Compsci 201
Work, Nbody, ArrayLists

```
jshell> String[] a = {"ant", "bat", "cat", "dog"}
a ==> String[4] { "ant", "bat", "cat", "dog" }

jshell> System.out.println(a)
[Ljava.lang.String;@5b275dab

jshell> System.out.println(Arrays.toString(a))
[ant, bat, cat, dog]
```

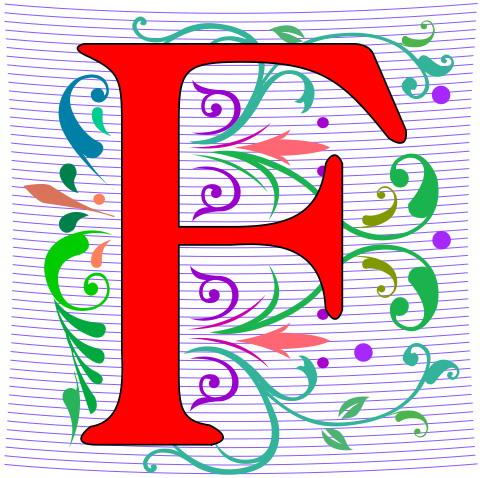
Susan Rodger

January 29, 2020

`F` is for …

- **Folder**
  - aka Directory – where things are stored in Git

- **Function**
  - Abstraction – a method in Java



PFTW

- Getting things done in 201
  - How to succeed and enjoy the effort
- Mundane Java-isms
  - From char to autoboxing: primitives
  - What is **`this`**?
- Generic classes: How ArrayList works
  - Design, create, test, measure

Getting Things Done in 201

- What do these data mean for you, for me, for the community of 286 students in Compsci 201?

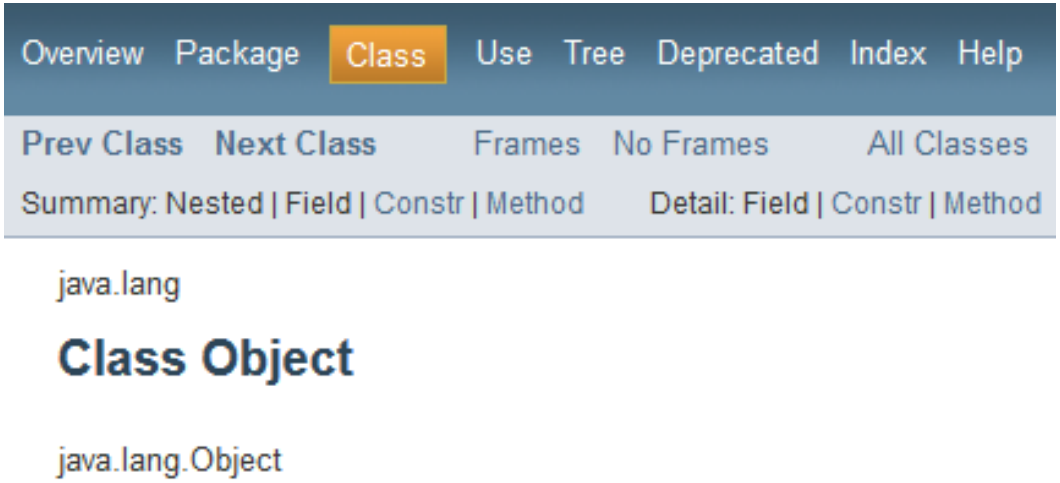
1							
2		How long did you take to complete this APT					
3	**Choose the APT**	**< 1 hour**	**1-2 hours**	**2-4 hours**	**4-6 hours**	**6-10 hours**	**> 10 hours**	**Total**
4	AccessLevel	241	20	5				266
5	CirclesCountry	160	81	20	4	1		266
6	Common	94	38	14	4			150
7	DNAMaxNucleotide	112	97	36	14	3	3	265
8	Gravity	254	4	2		1		261
9	SandwichBar	109	100	36	13	1		259
10	Totality	236	27	4				267
11	**Grand Total**	1206	367	117	35	6		

