

CISC 3115

Array, ArrayList, Arrays, and Collections

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Discussed in previous modules
 - Math, Random, Date & Time
 - Wrapper classes of the primitive data types
 - BigInteger, BigDecimal
 - String, StringBuilder, StringBuffer
- ArrayList, array, Arrays, and Collections

ArrayList: Motivation

- What is an array?
 - A collection of items, typically indexed, with equal random access time.
- What are the limitations of array?
 - The size of an array must be known when the array is being created, and is fixed.
- Vector
 - (Mostly, if not all) random access takes equal amount of time
 - It can grow and shrink as needed
 - Java's [java.util.ArrayList](#) is the realization of the vector data structure.

The ArrayList Class

java.util.ArrayList<E>

```
+ArrayList()  
+add(o: E) : void  
+add(index: int, o: E) : void  
+clear(): void  
+contains(o: Object): boolean  
+get(index: int) : E  
+indexOf(o: Object) : int  
+isEmpty(): boolean  
+lastIndexOf(o: Object) : int  
+remove(o: Object): boolean  
+size(): int  
+remove(index: int) : boolean  
+set(index: int, o: E) : E
```

Creates an empty list.

Appends a new element `o` at the end of this list.

Adds a new element `o` at the specified index in this list.

Removes all the elements from this list.

Returns true if this list contains the element `o`.

Returns the element from this list at the specified index.

Returns the index of the first matching element in this list.

Returns true if this list contains no elements.

Returns the index of the last matching element in this list.

Removes the element `o` from this list.

Returns the number of elements in this list.

Removes the element at the specified index.

Sets the element at the specified index.

Remark: Generic Type

- ArrayList is known as a generic class with a generic type E.
 - One can specify a concrete type to replace E when creating an ArrayList.
 - This type indicates the type of objects to which the list contains references to
 - Example:

```
ArrayList<String> cities = new ArrayList<String>();
```
 - This ArrayList object can be used to store (the references to) strings

Differences and Similarities between Arrays and ArrayList

<i>Operation</i>	<i>Array</i>	<i>ArrayList</i>
Creating an array/ArrayList	<code>String[] a = new String[10]</code>	<code>ArrayList<String> list = new ArrayList<>();</code>
Accessing an element	<code>a[index]</code>	<code>list.get(index);</code>
Updating an element	<code>a[index] = "London";</code>	<code>list.set(index, "London");</code>
Returning size	<code>a.length</code>	<code>list.size();</code>
Adding a new element		<code>list.add("London");</code>
Inserting a new element		<code>list.add(index, "London");</code>
Removing an element		<code>list.remove(index);</code>
Removing an element		<code>list.remove(Object);</code>
Removing all elements		<code>list.clear();</code>

Iterating ArrayList

- Use the for loop

- Example

```
for(int i=0; i<arrayList.size(); i++) {  
    ...  
}
```

- Use the enhanced for loop

- Example

```
for (Fruit fruit: fruitList) {  
    ...  
}
```

- Use the forEach method of the ArrayList

Arrays to Array Lists

- Creating an ArrayList from an array of objects:
 - Using the [java.util.Arrays](#) class
 - Example

```
String[] array = {"red", "green", "blue"};
```

```
ArrayList<String> list =
```

```
    new ArrayList<String>(Arrays.asList(array));
```


Array Lists to Arrays

- Creating an array of objects from an ArrayList:
 - Using ArrayList's size() toArray method
 - Example

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

```
.....
```

```
String[] array1 = new String[list.size()];
```

```
list.toArray(array1);
```

Using the Collections Class

- The [java.util.Collections](#) class consists exclusively of static methods that operate on or return collections, e.g., ArrayList
- Example: min, max, shuffle an ArrayList

ArrayList: min and max

- Example: min

```
String[] array = {"red", "green", "blue"};
```

```
System.out.println(Collections.max(  
    new ArrayList<String>(Arrays.asList(array))));
```

- Example: min

```
String[] array = {"red", "green", "blue"};
```

```
System.out.println(Collections.min(  
    new ArrayList<String>(Arrays.asList(array))));
```

ArrayList: shuffle

- Example: shuffle

```
Integer[] array = {3, 5, 95, 4, 15, 34, 3, 6, 5};
```

```
ArrayList<Integer> list = new
```

```
    ArrayList<>(Arrays.asList(array));
```

```
java.util.Collections.shuffle(list);
```

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

Example: Design Stack Using ArrayList

- Observe the animation
 - <https://liveexample.pearsoncmg.com/dsanimation/StackBook.html>

Questions

- Concept of array and vector
- Java's ArrayList
- Two utility classes
 - Arrays and Collections

Exercise

- This exercise builds from the previous exercise where we implemented a hierarchy of fruits.
 - Create subdirectory/subfolder for the exercise
 - Make sure each class in the hierarchy has its own implementation of the toString() method
 - Add an equal method and a hashCode method to each class, and make sure the contract of the hashCode method is maintained.
 - Revise the GoldenDelicious and Macintosh classes so that they cannot be subclassed.
 - In your client class, create an ArrayList of fruits of various kind.
 - Print the list of fruits
 - Take two random fruits out of the list, and compare whether they are equal

Exercise

- Create a Stack using ArrayList (Section 11.13, Listing 11.10) and use the Stack to evaluate postfix expressions, e.g.,
 - $3\ 4\ -\ 5\ +$
 - $3\ 4\ \times\ 5\ 6\ \times\ +$