CIS 211

ArrayList, ArrayIntList
reading: 10.1, 15.1, 15.2

based on slides created by Marty Stepp
http://www.cs.washington.edu/143/

Warm-Up

Write a method `printIntArray(int[] a)` that prints out an array of ints as a comma-separated list surrounded by brackets.

Example outputs:
[]
[42, 8]
[-1, 2, 0]
[5]

Today’s topic: ArrayList and ArrayIntList (Ch.10.1, 15.1, 15.2)

Answer

Here’s what we came up with in class:

```java
public static void printIntArray(int[] a) {
    System.out.print("[");
    for (int i = 0; i < a.length; i++) {
        System.out.print(a[i]);
        if (i < a.length - 1) {
            System.out.print(",");
        }
    }
    System.out.print("]");
}
```

Goal

We want to maintain a list of objects that will automatically grow as we add to it, e.g.:
- List of students in a class (Blackboard)
- List of events in a calendar (Google Calendar, iCal)
- List of emails in an inbox (Any email program)
Lists

- **list**: a collection storing an ordered sequence of elements
  - each element is accessible by a 0-based **index**
  - a list has a **size** (number of elements that have been added)
  - elements can be added to the front, back, or elsewhere
  - in Java, a list can be represented as an **ArrayList** object

---

Idea of a list

- Rather than creating an array of boxes, create an object that represents a "list" of items. (initially an empty list.)
  
  ```
  []
  ```

- You can add items to the list.
  - The default behavior is to add to the end of the list.
  
  ```
  [hello, ABC, goodbye, okay]
  ```

- The list object keeps track of the element values that have been added to it, their order, indexes, and its total size.
  - Think of an "array list" as an automatically resizing array object.
  - Internally, the list is implemented using an array and a size field.

---

**ArrayList methods (10.1)**

- `add(value)` appends value at end of list
- `add(index, value)` inserts given value just before the given index, shifting subsequent values to the right
- `clear()` removes all elements of the list
- `indexOf(value)` returns first index where given value is found in list (-1 if not found)
- `get(index)` returns the value at given index
- `remove(index)` removes/returns value at given index, shifting subsequent values to the left
- `set(index, value)` replaces value at given index with given value
- `size()` returns the number of elements in list
- `toString()` returns a string representation of the list such as "[3, 42, -7, 15]"

---

**ArrayList methods 2**

- `addAll(list)` adds all elements from the given list to this list
- `addAll(index, list)` adds all elements from the given list to this list (at the end of the list, or inserts them at the given index)
- `contains(value)` returns true if given value is found somewhere in this list
- `containsAll(list)` returns true if this list contains every element from given list
- `equals(list)` returns true if given other list contains the same elements
- `iterator()` returns an object used to examine the contents of the list (seen later)
- `lastIndexOf(value)` returns last index value is found in list (-1 if not found)
- `remove(value)` finds and removes the given value from this list
- `removeAll(list)` removes any elements found in the given list from this list
- `retainAll(list)` removes any elements not found in given list from this list
- `subList(from, to)` returns the sub-portion of the list between indexes from (inclusive) and to (exclusive)
- `toArray()` returns the elements in this list as an array
This week, we're going to build our own version of ArrayList from scratch:

ArrayIntList

Key idea: Represent a list as a partially filled array.

index 0 1 2 3 4 5 6 ... 98 99
value 17 932085 -32053278 100 3 0 0 ... 0 0
size 5

(Chapter 10.1, 15.1, 15.2)