Java
Object-Oriented Programming

https://201wo.to

https://canvas.duke.edu/courses/70546

Owen Astrachan and **Alex Steiger**

ola@duke.edu, asteiger@cs.duke.edu

- ***Boolean***
  - True, False: 1 bit to represent 2 values?

- ***Bit***
  - 0, 1, **B**inary Dig**IT**

Administrative

- All material is online, https://201wo.to
  - Click “Section 02” or visit https://duke.is/cs201-02
- This week: Discussion on Friday

- Next week: APTs due Thursday
- Following week: P0 and APTs
- Reminder: No class on MLK, January 19

Syllabus Reminders

- APTs due Thursday, accepted through Friday
  - No credit for late APTs, try to do some each week
  - There will be many "extra/optional" APTs

- Projects typically due Tuesday
  - Typically, early engagement points
  - 48 hours no penalty, then 10% per day
  - Start early, ask questions

Completing WOTOs

- Designed to help ensure understanding
- Designed to create a break during lecture
- Designed to reward attending in person, but
  - Engagement points for "on time" completion
  - Engagement point for completion next morning

- Related to engagement exercises/questions?
  - Yes! outside-of-class engagement

Plan for the Day

- Learn enough Java to complete APTs
  - Strategy and workflow for completing APTs
- Set the stage for enough Java for P0
  - Project will go live this week
- Hope git, ssh, and VSCode work simply
  - Why for some and not everyone?

Java overview

- Object-oriented, strongly typed language
  - Compiled from .java source code into byte code
  - Byte code in .class file executed by JVM, Java virtual machine
  - VS Code does this for you -- you downloaded the JVM and a Java runtime environment

- Object-oriented: based on classes (and more)

Java is statically and strongly typed

- Variables have type identified in source code
  - Strong and static typing, compile time errors
  - Python is dynamically typed, runtime errors

- Variables have a name, a type, and a value
  - The value can change, name and type do not

What is a class?

- Class is a programming construct
  - Encapsulates ***state*** and ***behavior***

- object: instance of a class
  - Class is a template, blueprint, object factory

- Class methods operate on objects
  - **`str.length(), scn.next()`**

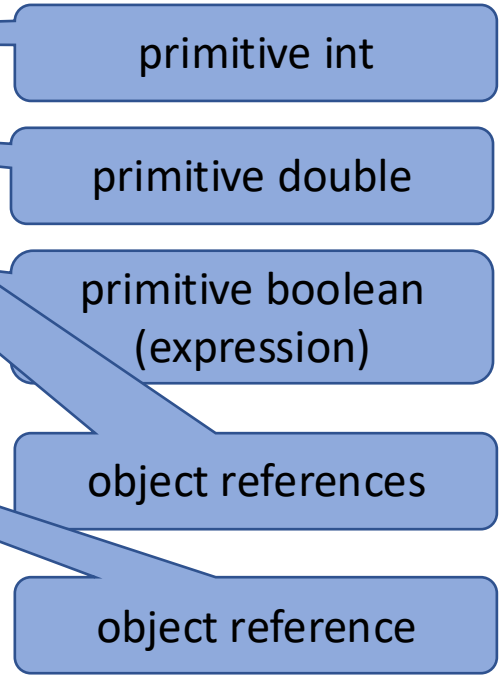
Analogies: helpful and too much detail?

Prompt: As an analogy for the concept of object-oriented programming in which an object is an instance of a class, so that the class is (a) a blueprint, (b) a factory for creating objects, do these make sense? Is there something else?

