News

How to structure the class?

Tournament? Not many RSVP's

If you have not scheduled a Project01 interview with me, you are late

- Please do so ASAP

Exceptions Review

```
try { }
catch { }

throw new SomeTypeOfException;
void MyFunction(int a, String b) throws SomeException, OtherException { }
```

CS112 –
Java
Programming

Spring 2024

Arrays and other Data Structures

Start to study advanced data structures
What's a data structure? A structure for storing, retrieving, and manipulating data, with desired properties
Data, list, stack, tree, dictionary, etc

Array Review

```
String[] userIds = new String[40];


System.out.println(userIds[41]);
```

Constructed with new
Specify its length at creation
Must construct each element in the list (if not a basic type)
userIds.length is a member variable in the array
ArrayIndexOutOfBoundsException

Alternate Array Syntax

The brackets of the array type can be associated with the element type or with the name of the array

Therefore the following two declarations are equivalent:

```
double[] prices;
double prices[];
```

The first format generally is more readable and should be used

Hmm.

Multidimensional Arrays

```
String[][] userIds = new String[40][50];
for(int n = 0; n < 40; n++) {
      userIds[n] = new String[50];
      for(int p = 0; p < 50; p++) {
            userIds[n][p] = new String("element " + n + "," + p);
      }
}
```

Is that ok?
What is the type of userIds?
What is the type of userIds[17]?
What is the type of userIds[17][3]?

Multidimensional Arrays

```
String[][] userIds = new String[40][];
for(int n = 0; n < 40; n++) {
      userIds[n] = new String[(int) (50*Math.random())];
      for(int p = 0; p < userIds[n].length; p++) {
            userIds[n][p] = new String();
      }
}
```

Is that ok?
YES. SubARRAYS NEED NOT ALL BE THE SAME LENGTH
Must specify at least first index
Must construct (with new) each subcomponent
Why would we use multidimensional arrays?
Video! Width, height, time, color. Audio! Time, left/right channel. Stock price! Time vs NYSE, S&P, NASDAQ, etc.

Reference vs Object - Review

```
void PlayWithArray(int[] data) { // make middle element negative
      data[data.length/2] = -1;
}

…

int[] samples = new int[32000];
for(int n = 0; n < 32000; n++) { samples[n] = n; }
PlayWithArray(samples);
System.out.println(samples[16000]);
```

What does this do? Prints 16k or -1? Try it out!
Does PlayWithArray change the value of samples outside of that function?
Yes!
Not immutable? CORRECT
Strings, Wrapper classes are immutable. We can build classes to be immutable or not. But arrays are not immutable.
Reference to array is passed. Referred-to array object CAN be modified by the reference passed inside to PlayWithArray

Array Productivity Boosters

```
import java.util.Arrays;
```

`Arrays.equals()`: compares two arrays element by element

`Arrays.fill()`: fill an array with a given value

`Arrays.copyOf()`: make a copy of a given array, shortening or lengthening if desired

`Arrays.sort()`

Lots of other methods also. Just FYI in case you want these
Do some in Eclipse!

A new class: ArrayList

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

A **separate class**, richer than built-in array type

Can add, delete elements after construction!

Only can store Objects (including Wrappers), not built-in types

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

This is a new class.
NEW SYNTAX. ArrayList is a "generic" (like a template) not an actual type. Add an Object in brackets to get the actual type
ArrayList<Double>, ArrayList<String>, etc are actual types.

ArrayList

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

int indx = 0;
myList.add(3.14); // end of arraylist
myList.add(indx, 3.14159); // at index position 'indx'
myList.get(indx);
myList.set(indx, 3.14159265);
myList.remove(indx);

Collections.sort(myList); // java.util.Collections;
```

Args are indices into the arraylist!

Plenty more methods: size(), methods to find index of a value, etc.

This is great! Why not use it all the time?

- No built-in types
- Less efficient than built-in arrays
- But very useful and used A LOT

