8-0: The ArrayList class

- 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.

8-1: The ArrayList class

- An example:
  ```java
  ArrayList band = new ArrayList(3);
  band.add("john");
  band.add("paul");
  band.add("george");
  band.add("ringo");
  ```

8-2: The ArrayList class

- 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

- We can use indexOf to find where an element is located.
- remove lets us remove things.
  ```java
  int index = band.indexOf("ringo");
  band.remove(index)
  ```
8-6: Specifying element types

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);
```

8-7: 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.