- Claude: https://bit.ly/oop-analogy-claude

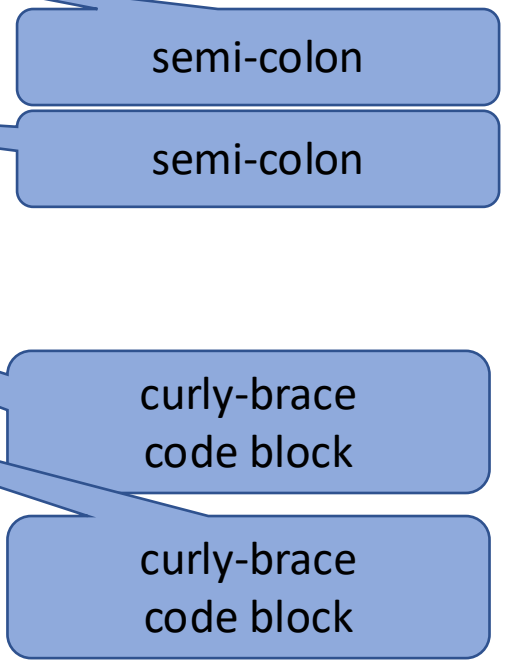
Revisit word counting program

```
6   Scanner s = new Scanner(new File("data/twain5.txt"));
7   ArrayList<String> list = new ArrayList<>();
8   int wcount = 0;
9   double start = System.nanoTime();
10  while (s.hasNext()) {
11      String word = s.next().toLowerCase();
12      wcount += 1;
13      if (! list.contains(word)) {
14          list.add(word);
15      }
16  }
17  double end = System.nanoTime();
18  double time = (end-start)/1e9;
19  System.out.printf("unique #: %d, total #: %d\n",list.
20  System.out.printf("time: %2.3g\n", time);
21  s.close();
```



Statements and blocks: ; and { and }

```
6   Scanner s = new Scanner(new File("data/twain5.txt"));
7   ArrayList<String> list = new ArrayList<>();
8   int wcount = 0;
9   double start = System.nanoTime();
10  while (s.hasNext()) {
11      String word = s.next().toLowerCase();
12      wcount += 1;
13      if (! list.contains(word)) {
14          list.add(word);
15      }
16  }
17  double end = System.nanoTime();
18  double time = (end-start)/1e9;
19  System.out.printf("unique #: %d, total #: %d\n",list.
20  System.out.printf("time: %2.3g\n", time);
21  s.close();
```



Java Primitives and Objects

- Java two different kinds of values/variables
  - Primitive: int, double, char, boolean, long, …
  - Objects: pointers/references to object/memory

- Primitive types: ranges

```
jshell> Integer.MIN_VALUE
$1 ==> -2147483648

jshell> Integer.MAX_VALUE
$2 ==> 2147483647
```

```
jshell> Double.MIN_VALUE
$3 ==> 4.9E-324

jshell> Double.MAX_VALUE
$4 ==> 1.7976931348623157E308
```

```
jshell> Integer.MAX_VALUE + 1
$1 ==> -2147483648
```

Primitive types: variable holds a ***value***

- **int**, **long**: common integer types
- **double**: common decimal number type
- **boolean**: true or false
- **char**: 'z', '!'; number that prints as a character
  - Note single 'quotes'

- Also byte, short, float, but we won't use these

Object types: variable holds a ***reference***

- String, Scanner, ArrayList (array), URL
  - Variable stores ***reference*** or pointer to an object
  - The value is like an address in memory

- Typically call ***new*** to create an object, store reference to access the object
  - Strings are special
  - See dot notation in file-reading program

Java expressions and operators

- Arithmetic operators: int, double
  - Watch for mixing types: 3/2 is an int
  - Watch for overflow: x = 1000000000, x*3 < 0
- Logical operators for boolean expressions
  - We will see short-circuit evaluation
- Relational operators: <, >, and …

Some primitive Java operators

`+, -`	Add, subtract
`*, /, %`	Multiply, divide, modulus: 5/4 = 1, 5.0/4 = 1.25
`<, <=`	Less than, less than or equal to
`>, >=`	Greater than, greater than or equal to
`==`	Equal (only for primitive types!!!)
`!`	Logical NOT (!a means a must not be true, aka false)
`&&`	Logical AND (a && b means a and b both true)
`||`	Logical OR (a || b means one (both) of a and b true)

Coding Interlude

- Looking at limits of primitives using jshell

Primitive type double in Java

- More accurate than float type
  - 64 bits rather than 32 bits (int, float)
  - 3.14159, Math.sqrt(25), 1.7E300

- Mixed type conversion?
  - 3 + 2.2 is a double, 5.2
  - 1.0/2 is a double, 0.5
  - 5.0 + 2/3 is a double, 5.0

Learning new things

- Storage for LLMs matters more than in 201
  - 64 bit double, 32 bit float
    - 16bit FP16/Float16 in C and others

- NVIDIA changes the landscape?
  - https://bit.ly/nvidia-float talks about 8-bit float!
  - One H100 Tensor Core GPU? > $31,000
    - Intel or AMD high-end cores: $300?

Primitive type boolean in Java

- Values: true, false, ***short-circuit evaluation***

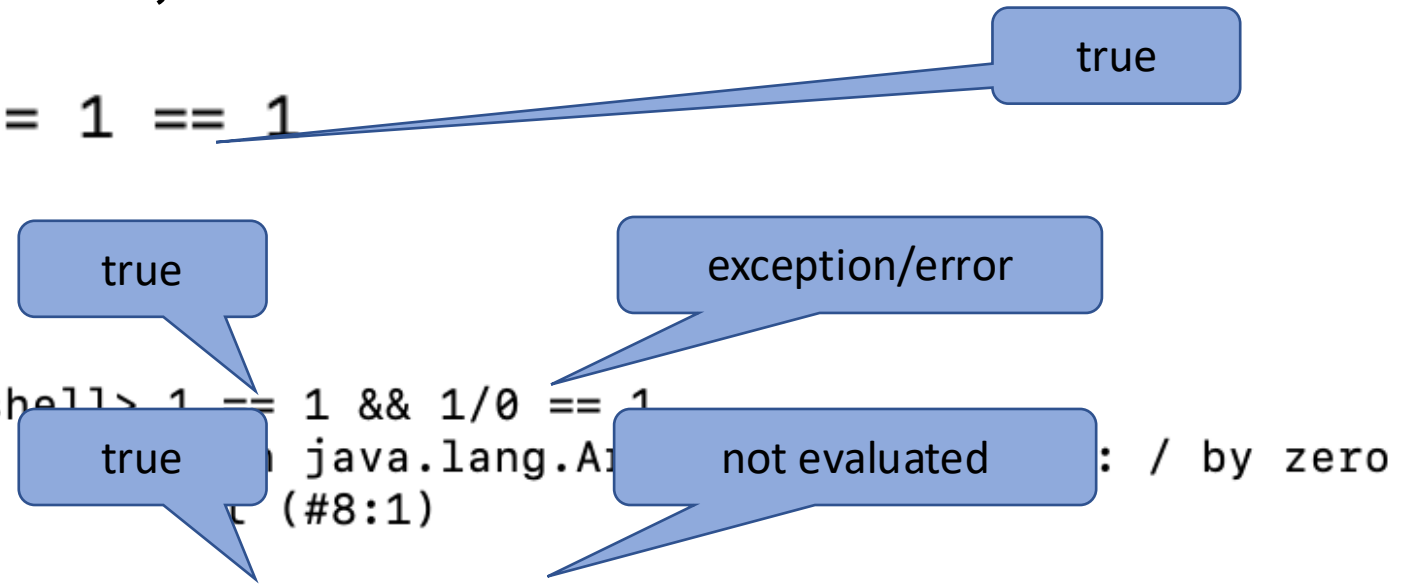
```
jshell> boolean x = 1 == 1
x ==> true

jshell> x && !x
$6 ==> false

jshell> x || !x
$7 ==> true
```

```
jshell> 1 == 1 && 1/0 == 1
|  java.lang.Ar                : / by zero
|        at (#8:1)

jshell> 1 == 1 || 1/0 == 1
$9 ==> true
```



From primitives to objects/references

- All variables have: ***name, type, value***
  - int x = 42; double y = 1.25;

- All expressions have: ***type, value***

```
• 5 < 7,
• true,
```

```
20/3,
   6,
```

```
57 % 2,
     1,
```

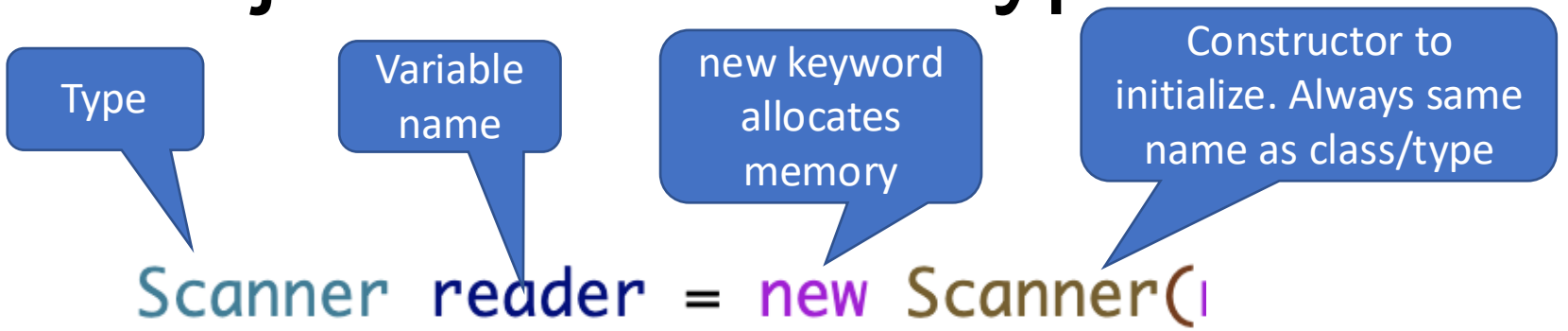
```
1.0/2,
  0.5,
```

```
1 == 1
true
```

Java Object/Reference/Class Types

- Standard object types we use in 201
  - String, Scanner, ArrayList, Set, URL, File, …
- User-defined object types: Person201
  - All Java code is in a class, file: Person201.java
- Object variables: references or pointers
  - Different from primitive variables

Java object/reference types



- Variable stores a **reference** to an **object**, i.e., a place in memory.
- Can access method calls with the **dot operator**.

```
while (reader.hasNext()) {
    String word = reader.next();
```

The Java class String

- Object variable: reference, pointer (label)?
  - Differences: `==` and `.equals`
  - ***Not like primitive variables***


```
String s = new String("Duke")
String t = new String("Duke");
// only one if statement is true!!!
if (s == t) {they label the same box}
if (s.equals(t)) {contents the same}
```

false

true

The Java class String

- Object variables: reference, pointer (label)
  - new returns pointer/label
  - = copies pointer, "gets"


```
String s = new String("Duke")
String t = s;
// both statements are true!!!
if (s == t) {they label the same box}
if (s.equals(t)) {contents the same}
```

	true
	true

Strings are object references, but …

- Do not need to call new, you can, but …

```
jshell> String s = "hello"
s ==> "hello"

jshell> String t = new String("hello")
t ==> "hello"

jshell> s == t
$12 ==> false
```

```
jshell> String w = "hello"
w ==> "hello"

jshell> s == w
$14 ==> true
```

- Strings can be "interned", may differ across OSs

String concepts, String methods

- Can use new, but also assign literal: **`"hello"`**
- **`.length()`** method for # characters in string
- **`.charAt(2)`** method accessing k^{th} character
  - Returns primitive char value
- **`.substring(j,k)`**, index **`j`** up to index **`k-1`**
  - Note **`s.substring(j,k).length() == k-j`**

- **`s + t`** overloaded as **`s.concat(t)`**

WOTO: L02-A on PrairieLearn

https://pl.cs.duke.edu

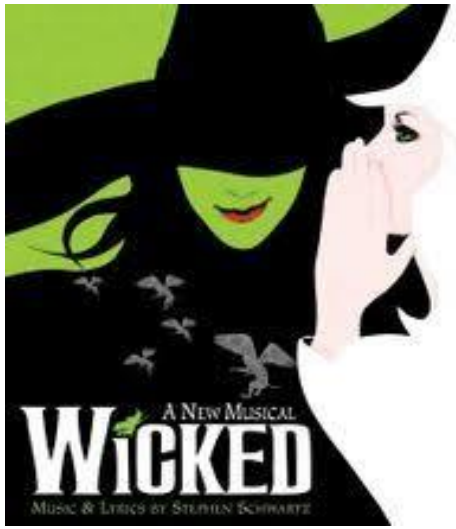
APT: Algorithmic Problem-Solving & Testing

