Notes – ArrayList

java.util.ArrayList allows for expandable arrays.

An ArrayList has the following advantages over an array:

- An ArrayList automatically expands when needed.
- An ArrayList has methods for inserting, deleting, and searching.

Use an ArrayList when there will be a large variation in the amount of data that you would put into an array. Arrays are used only if there really is a constant amount of data. For example, storing information about the days of the week should use an array because the number of days in a week is constant. Use an array list for your email contact list because there is no upper bound and the number of contacts can be constantly changing.

A disadvantage of an ArrayList is that it holds only Objects (eg, Integer) and not primitive types (eg, int). Luckily with Java 1.5 comes autoboxing—this means that Java will automagically convert an int to and Integer and a double to a Double for you. Lucky you!

To use an ArrayList
In order to use an ArrayList, you must include the following line at the top of your program:

```
import java.util.ArrayList;
```

ArrayLists are really just Arrays that resize
ArrayLists are implemented with an underlying array, and when that array is full and an additional element is added, a new array is allocated and the elements are copied from the old to the new. Because it takes time to create a bigger array and copy the elements from the old array to the new array, it is a faster to create an ArrayList with a size that it will commonly be when full. Of course, if you knew the final size, you could simply use an array. However, for non-critical sections of code programmers typically don't specify an initial size.

To create an ArrayList

```
ArrayList<Integer> alist = new ArrayList<Integer>(); //holds Integers
ArrayList<String> alist2 = new ArrayList<String>(); //holds Strings
```

ArrayList constructors

Result	Constructor	Description
alist =	`new ArrayList()`	Creates ArrayList with initial default capacity 10.
alist =	`new ArrayList(`*capacity*`)`	Creates ArrayList with initial int capacity *capacity*.

Note: Capacity is not the same this as size! Capacity is how many elements can be stored in the underlying native-array before it must be resized. Size is how many elements are actually stored in the ArrayList. When an ArrayList is first created, the size is always zero, regardless of what the initial capacity is.

When declaring an ArrayList, if you put nothing in the ()'s at the end of the declaration then the default capactiy of 10 is used. If you put a number in the ()'s, then that capacity is used. Regardless, the size of the ArrayList immediately after creation is 0.

To add elements to the end of an ArrayList

```
alist2.add("hello"); // adds to the end of the ArrayList alist1
```

To find out how many elements are in an ArrayList

Find out how many Objects are in the ArrayList using the method size(). This is similar to length for an array and length() for Strings.

```
alist2.size()
```

To get the elements from an ArrayList

Use the get method to get just an element from an ArrayList. Remember indexing begins at 0.

```
// gets and prints the first element.
System.out.println( alist1.get(0) );
```

Downcasting

If you don't use generics to specific the datatype being stored in the ArrayList, then you will have to **downcast** the Object returned to its specific type. To downcast the statement above, you would do the following:

```
System.out.println( (String)alist1.get(0) );
```

So to make your life easier, always specify the datatype you will be storing in your ArrayLists!

To print out an ArrayList

Use a for loop with an integer index to get all the elements from an ArrayList and print them out.

```
for (int i = 0; i < alist1.size(); i++)
{
    System.out.println(alist1.get(i));
}
```

Or just say

```
System.out.println(ArrayListName);
```

Sorting and Searching

Call the Collections.sort(*yourArrayList*) or Collections.sort(*yourArrayList, yourComparator*). Check out Collections for other useful utility methods.

Common ArrayList methods
Here are some of the most useful ArrayList methods.

a is an ArrayList, *i* is an int, *obj* is an Object.

Result	Method	Description
	`a.add(obj)`	adds *obj* to end of ArrayList *a*
	`a.add(i, obj)`	Inserts *obj* at index *i*, shifting elements up as necessary.
	`a.clear()`	removes all elements from ArrayList *a*
b =	`a.contains(obj)`	Returns true if ArrayList *a* contains *obj*
obj =	`a.get(i)`	Returns the object at index *i*.
i =	`a.indexOf(obj)`	Returns index of first occurrence of *obj*, or -1 if not there.
i =	`a.lastIndexOf(obj)`	Returns index of last occurrence of *obj*, or -1 if not there.
	`a.remove(i)`	Removes the element at position *i*.
	`a.removeRange(i, j)`	Removes the elements from positions *i* thru *j*.
	`a.set(i,obj)`	Sets the element at index *i* to *obj*.
i =	`a.size()`	Returns the number of elements in ArrayList *a*.
oarray =	`a.toArray(Object[])`	The array parameter can be any Object subclass (eg, String). This returns the values in that array (or a larger array if necessary). This is useful when you need the generality of an ArrayList for input, but need the speed of arrays when processing the data.

Comprehension Questions:

1. **When do you use an ArrayList?**
2. **Give one advantage of an ArrayList.**
3. **Give one disadvantage of an ArrayList.**
4. **What is the underlying structure of an ArrayList?**
5. **What happens when the underlying structure becomes full?**
6. **What must you import in order to use ArrayLists?**
7. **Give the code to create a new ArrayList called flowers.**
8. **If you do not set the capacity of an ArrayList, what is the initial capacity given?**
9. **Give the code to add "Daisy" to the ArrayList flowers.**
10. **How can you find out how many Objects are stored in the ArrayList classroom?**
11. **Get the 3rd Object from the ArrayList a and store the value in the variable String s?**
12. **How would you print out all the String Objects stored in the ArrayList fileNames?**
13. **What does the word downcast mean?**