From Last Time …
Go over
WOTO: Correctness Counts

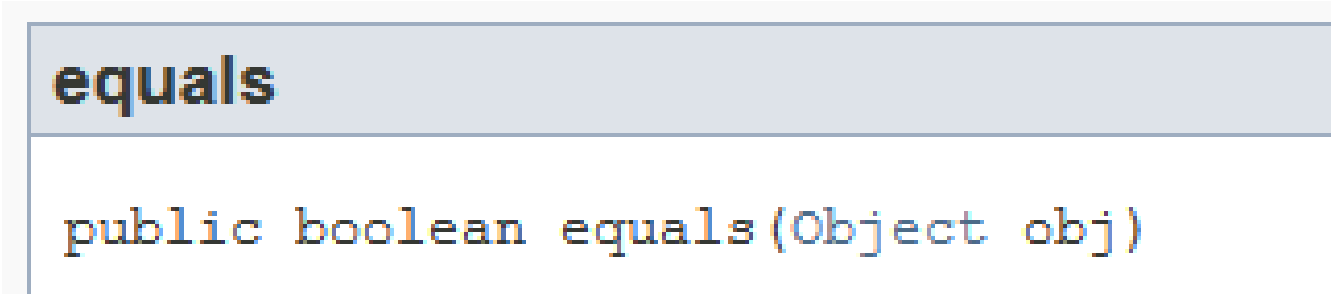
http://bit.ly/201spring20-0124-2

Object class, equals method

- In JavaDoc



- Signature of equals method



@Override .equals

- Create a new Point method
  - Use annotation @Override, help with errors

```
boolean equals(Object o) { …
```

- Must use this signature, to implement:
  - Cast parameter appropriately
  - Compare instance fields

Point inherits Object.equals

- **This doesn't work for Point objects!**
  - *Default* simply uses ==, no idea about points
  - a.equals(b) if a and b *reference the same object*
  - Two different (0,0) points not the same

Point equals fixed!

```
@Override
public boolean equals(Object o) {

    Point other = (Point) o;
    if (other.myX == myX && other.myY == myY) {
        return true;
    }
    return false;
}
```

Contract for Equality

- Reflexive **`x.equals(y)`** then **`y.equals(x)`**
- Transitivity: **`x.eq(y), y.eq(z)`** then **`x.eq(z)`**

- Check **`x.equals(x)`** as a special case with ==
- Check **`this.getClass() == o.getClass()`**
  - Don't want to have an apple == orange

- Cast Object parameter and use instance variables
  - See Point as example

Amanda Randles, Duke 2005

- ACM Grace Murray Hopper Award (<= 35 yo)

For developing HARVEY, a massively parallel circulatory simulation code capable of modeling the full human arterial system at subcellular resolution and fostering discoveries that will serve as a basis for improving the diagnosis, prevention, and treatment of human diseases.

//XXX and Amanda Peters
//Compsci 100: Huffman Coding
//November 19, 2002

I felt like working in a pair was a really successful way to complete the program. It helped the most when it came to working out basic logic and finding errors. I found it really helpful because he often would see the basic logic to the code and I could help more with the implementation. I feel like it was a successful group and we both contributed a lot.

Reading Points

- We'll typically use a Scanner to read values
  - Use **`.hasNext()`**, **`.hasNextDouble()`**, …
  - If/while there's more to read? Call **`.next()`**

- Method **`.next()`** returns a String
  - Method **`.nextDouble()`** returns a double …

- See PointReader.java class, useful in NBody

Scanner Sources for Reading

- Construct a Scanner from System.in
  - Reads from keyboard/console
  - **`.hasNextX()`** true until end-of-file OR no X
    - Control-D on OS X, Control-Z on Windows
- Construct a Scanner from a File
  - Reads from file, *exception could happen*
  - **`.hasNextX()`** true until all of file read OR no X
    - Each call of **`.nextX()`** returns the next X, internally the Scanner "remembers" where it last read

Scanner hasNext and next

- Think about scanner as a long reel/source of data
  - If **`.hasNext()`** returns true, there is something to read by Scanner cursor/reader
  - Calling **`.next()`** returns *and advances* cursor
  - Scanner object maintains cursor internally
- Source: file, String, terminal, ..

