Arrays versus ArrayList

Barb Ericson
Georgia Institute of Technology

Learning Goals

- Understand when you might want to use an array
- Understand when you might want to use an ArrayList
- Understand the major differences between Arrays and ArrayLists
- Understand how to use an Iterator

Arrays

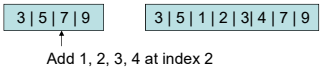
- We use arrays to hold many things of the same type
  - Like sample values in a sound
  - Like pixels in a picture
- We can create arrays of primitive types or objects
  - double[] gradeArray = new double[10];
  - Picture[] pictureArray; // declare the pictureArray
  - pictureArray = new Picture[5]; // create the array
- We don't have to name each element
  - We name the array
  - We use indices to access each element (starting at 0)
    - pictureArray[3];

Benefits of Arrays

- Holds many items that are related and are of the same type
- Quick access to any item based on the index of that item
  - Calculates address based on the size of items in the array and distance from the beginning of the array
- Easy to loop through all items in the array
  - Can use a for-each loop

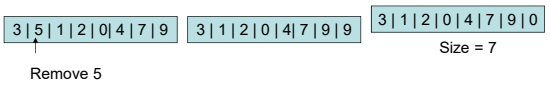
Adding to a Full Array

- To add items into a full array
  - You would need to create a bigger array
  - Copy all the old items before the insert point (if any)
  - Add the new items
  - Copy any remaining old items (ones past the insertion point)



Removing From a Full Array

- You could just copy the items to the left
  - But then you have values at the end that aren't really in the array any more
    - You can zero out primitive types
    - You can set object references to null
      - But what if these are in the array as normal values?
      - You can keep track of the number of valid positions in the array



ArrayList

- Is a implementation of the List interface
  - Using an array
  - The array will grow or shrink as needed to fit the data
  - You can add items to the List
    - They will be added at the end
  - You can add items at a specified index
    - Other items will be moved to the right to make room
  - You can remove items
    - Items will be moved to the left

Differences between Array and ArrayList

- Get the number of items in
  - Array: arrayName.length (public field)
  - ArrayList: size() (method call)
- How you declare them
  - Array: Type[] arrayName (use square brackets)
  - ArrayList: List (use interface name as type)
- How you create them
  - new int[5];
  - new ArrayList();
- Can store
  - Arrays: primitive types or objects
  - ArrayList: only objects (can wrap primitive types)

Interface Name as Type

- Any class that implements an interface *can* be referred to by a variable that is declared with the interface type
  - ArrayList implements the List interface
  - List pictureList = new ArrayList();
  - This allows you to change your mind in the future about which class to use and only change the code in one place
    - Where you create the list
    - Instead of everywhere you specify the type

Processing a List

- If you just want to loop through all items in a list
  - Use a for-each loop
- If you want to remove an item from the list during the loop
  - Use an Iterator and call remove on the Iterator
- If you want to add an item to the list during the loop
  - Use a ListIterator

Processing a List using a While Loop

- You can also process a list using a while loop
  - But be careful if you remove items from the list during the loop
  - Only increment the index if you didn't remove an item from the list

```
int index = 0;
while (index < theList.size()) {
  obj = theList.get(index);
  if (obj.test()) theList.remove(index);
  else index ++;
}
```

- It is easier to use an iterator

Iterators

- Really an interface with these methods:
  - hasNext() returns a boolean value
  - next() returns the next item in the list
  - remove() returns the last returned item from the list

```
Iterator iterator = theList.iterator();
while (iterator.hasNext()) {
  obj = iterator.next();
  if (obj.test()) iterator.remove();
}
```

ListIterator

- An interface
- Inherits from the List interface
- Allows you to process a list from the front or back
  - hasNext() or hasPrevious() to test for elements
  - next() or previous() to get an element
- Can add or set items in the list
  - add(E o) before next or set(E o) to replace last

Summary

- Use arrays to hold groups of data of the same type
  - Especially when the size of the data is known and doesn't change
- Use ArrayList to hold objects in order
  - Especially when you don't know how much data to expect
  - And there may be some additions or deletions
  - There isn't too much data
- Use iterators to process a list
  - Especially if you want to remove or add items during the loop