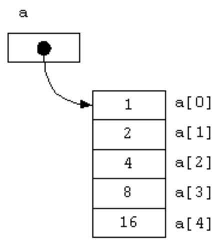
Ch 18.2 Lists

The `List` interface is-a `Collection`. It adds methods to `Collection` that manipulate elements in a `List` using an integer index:

- `void add(int index, E element)` - add element at index position
- `E get( int index)` - get element at index position
- `int indexOf( E element)` - return index of element
- `E remove( int index)` - remove element at index position

There are two flavors: `ArrayList` and `LinkedList`.

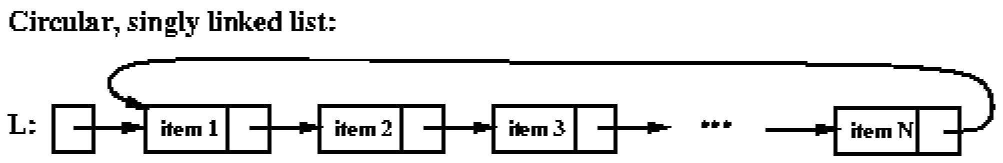
ArrayList
An array stores elements stored in contiguous memory.
+ Efficient traversal, access, and append
- Inefficient insert/remove
- Fixed size



ArrayList is an array that automatically resizes itself as needed.

Use the interface type for your variables. Examples:
```
List<String> cities = new ArrayList<String>();
// use List for method params
foo( List<String> bar) { … }
```

LinkedList
A linked List is a series of connected nodes. Each node holds an element and a pointer to the next node in the list. There are many flavors of linked lists. Please see CSC 210.
+ Efficient traversal, insert/remove
+ No size limit
- Slower access, no array[i]
- More memory



Additional methods specific to `LinkedList`:

- `addFirst( E), addLast( E)` - add element as first/last in list
- `getFirst(), getLast()` - returns the first/last element in list
- `removeFirst(), removeLast()` - remove first/last element in list

Foreach loop is great!

```
List<FruitFly> flies;
// blah blah blah

for( FruitFly ff: flies) {
   // code for each FruitFly, ff
}
```

You can use `ListIterator` for more control. It's most useful when you need to remove a list element from the middle of a list.

```
List<DireWolf> wolves;
// blah

ListIterator<DireWolf> iter = wolves.listIterator();
while( iter.hasNext()) {
   DireWolf wolfie = iter.next();  // get the DireWolf
   System.out.println( wolfie);

   if( something()) {
      iter.remove();   // removes current element from list
   }
}
```

Yeah. Yuk. You can also use the iterator to add an element to the middle of the list.

Try this to see an array and linked list in action… some great Javascript animations of these data structures here:

www.cs.usfca.edu/~galles/visualization/Algorithms.html