Plain arrays vs. ArrayList

Tim Hoffman

the ArrayList class, the toArray() method and the Arrays library

ArrayList class contains an array and also contains many methods that operate on the array (add() remove() .get() etc.). ArrayList maintains its own internal count (.size()) and resizes itself automatically as soon as the underlaying array gets full.

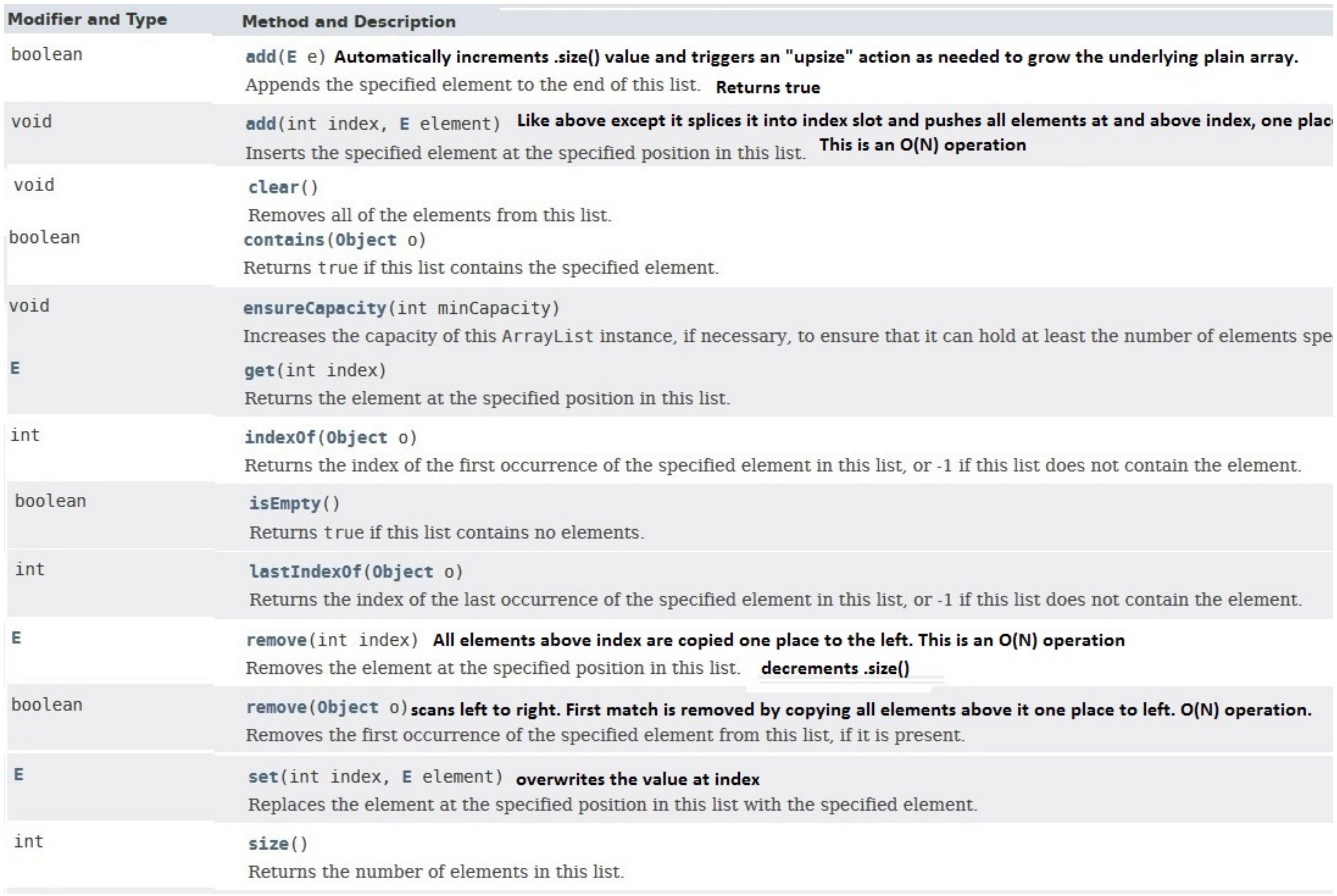
```
ArrayList<String> wordList = new ArrayList<String>();  // ArrayList
```

vs.

```
String[] wordList = new String[ 10 ]; // plain array
```

Notice there are several differences in how a ArrayList is declared vs. how a plain array is declared. A plain array requires the use of []s to specify a dimension for the array. An ArrayList does not require you to specify how big you want the ArrayList to be. This is because a ArrayList can change in size as you use it by letting you .add() elements to the array indefinitely. A plain array on the other hand requires you specify how big the array is and that capacity cannot be changed. The ArrayList will by default declare a plain array of a small capacity then automatically upsize that array as needed without any thought or action on your part. The most commonly used method will be the .add() method that will append a new element to the end of the array. The ArrayList provides several other methods that put new values into the array or remove elements. We will look at them shortly. Notice the **<String>** before the ArrayList constructor. This means your ArrayList is an array of Strings and you can only put strings into it. **You can put nulls into the list but we'll talk later about that and we don't need nulls for this assignment. An ArrayList can be defined to hold any non primitive type.

Some commonly used methods of the ArrayList are :



Modifier and Type	Method and Description
boolean	`add(E e)` **Automatically increments .size() value and triggers an "upsize" action as needed to grow the underlying plain array.** Appends the specified element to the end of this list. **Returns true**
void	`add(int index, E element)` **Like above except it splices it into index slot and pushes all elements at and above index, one plac** Inserts the specified element at the specified position in this list. **This is an O(N) operation**
void	`clear()` Removes all of the elements from this list.
boolean	`contains(Object o)` Returns `true` if this list contains the specified element.
void	`ensureCapacity(int minCapacity)` Increases the capacity of this `ArrayList` instance, if necessary, to ensure that it can hold at least the number of elements spe
E	`get(int index)` Returns the element at the specified position in this list.
int	`indexOf(Object o)` Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
boolean	`isEmpty()` Returns `true` if this list contains no elements.
int	`lastIndexOf(Object o)` Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
E	`remove(int index)` **All elements above index are copied one place to the left. This is an O(N) operation** Removes the element at the specified position in this list. **decrements .size()**
boolean	`remove(Object o)` **scans left to right. First match is removed by copying all elements above it one place to left. O(N) operation.** Removes the first occurrence of the specified element from this list, if it is present.
E	`set(int index, E element)` **overwrites the value at index** Replaces the element at the specified position in this list with the specified element.
int	`size()` Returns the number of elements in this list.

To ask an ArrayList how many elements are currently stored in the array you use the .size() method. To ask a plain array how many elements are currently stored in the array you use youriown count variable which you have to maintain. Interstingly an ArrayList will not let you ask it how big (capacity) the underlying plain array inside the class is. The user is not supposed to be worrying/thinking about that detail since the array is flexible and will grow as needed. You are to assume it will resize the underlying array as needed with no need for you to do anything.

The first and most obvious advantage of ArrayList over plain arrays is that you don't have to make any guesses about how big to initialize it to and you don't have to worry about running out of space or resizing it. You can just keep adding more elements to the ArrayList until your input is consumed. Let lok at how you *might use anArrayList to read in all the dictionary words into your List. We will use the .add() method which is built into the ArrayList class.*

Let's look at : ArrayListDemo1.java to see how to declare, use and print out the values in a ArrayList

Here are two input files use with our demo program. Put either file name on the command line: jumbles.txt OR dictionary.txt

Let's look at the ArrayList API to see the members of the the ArrayList.