N-Body Simulation

- Class CelestialBody represents Celestial Body
  - Planet, Sun, Asteroid
  - Models an object in 2D space, not 3D
  - Position, Velocity, Mass, Image for display

- Class NBody drives the simulation
  - Compute gravitational forces: physics
  - Time-step simulation
    - compute all forces, update ,display

Class CelestialBody

- Illustrates standard Java idioms
  - Constructors, Methods, Instance Variables
- State is private: six instance variables
  - **`myXPos`**, … using *my* convention - this object
  - Initialized by constructors
- Methods are public
  - Include *accessors* aka *getters* for state
  - No *setters*, cannot change **`myXPos`** other than via the update method, a *mutator*

The Object Concept

- Every instance variable and every non-static method accessed/called after Object.Dot
  - **`b.getX(), b.calcForcExertedBy(other)`**

- From within a class, e.g., CelestialBody
  - **`myXPos`**, **`getX(),this.myXPos`**,
  - All are equivalent as is **`this.getX()`**

- Some prefer always using this. – clearer?

NBody numbers

- Floating point issues, problems, quandaries
  - When is (a + b) + c != a + (b + c)
  - When is a/b * c != a*c / b
    - Watch for this in Gradescope tests!!

Debugging Arithmetic

- Order of operations with floating point values can result in overflow, underflow, more
  - Small number + Big number …

```
jshell> (5 + 1e20) + -1e20
$33 ==> 0.0

jshell> 5 + (1e20 + -1e20)
$34 ==> 5.0
```

Debugging double Arithmetic

- Integer values are not the same as Double values
  - 1/0 is … whereas 1.0/0 is …

```
jshell> Double.MAX_VALUE
$48 ==> 1.7976931348623157E308

jshell> Double.MAX_VALUE + 2
$49 ==> 1.7976931348623157E308

jshell> Double.MAX_VALUE * Double.MAX_VALUE
$50 ==> Infinity

jshell> 1.0/0
$51 ==> Infinity

jshell> Math.sqrt(Double.POSITIVE_INFINITY)
$52 ==> Infinity

jshell> Double.POSITIVE_INFINITY + Double.NEGATIVE_INFINITY
$53 ==> NaN

jshell> 1/0
|  Exception java.lang.ArithmeticException: / by zero
|        at (#54:1)
```

Completing NBody

- Please read the TL;DR document
  - Test at each step, push constantly using Git
- After using supplied Test… classes, proceed to simulation
  - Must be able to read data file to simulate
  - Understand the basics, read carefully
- Analysis: complete before submitting to Gradescope for final submission

Now look at DNAMaxNucleotide

- Return the strand from strands array with most occurrences of nucleotide nuc. Return longest such strand

```
public class DNAMaxNucleotide {
   public String max(String[] strands, String nuc) {
         // fill in code here
   }
}
```

- Example

2.

```
strands = {"agt", "aagt", "taattt", "ccatc" }
nuc = "g"
```

Returns: "aagt" since both "aagt" and "agt" have one occurrence of 'g', but "aagt" is longer.