```
//********************************************************************
//  Beatles.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a ArrayList object.
//********************************************************************

import java.util.ArrayList;

public class Beatles
{
   //-----------------------------------------------------------------
   //  Stores and modifies a list of band members.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      ArrayList<String> band = new ArrayList<String>();

      band.add("Paul");
      band.add("Pete");
      band.add("John");
      band.add("George");
```

continue

continue

```
        System.out.println(band);
        int location = band.indexOf("Pete");
        band.remove(location);

        System.out.println(band);
        System.out.println("At index 1: " + band.get(1));
        band.add(2, "Ringo");

        System.out.println("Size of the band: " + band.size());
        int index = 0;
        while (index < band.size())
        {
           System.out.println(band.get(index));
           index++;
        }
    }
}
```

continue

```
      System.out.p
      int locatior
      band.remove

      System.out.p
      System.out.p                                    (1));
      band.add (2,

      System.out.p                                    nd.size());
      int index =
      while (index < band.size())
      {
         System.out.println(band.get(index));
         index++;
      }
   }
}
```

Output

```
[Paul, Pete, John, George]
[Paul, John, George]
At index 1: John
Size of the band: 4
Paul
John
Ringo
George
```

Why talk about ArrayList?

Might be useful to you

Introduces topic of Data Structures

- Design of SW structures that provide useful functions to a programmer
- Storage, retrieval, organization/ordering, analysis, etc of data

I expect you to learn & memorize all methods of ArrayList & their behavior?? NO

We will learn about, design, build, and test our own data structures
Whole courses on this topic

Java Generics

What is a Generic?

```
ArrayList<String> myList = new ArrayList<>();
ArrayList<Card> cardDeck = new ArrayList<>();
```

First stores only `Strings`, second only `Cards`

- We get benefits of strict type checking
- Must be some "Object" type
- An ArrayList<Object> can store any object, but not much we can do with it

Cannot store a built-in type!!!

How do we write our own Generic?

```
class MyOwnClass<TYPE> {
    TYPE var1;
    TYPE var2;
    MyOwnClass(TYPE a, TYPE b) { var1 = a; var2 = b; }
    public TYPE get() { return var1; }
}
```

Nothing magic about word TYPE
Can have more than one TYPE in your class definition: show in Eclipse!

More Data Structures…

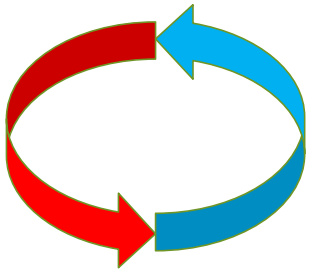
HashMap

A dictionary

- Enter a Key and Value: must be Objects
- Can look up Values by Keys
- Keys must be unique; not Values
- A "generic": must specify the types of the Key and Value to get an actual data type

```
HashMap<String, Integer> idNums = new HashMap<String, Integer>();
idNums.put("Paul", 12345);
idNums.put("Mom", 1);
System.out.println(idNums.get("Paul")); // prints 12345
```

A useful Interface…



Iterables and Iterators

An *iterable* is an object (with type derived from `interface Iterable`) that gives access to a collection of items one at a time

An *iterable* must have an `iterator()` method that makes an *iterator* that actually walks through the items one at a time

An iterator has a `hasNext()` method that returns `true` if there is at least one more item to process

The `next()` method returns the next item

Why do we need *iterable*? So a collection can have >1 *iterator* at a time.

Iterators

Several classes in the Java standard class library are iterators

The `Scanner` class is an iterator

- the `hasNext()` method returns true if there is more data to be scanned
- the `next()` method returns the next scanned token as a string

For-each Loop

The for-each version of the `for` loop can be used when processing any `Iterable` object, including arrays!

