ArrayLists

Using arrays to store data

- Arrays store multiple values of the same type
- Individual values are referred to by index
- Convenient to use
  - a[i] = x;
  - x = a[i];
- Limitations of arrays
  - Fixed in size
  - Don’t indicate which values are valid
  - Often this limitation brings us to use ArrayLists
Lists

- **list**: a collection storing an ordered sequence of elements, each accessible by a 0-based index
  - a list has a **size** (number of elements that have been added)
  - elements can be added to the back, or elsewhere

Java Collections and ArrayLists

- Java includes a large set of powerful **collection** classes.
- The most basic, **ArrayList**, is can store any type of **Object**.
- All collections are in the **java.util** package.
  ```java
  import java.util.ArrayList;
  ```
Type Parameters (Generics)

ArrayList<Type> name = new ArrayList<Type>();

- When constructing an ArrayList, you can specify the type of elements it will contain between < and >.
  - We say that the ArrayList class accepts a type parameter, or that it is a generic class.

ArrayList<String> names = new ArrayList<String>();
names.add("Asa");
names.add("Nathan");

---

ArrayList methods

<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 at given index, shifting subsequent values 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 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>
<tr>
<td>toString()</td>
<td>returns a string representation of the list such as &quot;[3, 42, -7, 15]&quot;</td>
</tr>
</tbody>
</table>
ArrayList methods 2

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>addAll(list)</code></td>
<td>Adds all elements from the given list at the end of this list.</td>
</tr>
<tr>
<td><code>addAll(index, list)</code></td>
<td>Inserts the list at the given index of this list.</td>
</tr>
<tr>
<td><code>contains(value)</code></td>
<td>Returns true if given value is found somewhere in this list.</td>
</tr>
<tr>
<td><code>containsAll(list)</code></td>
<td>Returns true if this list contains every element from given list.</td>
</tr>
<tr>
<td><code>equals(list)</code></td>
<td>Returns true if given other list contains the same elements.</td>
</tr>
<tr>
<td><code>remove(value)</code></td>
<td>Finds and removes the given value from this list.</td>
</tr>
<tr>
<td><code>removeAll(list)</code></td>
<td>Removes any elements found in the given list from this list.</td>
</tr>
<tr>
<td><code>retainAll(list)</code></td>
<td>Removes any elements not found in given list from this list.</td>
</tr>
<tr>
<td><code>subList(from, to)</code></td>
<td>Returns the sub-portion of the list between indexes <code>from</code> (exclusive) and <code>to</code> (inclusive).</td>
</tr>
<tr>
<td><code>toArray()</code></td>
<td>Returns an array of the elements in this list.</td>
</tr>
</tbody>
</table>

Learning about classes

- The Java API Specification is a huge web page containing documentation about every Java class and its methods.
- The link to the API Specs is on the course web site.
Modifying while looping

- Consider the following flawed pseudocode algorithm to remove plural elements from a list:

```java
removeEndS(list) {
    for (int i = 0; i < list.size(); i++) {
        get element i;
        if it ends with an 's', remove it.
    }
}
```

- What does the algorithm do wrong?

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>&quot;she&quot;</td>
<td>&quot;sells&quot;</td>
<td>&quot;seashells&quot;</td>
<td>&quot;by&quot;</td>
<td>&quot;the&quot;</td>
<td>&quot;seashore&quot;</td>
</tr>
<tr>
<td>size</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ArrayList of primitives?

- The type you specify when creating an ArrayList must be an `object` type; it cannot be a primitive type.

- The following is illegal:

```java
// illegal -- int cannot be a type parameter
ArrayList<int> list = new ArrayList<int>();
```

- But we can still use ArrayList with primitive types by using special classes called `wrapper` classes in their place.

```java
ArrayList<Integer> list = new ArrayList<Integer>();
```
Wrapper classes

- A wrapper is an object whose purpose is to hold a primitive value and to provide more functionality.
- Once you construct the list, use it with primitives as normal (autoboxing):

  ```java
  ArrayList<Double> grades = new ArrayList<Double>();
  grades.add(3.2);
  grades.add(2.7);
  ```

<table>
<thead>
<tr>
<th>Primitive Type</th>
<th>Wrapper Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>Integer</td>
</tr>
<tr>
<td>double</td>
<td>Double</td>
</tr>
<tr>
<td>char</td>
<td>Character</td>
</tr>
<tr>
<td>boolean</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

Wrapper classes - continued

- Autoboxing:

  ```java
  ArrayList<Double> grades = new ArrayList<Double>();
  // Autoboxing: create Double from double 3.2
  grades.add(3.2);
  grades.add(2.7);
  double sum = 0.0;
  for (int i = 0; i < grades.size(); i++) {
    //AutoUNboxing from Double to double
    sum += grades.get(i);
  }
  ...
  ```
Looking ahead: Interfaces

- An Java interface specifies which public methods are available to a user.
- A class implements an interface if it provides all the methods in the interface.
- Interfaces allow for a common behavior amongst classes, eg the **Collection interface** is implemented by many classes (LinkedList, ArrayList...).
- ArrayLists implement the Java Collection Interface. Let's go look on line...