Arrays

- What can we do with arrays?
  - Manage collections of information
  - Put a value into an array
    a[i] = 1;
  - Get a value out of an array
    value = a[i];
  - Ask the array what its size is
    size = a.length;

Array Limitation 1

- Fixed in size
- How do we get around this?
  - Make the array as big as we think it ever needs to be
  - Add another data member to the class that remembers how many entries are in the array

Example from Asteroids:

    int bullets[1000];
    int numBullets;

Array Limitation 2

- Adding an element at a specific position (as opposed to replacing an element) may require moving other elements

    public void insertInFront(int value) throws ArrayFullException {
        if (numElements == a.length) {
            throw new ArrayFullException();
        }
        for (int i = numElements; i > 0; i--) {
            a[i] = a[i-1];
        }
    a[0] = value;
    numElements++;
    }

Wednesday, March 6, 13
Removing an element at a specific position may require moving elements to avoid gaps

```java
public int removeFromFront() throws ArrayEmptyException {
    if (numElements == 0) {
        throw new ArrayEmptyException();
    }
    int returnValue = a[0];
    for (int i = 0; i < numElements-1; i++) {
        a[i] = a[i+1];
    }
    numElements--;
    return returnValue;
}
```

Wouldn’t it be nice if...

- Collections could change size as necessary
- Inserting into a specific position was as easy as assigning to an array element
- Removing an element was easy

ArrayLists to the rescue!!

**Interface java.util.List**

- Array-like operations:
  - public void set (int index, Object o)
  - public Object get (int index)
  - public int size()
- Insertion and removal:
  - public void add (Object o)
  - public void add (int index, Object o)
  - public Object remove (int index)
  - public boolean remove (Object o)
- Other useful operations:
  - public boolean contains (Object o)
  - public int indexOf (Object o)

Wednesday, March 6, 13
### Arrays vs Lists

<table>
<thead>
<tr>
<th></th>
<th>Arrays</th>
<th>Lists</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can change size</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Access by position</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Easy insertion &amp; removal</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Elements can be any type</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

#### List Hierarchy

- **List**
  - **AbstractList**
    - **ArrayList**
    - **LinkedList**

### ArrayList

- Implements all methods from List interface
- Positional access like arrays
- Easy insertion & removal
- Search operations
- Can grow as necessary
- Has a capacity in addition to a current size

```java
public ArrayList()
public ArrayList(int initialCapacity)
public void ensureCapacity(int minCapacity)
public void trimToSize()
```
ArrayList Methods

myList — "Vedika" "Anne"
myList.add("Nikki");
myList — "Vedika" "Anne" "Nikki"
myList.add("Mina", 0);
myList — "Mina" "Vedika" "Anne" "Nikki"
myList.set("Ashley", 0) returns "Mina"
myList — "Ashley" "Vedika" "Anne" "Nikki"

ArrayList Methods

myList — "Ashley" "Vedika" "Anne" "Nikki"
myList.contains("Vedika") returns true
myList.indexOf("Anne") returns 2
myList.get(0) returns "Ashley"

ArrayList Methods

myList — "Ashley" "Vedika" "Anne" "Nikki"
myList.remove("Nikki") returns true
myList — "Ashley" "Vedika" "Anne"
myList.remove("Maddie") returns false
myList.remove(0) returns "Ashley"
myList — "Vedika" "Anne"
myList.remove(2) throws IndexOutOfBoundsException
myList.add(5, "Kate") throws IndexOutOfBoundsException

Wednesday, March 6, 13
Declaring and Constructing an ArrayList

// Construct an ArrayList that can hold Strings
ArrayList<String> links = new ArrayList<String>();

Javadoc for Generics

Class ArrayList<E>
   boolean add (E e)
   E get (int index)
   E remove (int index)
   E set (int index, E element)

* There is no type named “E”
* Means whatever type parameter is used in the declaration

ArrayList<String> links;
-> add and set must be passed Strings
-> get, remove and set will return Strings