

## Computer Science 136

### Data Structures

#### Lecture #9 (September 29, 2021)

##### 1. Announcements:

- (a) Lab 0 returned. Go to evolene and ok the merge of our grading comments into your project. Details in email from Lida.
- (b) Lab 2 out: Recursion. Several problems, some easy, some more difficult.
- (c) Questions?

##### 2. A design method, using interfaces and abstract classes, in Java.

- (a) Interfaces describe the contract.
- (b) Abstract classes *implement* as much as is possible without committing to a specific approach.

##### 3. The `List<T>`, an important Java interface.

- (a) Includes *many* methods: `size`, `clear`, `isEmpty`, `contains`, `indexOf/lastIndexOf`, `add/remove`, `set/get`. In addition, many convenience routines: `addFirst/addLast`, `removeFirst/removeLast`.
- (b) Notice that many of these methods appear in `Vector`. Java's `Vector<T>` class implements, among other things, the `List<T>` interface.
- (c) `AbstractList` implements many of the convenience methods — methods that may be cast in terms of others.

##### 4. The `Node<T>` class: two logical fields, a `value` and `next`, a link to another `Node<T>`.

##### 5. The `SinglyLinkedList<T>` class (`structure` packages only).

- (a) A complete implementation, based on `Node<T>`.
- (b) Generally implements things iteratively.
- (c) Think about recursive approaches: many require helper methods.

##### 6. Doubly-linked lists.

- (a) Every nodes has two links—one to previous node, the other to the next node.
- (b) Insertion and deletion are a bit more complex and must handle special cases (empty list, or list with one element, or element at one end of list or other).
- (c) But, typically, we keep two pointers in the list: a pointer to the head, and one to the tail.
- (d) Adding a bit more space overhead increases the speed.

##### 7. Potential improvement: Using a dummy node:

- (a) Some of the complexity of handling the base case in linked lists can be avoided with the use of a *dummy node*.
- (b) The dummy node does not hold data, but is a *sentinel* for the end of the list. It avoids always having to check for a null reference.
- (c) Consider the code for removing a node from the middle of a doubly linked list.

##### 8. Aside: `CircularList`, singly linked, but has quick access to tail.

---

#### Notes: