Java Collections

Chapter 13

Collection

A repository for other objects

May be ordered (or not)

Support add and remove, plus others

Abstract Data Type

- collection of data and operations on that data
- well defined interface of operations

Collections may be implemented in various ways

Operations are separated from implementation choice

Perfect for Objects!

Java Collections Framework

JCF is a unified way to represent and manipulate a collection of objects

- Interfaces, Implementation, Algorithms

Abstract requirements for specific types of collections

Robust implementations with lots of useful capabilities (tested and reliable)

Algorithms to perform useful tasks (like searching and sorting) without YOU having to write them

- *These can be more than you really need (since they are intended for everyone)*

Examples: (partial)

- ArrayList
- LinkedList
- Set
- SortedSet
- Queue
- Deque

ArrayList

public class ArrayList<E> extends AbstractList<E> implements List<E>, RandomAccess, Cloneable, Serializable

- get(int)
- size( )
- set (int, E)
- add (int, E)
- remove (int)
- et al

Set

Collection that cannot contain duplicates

- no order implied

add (E)

remove (E)

isMember ( )

isEmpty (E)

union (Set)

intersection (Set)

Deque

Double-ended Queue

can add/remove from either end

addFirst(E)	addLast(E)
removeFirst(E)	removeLast(E)
et al	

Stack

Not part of Collections…Why?

How could you implement a Stack?

- from slides that we have discussed…

Linear Data Structures

stack, queue, array, arraylist etc are "linear" data structures

- elements are organized in a linear fashion

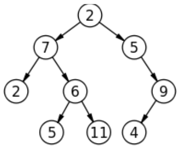
Sets are not (necessarily) linear

Non-Linear Data Structures

Trees and Graphs are examples

Data is not organized linearly

Trees



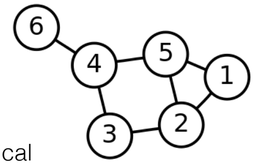
Hierarchical organization of "nodes"

- *(think linked list, but with more than one link!)*
- add, remove, find, etc.
- access child(ren), parent, sibling(s)

Very useful

- Hierarchical organization (organization charts)
- Phrase analysis (sentence structure)
- File organization on computers
- Cryptography and file compression
- Artificial Intelligence (decision trees)
- Used to implement other ADT's (heaps, priority queues, etc)
- etc

Graphs



Connections of "nodes"; NOT hierarchical

Many edges between many nodes

Uses

- Highway systems and logistics (connectivity)
- Electrical circuit simulations
- Path finding and navigation (shortest path)
- Internet message routing
- Airline routing (can you get from here to there?)
- etc

Problem Solving

How might you implement a ***binary tree*** data structure?

- Each node may have 0, 1, or 2 children (left&right)
- Each node (except root) has one parent