Data Structures

what sorts of things do you want to do to the data? add, remove, search, iterate, sort, find min/max

List: ordered collection of elements; may contain duplicates
Set: collection that cannot contain duplicates; no inherent ordering
Map: mapping of keys to values; each key maps to at most one value

Lists
• positional access (get, set, add, remove)
• searching (contains): maintains sequence in which elements were inserted
• iteration

LinkedList vs. ArrayList
• LinkedList has the constant overhead of creating new Object every time you add something; greater than the overhead of having to expand the underlying array in ArrayList
• LinkedList has overhead from get method: need to walk through the list; this is constant time in ArrayList
• LinkedList is better if you need to add a lot of elements to the front (ArrayList is fast at adding them to the back)
• LinkedList is better if you need to remove something from the middle
• ArrayList has a tuning parameter (initial capacity); LinkedList has none
• ArrayList is unsynchronized; Vector is synchronized

Sets
• non-positional access (add, remove... but not get/set)
• searching (contains): sequence not guaranteed to be maintained (it is in some cases, though)
• iteration

TreeSet vs. HashSet
• HashSet is much faster (amortized constant time vs. log time)
• TreeSet orders elements based on values (HashSet ordering is undefined)
• LinkedHashSet: hash table with a linked list connecting the elements, so that you get insertion-ordered iteration (at the cost of some extra management overhead)
• HashSet iteration depends on number of elements and number of buckets, so it's possible that an initial capacity that's too high could waste space and time
• Default initial capacity is 16: internally the value is always rounded up to a power of 2