Warm-up Questions

1. Take a couple pieces of candy if you want
2. Write a method `remove(int index)` for `ArrayIntList` that removes the item at the specified index and shifts all later values left.
3. Where in the book can the material covered on Wednesday be found?
4. How much wood could a woodchuck chuck if a woodchuck knew Java programming?

---

Woodchuck.java

```java
public class Woodchuck extends Mammal {
  int numChucked; // total amount of chucked wood

  /**
   * Construct a woodchuck.
   */
  public Woodchuck() {
    numChucked = 0;
  }

  /**
   * Chuck a single piece of wood.
   * @param wood to be chucked
   */
  public void chuck(Wood wood) {
    if (wood.isChuckable) {
      wood.setChucked(true);
      numChucked++;
    }
  }

  /**
   * Force woodchuck to laugh at your jokes.
   */
  public void chuckle() {
    System.out.println("Ha ha!");
  }
}
```

---

CIS 211, Lecture 3: More ArrayIntList!

April 8, 2011
Daniel Lowd
### Last Time

- Introduced ArrayList as an array that automatically grows as you add stuff to it.
- Started building ArrayIntList from scratch as an example of a collection.
- Key Idea: Use objects!
  - Encapsulate state and behavior
  - Hide messy details behind clean interface (e.g., radio)

### ArrayIntList so far...

Constructor for a default capacity:
```
ArrayIntList list1 = new ArrayIntList();
```
Add items to the list:
```
list1.add(42);
```
Print out:
```
list1.print();
```

### Today: More ArrayIntList!

- Private fields
- toString()
- Multiple constructors
- Inserting and removing items
- Growing automatically
- Other miscellaneous methods

(We may finish some of this on Monday.)

### ArrayList methods (10.1)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add(value)</td>
<td>appends value at end of list</td>
</tr>
<tr>
<td>add(index, value)</td>
<td>inserts given value just before the given index, shifting subsequent values to the right</td>
</tr>
<tr>
<td>clear()</td>
<td>removes all elements of the list</td>
</tr>
<tr>
<td>indexOf(value)</td>
<td>returns first index where given value is found in list (-1 if not found)</td>
</tr>
<tr>
<td>get(index)</td>
<td>returns the value at given index</td>
</tr>
<tr>
<td>remove(index)</td>
<td>removes/returns value at given index, shifting subsequent values to the left</td>
</tr>
<tr>
<td>set(index, value)</td>
<td>replaces value at given index with given value</td>
</tr>
<tr>
<td>size()</td>
<td>returns the number of elements in list</td>
</tr>
</tbody>
</table>
| toString()  | returns a string representation of the list such as 
  | "[3, 42, -7, 15]"                                                          |
### ArrayList methods 2

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addAll (<code>list</code>)</td>
<td>adds all elements from the given list to this list</td>
</tr>
<tr>
<td>addAll (<code>index</code>, <code>list</code>)</td>
<td>(at the end of the list, or inserts them at the given index)</td>
</tr>
<tr>
<td>contains (<code>value</code>)</td>
<td>returns true if given value is found somewhere in this list</td>
</tr>
<tr>
<td>containsAll (<code>list</code>)</td>
<td>returns true if this list contains every element from given list</td>
</tr>
<tr>
<td>equals (<code>list</code>)</td>
<td>returns true if given other list contains the same elements</td>
</tr>
<tr>
<td>iterator()</td>
<td>returns an object used to examine the contents of the list (see later)</td>
</tr>
<tr>
<td>listIterator()</td>
<td></td>
</tr>
<tr>
<td>lastIndexOf (<code>value</code>)</td>
<td>returns last index value is found in list (-1 if not found)</td>
</tr>
<tr>
<td>remove (<code>value</code>)</td>
<td>finds and removes the given value from this list</td>
</tr>
<tr>
<td>removeAll (<code>list</code>)</td>
<td>removes any elements found in the given list from this list</td>
</tr>
<tr>
<td>retainAll (<code>list</code>)</td>
<td>removes any elements not found in given list from this list</td>
</tr>
<tr>
<td>subList (<code>from</code>, <code>to</code>)</td>
<td>returns the sub-portion of the list between indexes from (inclusive) and to (exclusive)</td>
</tr>
<tr>
<td>toArray()</td>
<td>returns the elements in this list as an array</td>
</tr>
</tbody>
</table>