Algorithm - DNAMaxNucleotide

- Does this code make the algorithm clear?
  - Why must ***count*** be a helper method?
  - Why can't max = 0 before loop?

```
2     public String max(String[] strands, String nuc) {
3         String ret = "";
4         int max = 1;
5         for(String s : strands) {
6            int nc = count(s,nuc);
7            if (nc > max || (nc == max && s.length() > ret.length())) {
8                ret = s;
9                max = nc;
10           }
11        }
12        return ret;
13    }
```

Two Versions of Helper Method

- Iterating over each character of a string
  - Note that **`nuc`** is a one-character string
  - How does **`s.substring(a,b)`** work?

```
15      private int count(String s, String nuc) {
16          int total = 0;
17          for(int k=0; k < s.length(); k+= 1) {
18              String one = s.substring(k,k+1);
19              if (one.equals(nuc)) {
20                  total += 1;
21              }
22          }
23          return total;
24      }
```

Critique of another implementation

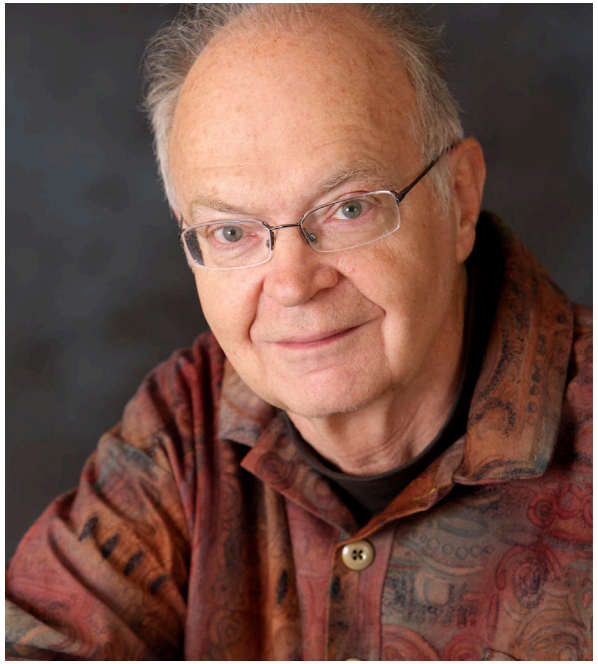
- **Where does this solution come from?**
  - Strings are immutable, s.replace(…)
    - Replace every "a" with "" (nothing)
  - Is this better? Different? More clever? More of a hack? …

```
15      private int count(String s, String nuc) {
16          int tot = s.length() - s.replace(nuc, replacement: "").length();
17          return tot;
18      }
```

WOTO

http://bit.ly/201spring20-0129-1

Donald Knuth



- aka “The Donald”
- Turing award (and others)
- Author of “The Art of Computer Programming”
  - Arguably most important book written in Computer Science
  - First publication: Mad Magazine

If you optimize everything you will always be unhappy.

Everyday life is like programming, I guess. If you love something you can put beauty into it.

https://www.youtube.com/watch?v=cK7yyjXfbc4

From Array to ArrayList

- Have **`int[], String[], CelestialBody[]`**
  - Array of any type, *but doesn't grow*
  - Can't use **`.contains`** with array, can't print
- The java.utils.Arrays class has some help

```
jshell> String[] a = {"ant", "bat", "cat", "dog"}
a ==> String[4] { "ant", "bat", "cat", "dog" }

jshell> System.out.println(a)
[Ljava.lang.String;@5b275dab

jshell> System.out.println(Arrays.toString(a))
[ant, bat, cat, dog]
```

java.util.ArrayList

- Growable array with many useful methods
- https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/List.html
  - Can only contain Object types (no primitives)

- Convert from array?
  - **`Arrays.asList`**
    - It's a List!
  - String yes, int no

```
jshell> ArrayList<String> b = new ArrayList<>()
b ==> []

jshell> b.add("ant")
$5 ==> true

jshell> b.add("bat")
$6 ==> true

jshell> b.add("cat")
$7 ==> true

jshell> b.size()
$8 ==> 3

jshell> System.out.println(b)
[ant, bat, cat]

jshell> b.indexOf("cat")
$10 ==> 2

jshell> b.indexOf("dog")
$11 ==> -1
```

