Lecture 01 Introduction

Wednesday, 29 July 2015
12:59 pm

Data structure is an organization of information, usually in memory, for better algorithm efficiency.

Abstract data types (ADTs) are a model for what behaviour a data structure should have.

Lab 01

Wednesday, 8 July 2015
5:14 pm

Compare two variables using assert:

- assertEquals(v1, v2)
- assertGreaterThan(v1,v2)
- assert…

JUnit is an Eclipse tool for unit testing
When submitting to PASTA, zip the 'src' folder.

Lecture 02 Lists and Iterators

Wednesday, 5 August 2015
1:08 pm

Lists

Lists are a collection of objects. Objects must:

- be the same type
- have an **order**

List ADT

size()	Returns number of elements in list
isEmpty()	Returns true if list is empty, else false
get(i)	Returns element at index i
set(i, e)	Replace element at index i with element e
add(i, e)	Insert element e at index i Existing elements are shifted up
remove(i)	Remove AND return element at index i Existing elements are shifted down

ArrayList class implements the List interface
It has the same methods as the list interface.

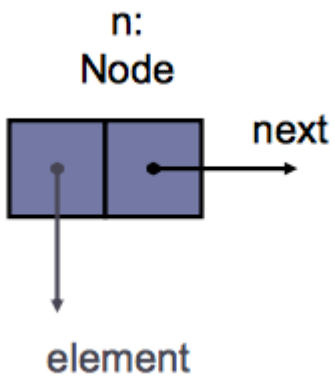
Position based lists

Position is different to an index in that the position stays with an element. i.e. elements can move across indices, whereas elements cannot move away from their position.

Size()	
isEmpty()	
First()	Return position of first element
Last()	Return position of last element
Before(p)	Return position immediately before p
After(p)	Return position immediately after p
Set(p,e)	
Remove(p,e)	
addBefore(p,e)	Insert e in front of element at position p
addAfter(p,e)	Insert e following the element at position p
addFirst(e)	Insert element e at front
addLast(e)	Insert element e at end

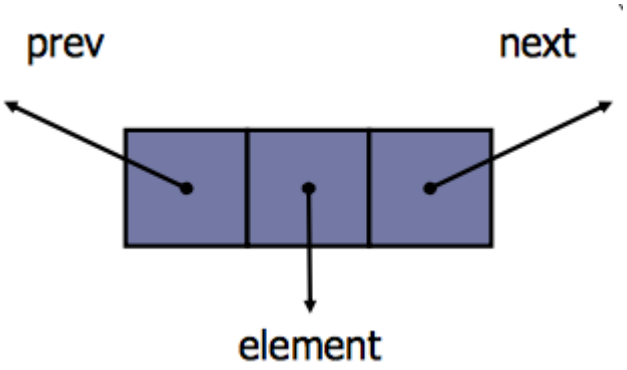
Nodes in a singly linked list:
Each node has

- An element
- A link to the next node

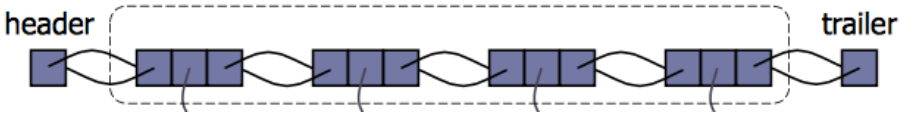


Nodes in a doubly linked list:
Each node as

- An element
- A link to the previous mode
- A link to the next node



This requires sentinel nodes: header and trailer.



Algorithm for inserting a node

addAfter(p,e)	
N.setElement(e)	
N.setPrev(e)	Link n to it's previous
N.setNext(p.getNext())	Link n to its next
(p.getNext()).setPrev(n)	Link p's old successor to n
P.setNext(n)	Link p to new successor, n
Return n	

Differences between Array and Linked List

Array	**Linked List**
• Stored close together for faster access • Allows random access (by index) • Cursor tracked by index i	• Efficient insertion and deletion • Can grow collection • Can shrink collection • No waste of space • Cursor tracked by pointer p to node

Generics

Types can be declared using generic names
These can then be instantiated using actual types

Iterators

Step through a collection of elements one at a time

hasNext()	Checks if there is a next element
next()	Returns the next element (cursor advances)

Iterable types are objects with an iterator() method

Add(e)	Add e at the current cursor position
Set(e)	Replace the element returned by next() or previous() methods
Remove()	Removes the element return by next() or previous() methods
Next()	Return the element e after the cursor and the advance cursor
hasNext()	Boolean, checks whether there is an element after the cursor
Previous()	Return the element e before the cursor and move cursor backward
hasPrevious()	Boolean, checks whether there is a element before the cursor

Lab 02

Saturday, 15 August 2015
12:42 pm

Recursion

- A method that calls itself
- Works for things where you can reduce a problem into a smaller form of itself
- Must have a recursive case to loop, and a base case to end

Week 2 reading

Thursday, 6 August 2015
7:29 AM

Generics

- Not defined at compilation time, becomes specified at run time
- Angle brackets used to enclose parameters

Singly linked lists

- Create a node class
  - Define element
  - Define next node
  - constructor, set and get methods
- Create a singlyLinkedList class
  - Define head and tail
  - Methods for traversing list

Doubly linked lists

- Create a DNode class
  - Define element,
  - Define next node,
  - Define prev node
  - constructor, set and get methods
- Create a doublyLinkedList
  - Define header and trailer
  - Methods for traversing list

Iterators

- Think of them like a pointer/cursor which is sticking into a node
- Keeps track of which node you're up to, and can easily link to the next node
- Efficiently transverses a list