

CS139 ArrayLists



Declaring an ArrayList

- ArrayList is a **generic** type.
 - Declaration determines the type of object it will store.
- Also referred to as **type-safe**.
- Declaring an ArrayList to store String objects:

generic type invocation

```
ArrayList<String> myList;
```

type argument

Instantiating an ArrayList

```
ArrayList<String> myList; // Declaring.
```

```
myList = new ArrayList<String>(); // Instantiating.
```

OR we can leave out the type argument during initialization...

```
myList = new ArrayList<>(); // Instantiating.
```

Don't forget the parentheses!

We are calling the constructor method of the ArrayList class.

ArrayLists and Primitive Types

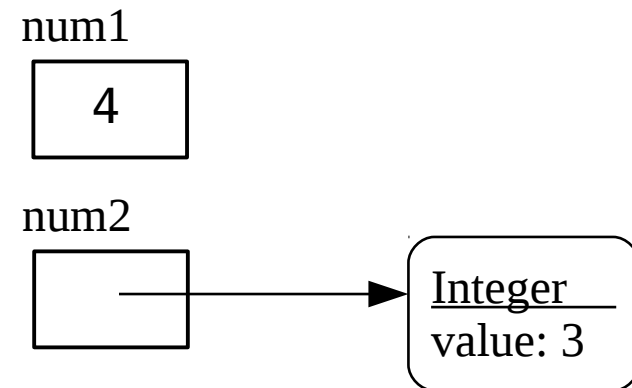
- ArrayLists can only store reference types, not primitive types

```
ArrayList<int> myList = new ArrayList<int>();
```

ArrayLists and Primitive Types: Solution

- Each primitive type has a wrapper class that can be used instead.

```
int num1;  
Integer num2;  
  
num1 = 4;  
num2 = new Integer(3);
```



ArrayLists and Primitive Types: Solution

- Each primitive type has a wrapper class that can be used instead.
- Java converts back and forth as needed:
autoboxing

```
ArrayList<Integer> myInts = new ArrayList<Integer>();  
  
myInts.add(3);  
  
int num = myInts.get(0);
```

Arrays vs. ArrayList

<code>array.length</code>	The <i>capacity</i> of the array.	<code>list.size()</code>	The number of elements stored in the ArrayList.
<code>String val = array[0];</code>		<code>String val = list.get(0);</code>	
<code>array[0] = "HI";</code>		<code>list.set(0, "HI");</code>	
	Not possible to resize array.	<code>list.add("Bob");</code>	Adds a new element at the end of the list. (This changes the size.)
	Not possible to resize array.	<code>list.remove(4);</code>	Removes the element at index 4. (all elements after 4 shift left.)

What will be printed by the following code?

```
ArrayList<String> letters = new
ArrayList<String>();

letters.add("A");
letters.add("B");
letters.add("C");
letters.remove(1);

System.out.println(letters.size());

for (String letter : letters) {
    System.out.println(letter);
}

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


What will be printed by the following code?

```
ArrayList<String> letters = new
ArrayList<String>();

letters.add("A");
letters.add("B");
letters.add("C");
letters.remove(1);

System.out.println(letters.size());

for (String letter : letters) {
    System.out.println(letter);
}

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

Output: 2
A
C
A
C