Intro to Programming II

ArrayLists

Chris Brooks

Department of Computer Science
University of San Francisco
Last time, we learned about how to use arrays in Java.

Java also provides an ArrayList class that can help manage arrays.

ArrayList is a generic container.

That means that we can use the same container to store different kinds of elements.
An example:

```java
ArrayList band = new ArrayList(3);
band.add(``john'');
band.add(``paul'');
band.add(``george'');
band.add(``ringo'');
```
Notice that we declared an initial array size.

The array is then able to grow dynamically beyond that.

- It’s better to allocate in advance if we can.

We can also remove elements, and access them.

We can also add in the middle of a list: `band.add(1, "ringo");`
8-3: Accessing elements

- `get(index)` lets us access the element at a particular index.
- Elements in an ArrayList are stored as *Objects*.
- This means that we need to cast them back to Strings.
- Can’t store primitives.

```java
String name = (String)band.get(2);
```
8-4: Finding elements

6 We can use indexOf to find where an element is located.
6 remove lets us remove things.

\[ \text{int index} = \text{band.indexOf(``ringo'')} \]
\[ \text{band.remove(index)} \]
8-5: Other methods

also:

- `size()` - tells us how many elements are in the array.
- `isEmpty()` - tells us whether there’s anything in the list.
- `contains(Object o)` - tells us whether an object is in the list.
- `clear()` - removes everything.
If we want to avoid casting elements, we can also define the ArrayList to accept particular sorts of objects:

- Java 1.5 only ...

```java
ArrayList<String> al = new ArrayList<String>(3);
al.add("john");
al.add("paul");
al.add("george");
al.add("ringo");

String name = al.get(2);
System.out.println(name);
```
ArrayList vs arrays

- ArrayLists manage inserting, searching, and removing for you.
- Can grow dynamically.
- Only work with objects, not primitives.
- Provides some nice convenience methods.