circular lists
  - Keep reference/pointer to end rather than beginning
    - What is the difference between adding to end & beginning?
    - getFirst vs getLast?
    - removeLast still hard!
    - How do you know when at end of list if searching?



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!
  - But don't try to memorize them!!!

Compact description of linked list variants:
https://wiki.cs.auckland.ac.nz/compsci105ss/index.php/Linked_Lists