```
int[] scores = …;
for (int score : scores) {
   System.out.println(score);
}
```

This is only appropriate when processing all array elements starting at index 0.
It can't be used to set the array values

Iterables: Vector, Set, Collection, ArrayList, arrays, etc

An example! Look at IterableDemo.java

Introduction to Sorting

Bubble Sort

Sorting

Things we sort must be comparable

- 8 < 9 < 15 < 101 < 4196

Sorting in Python was easy

```
values = [44, 13, -105, 71, 8]
values.sort()
print(values)
```

"Comparable" is an "Interface" in Java (look up online!)
Sorting puts them into order!

Sorting in Java

There are built-in `sort()` methods in Java also

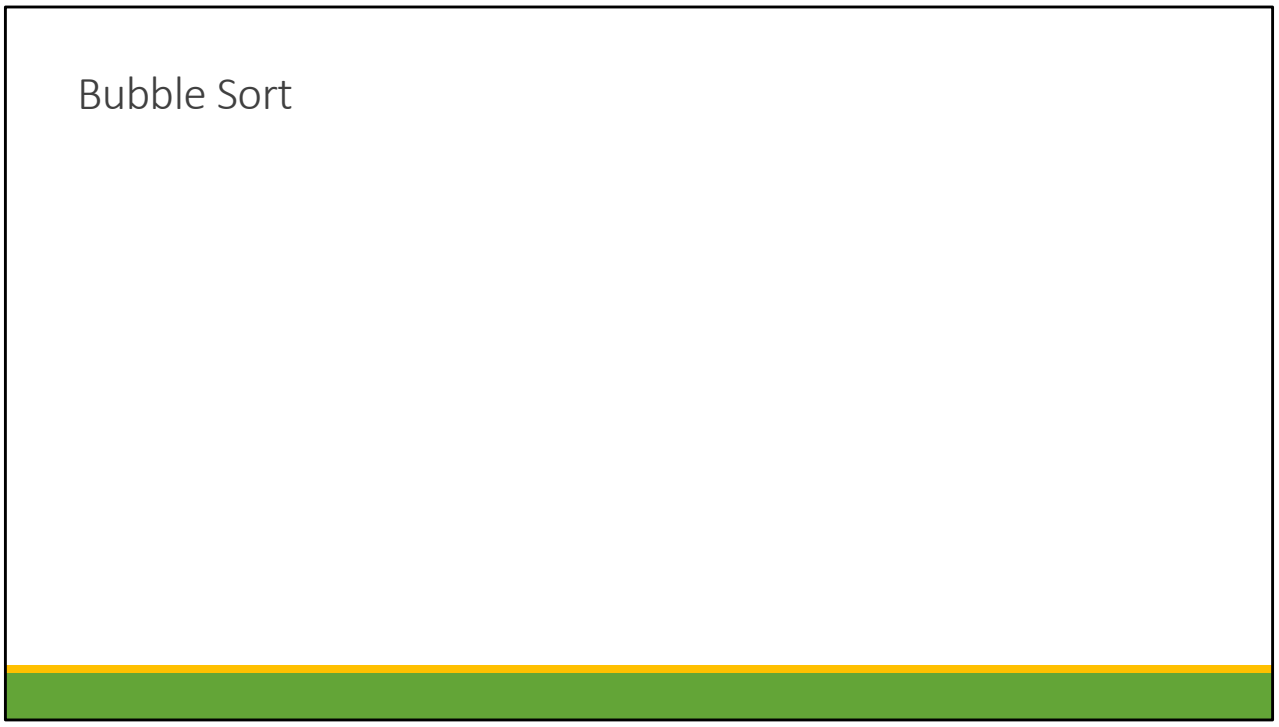
- Arrays.sort()
- Collections.sort()

But how do these functions actually work?

LOTS OF DIFFERENT WAYS TO DO SORTING

- Simple and complicated, fast and slow, lots of memory or a little
- We will study more methods in a few weeks, when we learn about recursion
- My real-world sort problem: bubble sort too slow with 1 MHz CPU

Bubble Sort



5, 1, 4, 1, 2, 8
Show technique on board visually
Then write code
Why called Bubble Sort? Largest values "bubble toward the top"

Bubble Sort

```
int length = 100;
int[] data = new int[length];
SetRandomData(data);

for(int endPoint = length-1; endPoint >= 1; --endPoint) {
     for(int j = 0; j < endPoint; j++) { // each little step
          if (data[j] > data[j+1]) { // do we swap?
               Swap(data, j, j+1);
          }
     }
}
```