

CISC 3120

C06: Java API & Libraries

Hui Chen

Department of Computer & Information Science

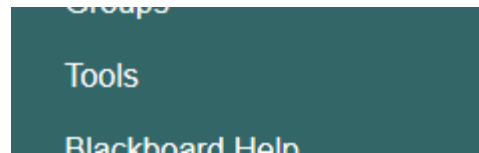
CUNY Brooklyn College

Outline

- Recap and issues
 - Grades and feedback
 - Java classes and objects
 - Java methods and flow controls
- Java API and libraries
- Assignments

Q: How am I doing?

- Check your grades often in CUNY Blackboard



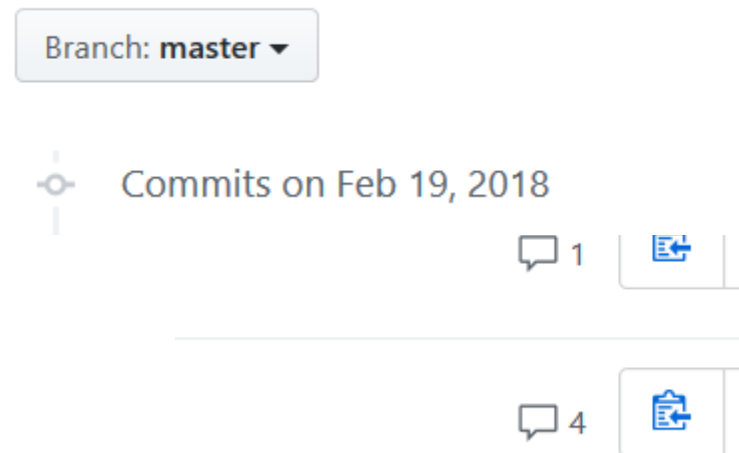
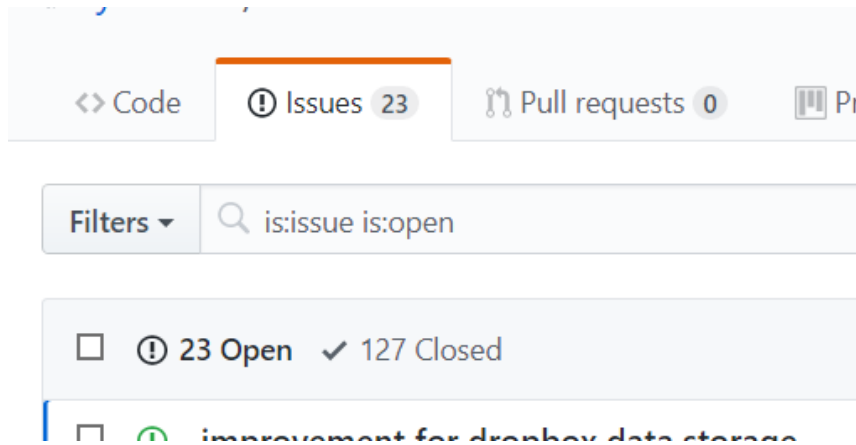
My Grades

Hide Link

Displays detailed information about your grades.

Q: How well did I do in an assignment?

- Feedback channel
 - Github issues
 - Github commit comments



Q: How can I improve?

- "I must get an 'A!'"
- Solution
 - Complete all assignments
 - Squeeze more time
 -

Recap

- Discuss Java from perspective of a C++ developer
- Classes and objects
 - How do we write Java classes?
 - How do we create Java objects?
 - How do Java objects work together?
- Flow controls
 - Selections, iterations, break, continue, and return

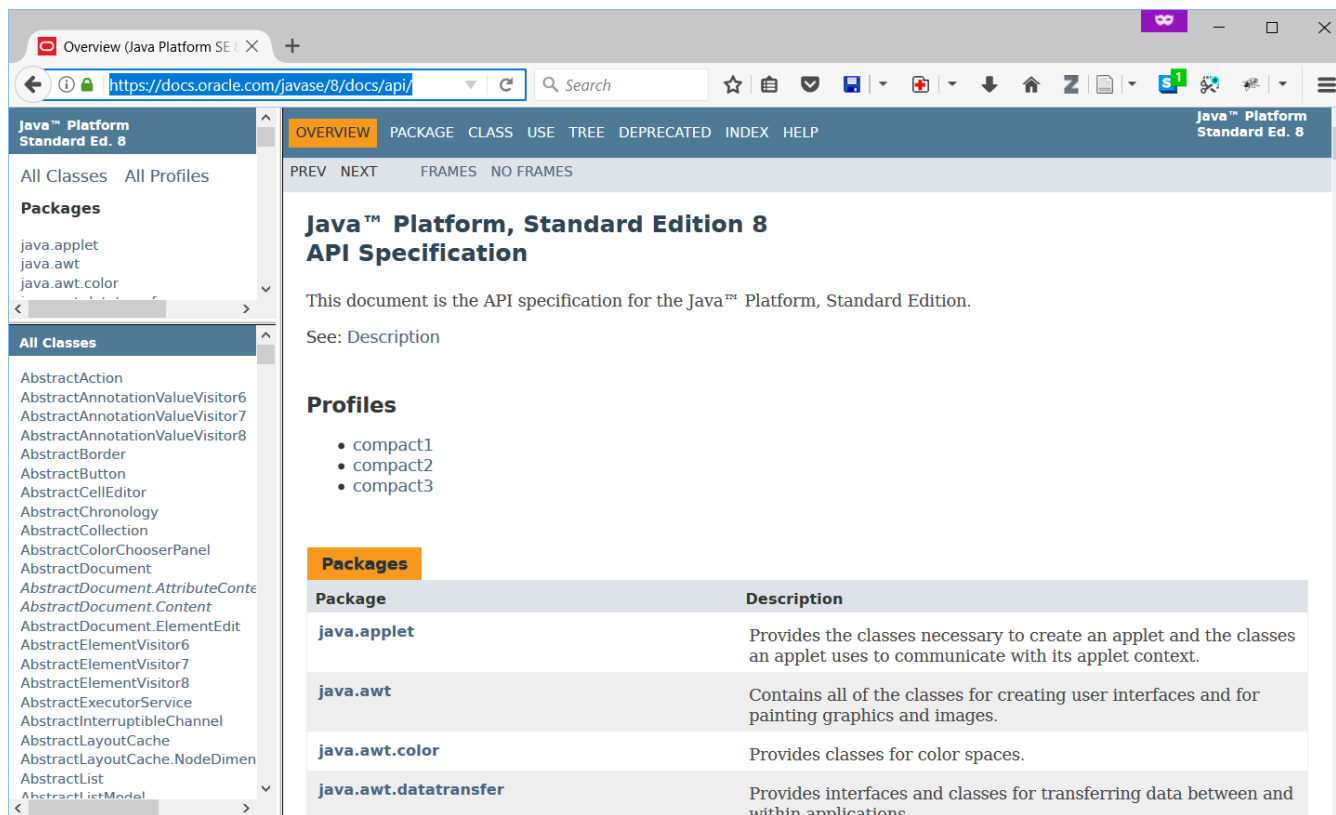
Java API and Libraries

“If I have seen further than others, it is by standing upon the shoulders of giants.”

-- Isaac Newton

Java APIs and Libraries

- <https://docs.oracle.com/javase/8/docs/api/>



The screenshot shows the Java Platform Standard Edition 8 API Specification website. The browser address bar displays the URL <https://docs.oracle.com/javase/8/docs/api/>. The page title is "Java™ Platform, Standard Edition 8 API Specification". The main content area includes a navigation menu with options like "OVERVIEW", "PACKAGE", "CLASS", "USE", "TREE", "DEPRECATED", "INDEX", and "HELP". Below the navigation, there is a section for "Profiles" with a list of options: compact1, compact2, and compact3. A "Packages" section is also visible, listing several packages with their descriptions:

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.

Use Java API Documentation

- API documentation is divided into packages
- The documentation of a class has a few sections
 - Examples
 - Explore: [Math](#) and [Random](#)
- When search using a Web search engine, include "Java", "api", and version
 - Examples
 - [java api 8 arrays](#)

Documentation Organization

- Package & class
- Description of the class
- Fields
- Constructors
- Methods

Questions?

- Use Java API documentation
 - Packages
 - Classes
 - Documentation structure
 - Navigate and search Java API documentation

Java API: Arrays and ArrayLists

- Arrays and lists are common data structures
- Basis to build more sophisticated data structures
 - Examples: queues, stacks, trees, graphs
- Examples in the Sample Programs repository

Array

- A collection of elements of the same data type
 - Each element has an index
 - Each element is accessed via the index at a constant time
- An array generally has a fixed size
 - Implication
 - Need to know beforehand how many elements there are.
 - What if we don't know it?
 - Over-provisioning

Examples in Java: 1-Dimensional

- Declaration and initialization
 - `int[] numbers = new int[10];`
 - `Dog[] dogs = new Dog[5];`
 - `byte[] bytes;`
 - `bytes = new byte[5];`
- Accessing element
 - `numbers[1] * 2.0`
 - `dogs[dogs.length-1].bark()`
 - `byte[0] = 64;`

Examples in Java: N-Dimensional

- 2-dimensional

- Declare and initialize a 2-dimensional array and assign random numbers to each element

```
double[][] numbers = new double[2][3];
for (int i=0; i<2; i++) {
    for (int j=0; j<3; j++) {
        numbers[i][j] = Math.random();
    }
}
```

Examples in Java: N-Dimensional

- 3-dimensional

- Declare and initialize a 3-dimensional array and assign random numbers to each element

```
int[][][] numbers = new int[2][3][2]; Random rng = new Random();
for (int i=0; i<2; i++) {
    for (int j=0; j<3; j++) {
        for (int k=0; k<2; k++) {
            numbers[i][j][k] = rng.nextInt(100);
        }
    }
}
```


Examples in Java: N-Dimensional

- N-dimensional: more examples

```
int[][][][] numbers = new int[2][3][9][2];
```

```
int[][][][][] moreNumbers = new int[3][4][5][2][8];
```

Array of Arrays

- Java and C++
 - Java does not really have a N-dimensional array as C++ does
 - In C++, all the elements of the array occupy a continuous block of memory
 - What Java has is in effect an array of arrays

Example: Array of Arrays

- Observe the following

```
int[][] numbers = new int[3][]; Random rng = new  
Random();
```

```
for (int i=0; i<3; i++) {
```

```
    numbers[i] = new int[rng.nextInt(100)+1];
```

```
}
```

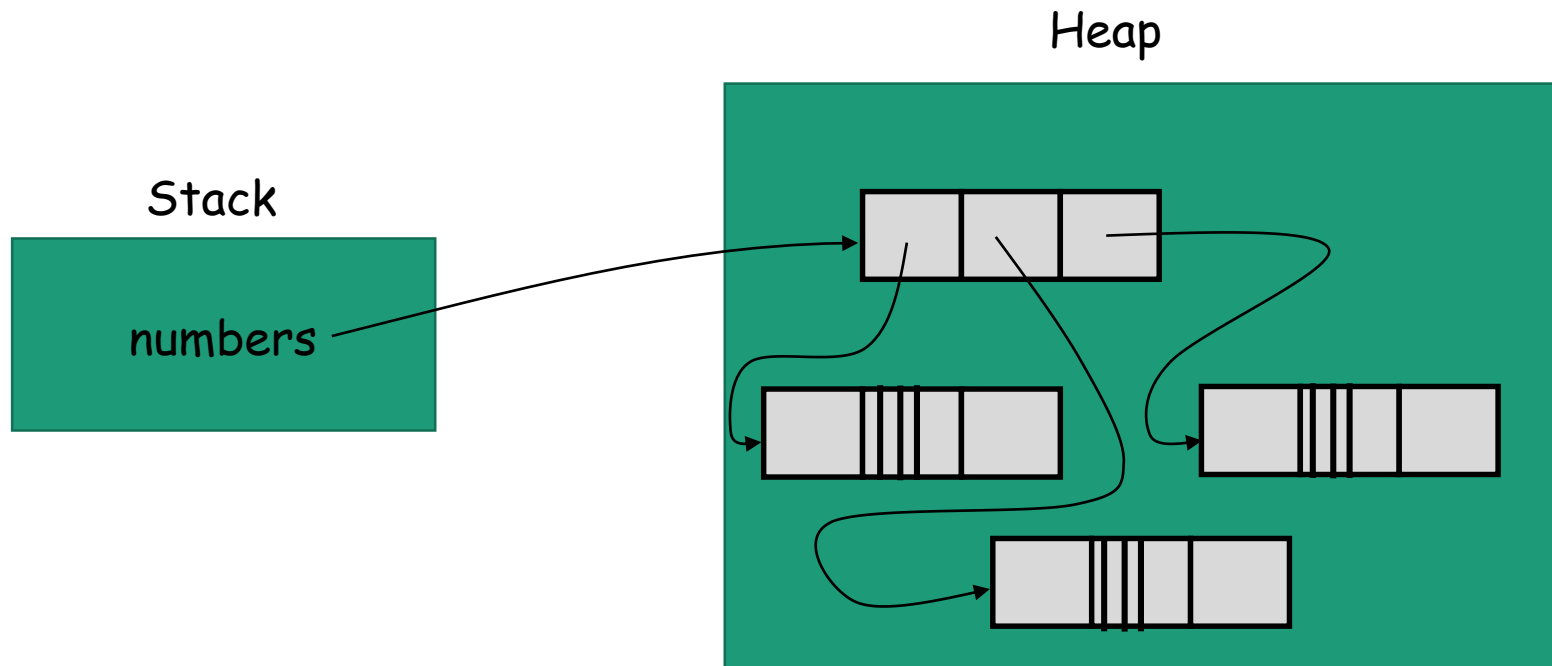
```
for (int i=0; i<3; i++) {
```

```
    System.out.println(numbers.length);
```

```
}
```

Interpreting Array of Arrays

- A Java array is an object, each can be an array object itself



Iterating Array of Arrays

- A better method

```
int[][][] numbers = new int[2][3][2]; Random rng = new  
Random();
```

```
for (int i=0; i<numbers.length; i++) {  
    for (int j=0; j<numbers[i].length; j++) {  
        for (int k=0; k<numbers[i][j].length; k++) {  
            numbers[i][j][k] = rng.nextInt(100);  
        }  
    }  
}
```

Java Array Operations

- The Java Arrays class
 - Copying and cloning
 - Insertion and deletion
 - Searching and sorting

Copying and Cloning Arrays of Primitive Types

- Use an iteration
 - Any dimensions
- Use `Arrays.copyOf()`
 - One dimension
- Use `System.arraycopy()`
 - One dimension
- Use `clone()`

Example: Using Iteration

```
int[][] copyOfNumbers = new int[numbers.length][];  
for (int i=0; i<numbers.length; i++) {  
    copyOfNumbers[i] = new int[numbers[i].length];  
    for (int j=0; j<numbers[i].length; j++) {  
        copyOfNumbers[i][j] = numbers[i][j];  
    }  
}
```


Example: Using `Arrays.copyOf()` and `Systems.arraycopy()`

- Using `Arrays.copyOf()`

```
int[][] copyOfNumbers = Arrays.copyOf(numbers, numbers.length);
```

- Using `Systems.arraycopy()`

```
int[][] copyOfNumbers = new int[numbers.length][];
```

```
System.arraycopy(numbers, 0, copyOfNumbers, 0, numbers.length);
```

Example: Using Array Object's clone() Method

- Must cast the return value to the proper data type

```
int[][] copyOfNumbers = (int[][]) numbers.clone();
```

Arrays.copyOf() and Systems.arraycopy()

- Arrays.copyOf() return a new array
- Systems.arraycopy requires the destination array object is already allocated
- Arrays.copyOf() uses Systems.arraycopy() under the hood
 - Basically, allocate a new array, and invoke Systems.arrayCopy()

Array clone() and Arrays.copyOf()

- Consider `Arrays.copyOf()` as an improved version of `Array clone()`
 - `Arrays.copyOf()` can resize the array with truncation or padding
 - When using `clone()`, one has to cast the return value of `clone()`

What should I use?

- Use `Arrays.copyOf()`
- If the destination array is already allocated, use `Systems.arraycopy()`
- When copying objects, you may have to write own code
 - Use iterations or/and the combinations of the above
 - Revisit when discussing `ArrayList`

Copying and Cloning of Array of Objects

- Deserve a further investigation
- Are you copying objects or references of objects?
- Revisit when discussing ArrayList

How to find answers like these?

- API documentation
 - Sometimes still insufficient
- Question & answer sites
 - Example: Stack Overflow
 - Always take with a grain of salt
- Examine the JDK source code (OpenJDK)
 - <http://hg.openjdk.java.net/jdk8/jdk8/>
 - Example:
 - <http://hg.openjdk.java.net/jdk8/jdk8/jdk/file/tip/src/share/classes/java/util/Arrays.java#l3202>

Inserting and Deleting Elements in Array

- Array's length is immutable
 - Without making a copy or clone, one cannot truly insert or delete elements in array
- Inserting element
 - Use over-provisioning
 - Make copy
- Deleting element
 - Marking unused element
 - Make copy

Searching

- Sequential search
 - Write your own iteration
 - Does not require the array is sorted
 - Time: $O(n)$
- Binary search
 - Use `Arrays.binarySearch(...)`
 - Requires that the array is sorted
 - Time: $O(\log(n))$

Sorting

- Use `Arrays.sort()`

Questions

- Arrays
 - Characteristics
 - Common operations
- The Arrays class

Limitation of Arrays

- Length is immutable
- What if we don't know the length beforehand?

ArrayList

- Java ArrayList is in effect a vector
 - Have similar characteristics as array when accessing an element
 - Dynamically grow size when needed
 - A little bit slower than arrays
 - More space than arrays

Array and ArrayList

- Array

- Example

- `String[] fruits = new String[5];`
 - `fruits[0] = new String("Apple");`
 - `System.out.println(fruits[0]);`

- ArrayList

- Example

- `ArrayList<String> fruitList = new ArrayList<String>();`
 - `fruitList.add(new String("Apple"));`
 - `System.out.println(fruitList.get(0));`

ArrayList

- An resizable-array implementation of a list
- where a list is an ordered sequence of elements
- In Java
 - Array-like characteristics
 - List-like access interface (methods you can invoke)
- Declaring and initializing
- Copying and cloning
- Insertion and deletion
- Searching and sorting

Creating ArrayList

- Constructor and Description
 - ArrayList()
 - Constructs an empty list with an initial capacity of ten.
 - ArrayList(int initialCapacity)
 - Constructs an empty list with the specified initial capacity.
 - ArrayList(Collection<? extends E> c)
 - Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.

Example: Creating ArrayList

- First try:
 - `ArrayList dogs = new ArrayList();`
 - Correct, but the compiler complains: what data type is of each element?
 - Use generics (we will discuss in the future, but you will get used to it before the discussion)
- `ArrayList<Dog> dogs = new ArrayList<Dog>();`

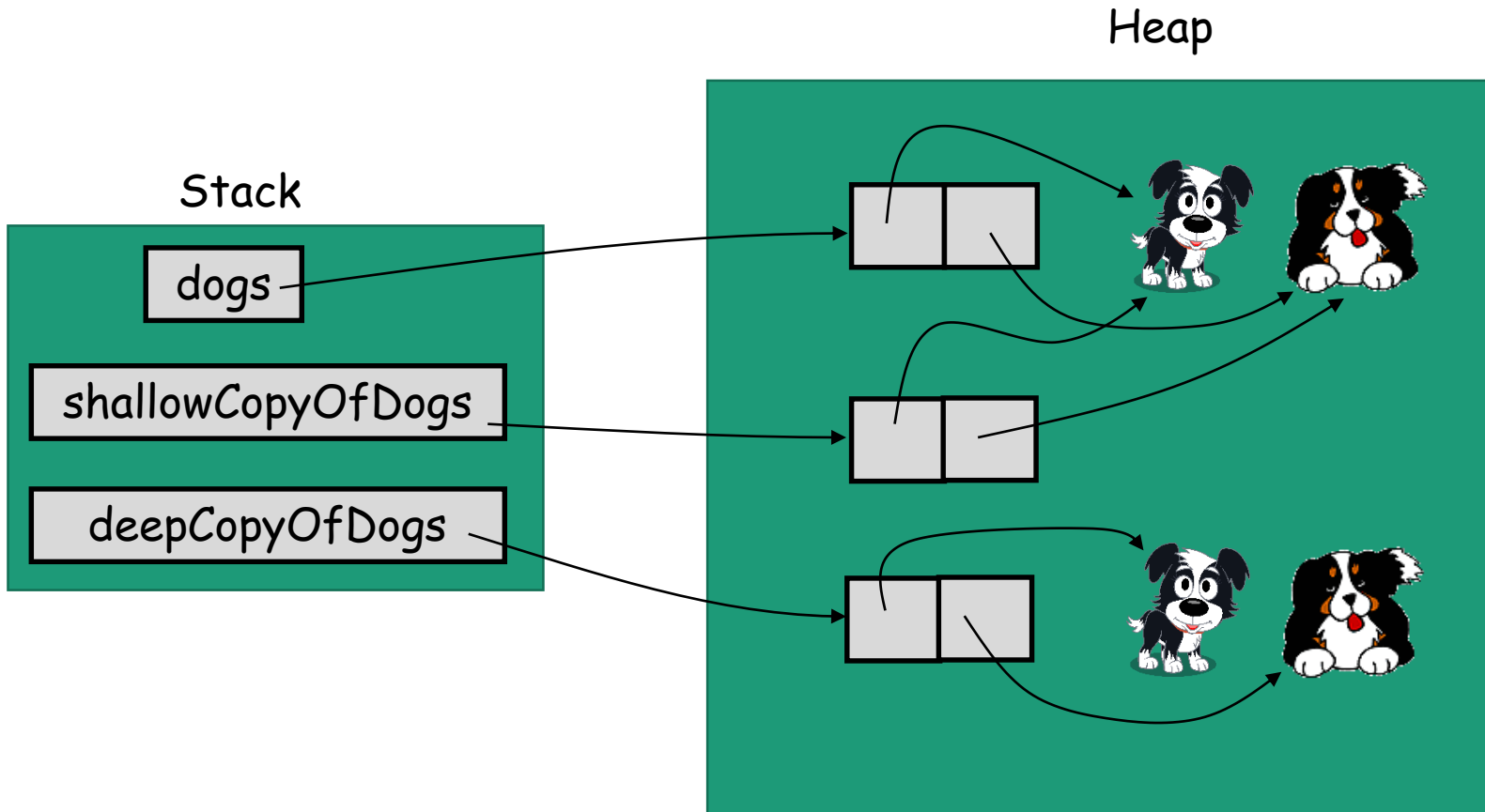
Example: Creating ArrayList

- More examples:
 - `ArrayList<Dog> dogs = new ArrayList<Dog>(100);`
 - `ArrayList<Integer> numbers = new ArrayList<Integer>();`
 - `ArrayList<Double> doubles = new ArrayList<Double>(50);`
- How about the constructor:
 - `ArrayList(Collection<? extends E> c)`
 - Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.

Example: Creating ArrayList, a Shallow Copy via Constructor

- Using the constructor
 - `ArrayList(Collection<? extends E> c)`
 - Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.
 - ArrayList is in fact a Collection
- Example
 - `ArrayList<Dog> dogs = new ArrayList<Dog>();`
 - `dogs.add(new Dog(1, "Buddy"));`
 - `ArrayList<Dog> shallowCopyOfDogs = new ArrayList<Dog>(dogs);`
- What is a "shallow copy", is there a deep copy?

Shallow and Deep Copy



Copy and Clone ArrayList

- Use the Constructor (just discussed) to obtain a shallow copy
- Use ArrayList's clone() method
 - Returns a shallow copy of this ArrayList instance.
 - However, not as nice as the constructor as you have to suppress the "type check" warning,

```
@SuppressWarnings("unchecked")
```

```
ArrayList<Dog> anotherShallowCopyOfDogs = (ArrayList<Dog>) dogs.clone();
```

How about Deep Copy?

- Implement yourself
- Be cautious, and always ask yourself, is the copy of the object a shallow copy or a deep copy?

A Footnote: Creating Object

- Why can you write like this in Java?
 - `dogs.add(new Dog(1, "Buddy"));`
- In comparison, you should not write this in C++
 - `dogs.add(new Dog(1, "Buddy"))`
- even if the add method's signature permits it.

Search ArrayList

- Sequential search on an object
 - Use ArrayList's `indexOf()` method
 - Does not require that the ArrayList is sorted
 - What if one wants to search on a "key"
 - e.g., by a dog's name?
- Binary search
 - Use the Collections class's `binarySearch` method since ArrayList is a collection
 - Require the ArrayList is sorted in the ascending order
 - Discussion in the future due to new concepts yet to be discussed (an example is given)

Sort ArrayList

- Multiple approaches (although essentially equivalent)
 - Use ArrayList's sort() method
 - Use the Collections class's sort() method
 - Discussion in the future due to new concepts yet to be discussed (an example is given)

Array or ArrayList, Which One to Use?

- Do you know how big your "list" should be?

Questions

- Array and ArrayList
 - Characteristics
 - Common operations
- The Arrays class (Array is an array)
- The Collections class (ArrayList is a collection)

Third Party Java Libraries

- Examples:
 - <https://github.com/google/guava>
 - <https://github.com/apache/commons-lang>
 - <https://github.com/qos-ch/slf4j>
 - <https://github.com/spring-projects/spring-framework>

Command Line Arguments

- Want to build an application that take many command line arguments
- Apache Commons CLI
 - <https://github.com/apache/commons-cli>

Example Application

- The "CmdLineArgsDemo" in the Sample Programs

Logging

- Want to build an application that can produce “logs” when it is even being released.
- The Simple Logging Facade for Java (SLF4J)
 - <https://github.com/qos-ch/slf4j>

Example Application

- The "LoggingDemo" in the Sample Programs repository

Assignments

- Required
 - Practice
 - CodeLab
- Optional
 - Revise the BeerSong with the command line arguments to use the Apache Commons CLI library
 - https://github.com/CISC3120SP18/SamplePrograms/tree/master/OOP/01_31/BeerSongClArg

Questions?

- Java API and Libraries
 - Array and ArrayList
- Introduction to Java API and Libraries
 - Arrays, ArrayList, Collections, Random, Math
- Third Party Java Libraries
 - Command line arguments and logging
- Assignments
 - Practice assignment
 - CodeLab