From Array to ArrayList

- Can make conversion with Object, e.g., String
  - Use Arrays.asList as a bridge, be careful

```
jshell> String[] a = {"cat", "dog"}
a ==> String[2] { "cat", "dog" }

jshell> ArrayList<String> b = new ArrayList<>(Arrays.asList(a))
b ==> [cat, dog]

jshell> b.add("fox")
$23 ==> true

jshell> b
b ==> [cat, dog, fox]
```

Primitive Array? do it yourself

- **No bridge from Arrays.asList since primitive**
  - Loop and use autoboxing/unboxing
  - Conversion of int to Integer and *vice versa*

```
jshell> int[] a = {1,2,3,4,5}
a ==> int[5] { 1, 2, 3, 4, 5 }

jshell> ArrayList<Integer> b = new ArrayList<>()
b ==> []

jshell> for(int val : a) b.add(val)

jshell> b
b ==> [1, 2, 3, 4, 5]

jshell> b.add(55)
$29 ==> true

jshell> b
b ==> [1, 2, 3, 4, 5, 55]
```

Objects, Primitives, Arrays/Lists

- array can hold any type: **`int[], String[]`**
- **`ArrayList`** only Object types, not primitives
  - Autoboxing allows for add/get int ::: Integer
- **`ArrayList<Object>`** a, **`a.toArray(…)`** array
  - Syntax is not intuitive, see examples in code
- **`Arrays.asList(Object[])`** to **`ArrayList`**
  - Actually returns **`List`**, not **`ArrayList`**, …

-- real world APT?

- https://leetcode.com/problems/unique-morse-code-words/
  - `"a" > ".-", "b" > "-..."`
  - `"z" > "--.."`
  - Note "gin" > "--...-." and "zen" > "--...-."
- **Given an array of strings, how many unique encodings are there?**
  - Also given String[] of 26 Morse codes, where code[0] = ".-" for "a"

High Level Ideas

- First step: what algorithm/method will you use?
  - Verify that it's correct. High level isn't easy

- Is it necessary to look at/process every string?
- What value is returned, how to determine value?

- Stop, think, don't code, …

LeetCode Solution

- What's a high level solution using known tools?
  - What is the method **`makeMorse()`** ?
  - Talk to your interviewer … it's a dialog

```
4       public int uniqueMorse(String[] words) {
5           HashSet<String> set = new HashSet<>();
6           for(String s : words) {
7               set.add(makeMorse(s));
8           }
9           return set.size();
10      }
```

From Nothing to Done

- Basic ideas: how do we access encodings in an array where code[1] is for 'b', "-…"
  - Arithmetic with char values, 'b' – 'a' == 1
  - What about (int) 'b' == 97?
    - https://youtu.be/xLpfbcXTeo8?t=49

- Loop over characters in a String?
  - Index **`k`** with **`s.charAt(k)`**
  - Or **`for(char ch : s.toCharArray()){`**

WOTO with Live/Leet Code

- **Ideas for solving LeetCode problem**
  - Given array of Strings, return number of unique Morse code encodings
  - How is a set useful here? Doable without?

From DNAMax to Morse Code

- loop on 18-23, why does ch-'a' serve as index?
  - Primitive char is an int except when printed

```
12      private String makeMorse(String s) {
13          String[] m = {".-","-...","-.-.","-..",".",".-.",
14                        "--.","....","..",".---","-.-",".-..",
15                        "--","-.","---",".--.","--.-",".-.",
16                        "...","-","..-","...-",".--","-..-","-.--","--.."};
17
18          String ret = "";
19          for(char ch : s.toCharArray()) {
20              int dex = ch-'a';
21              ret += m[dex];
22          }
23          return ret;
24
25      }
```

ArrayList<…>

- Generic aka parameterized type
  - Any Object subtype can be in ArrayList<..>

- Integer, Double, Char, Boolean are wrapper classes for primitives
  - Mostly these work. But immutable. Cannot increment an Integer, can create new one

WOTO

http://bit.ly/201spring20-0129-2