- First used in 201 in Spring 2003 with C++
- More than a million submissions since fall19

- Develop an algorithm and code to solve a problem using Java and standard APIs
- Typically viewed as a favorite part of 201 (101)

Defying Gravity: WOTO-Style

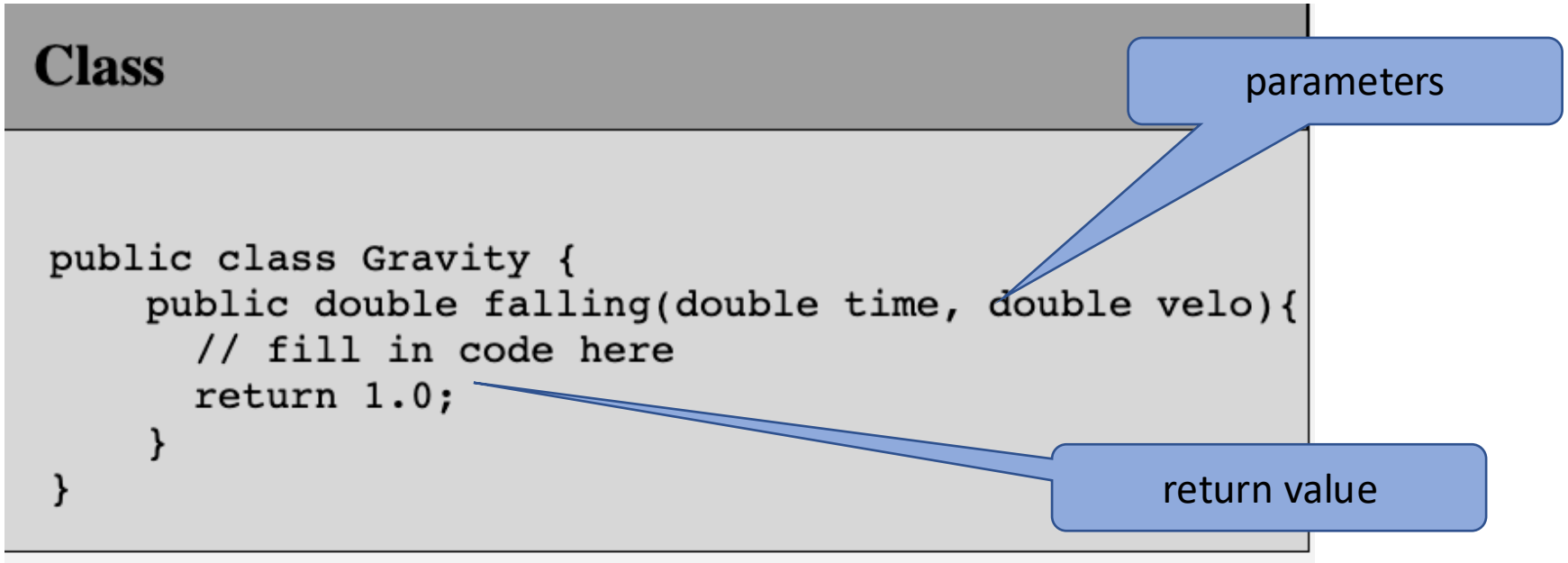
- How to do an APT
  - https://www2.cs.duke.edu/csed/newapt/gravity.html
  - From reading to coding with VSCode



https://youtu.be/Yf9Bt5WFZKs?t=111

Anatomy of falling due to Gravity

- Class `Gravity` has one method, **`falling`**
  - The method **`falling`** has two parameters
  - The method **`falling`** returns a double value



Think Before You Code

- Solve by hand … Check your understanding of examples … think about solution/code
  - Thing before typing: $d = v_0*t + 0.5*a*t^2$

Learning a new language

- Translate ***known***, e.g., Python, into the ***new***: Java
  - You'll view Java, the new language, in terms of the language you know
  - Eventually program idiomatically and colloquially!

- Arithmetic with integers and doubles as an initial and common vernacular?

Coding Interlude

- Working on Gravity APT in VSCode

Example.java is a file for class Example

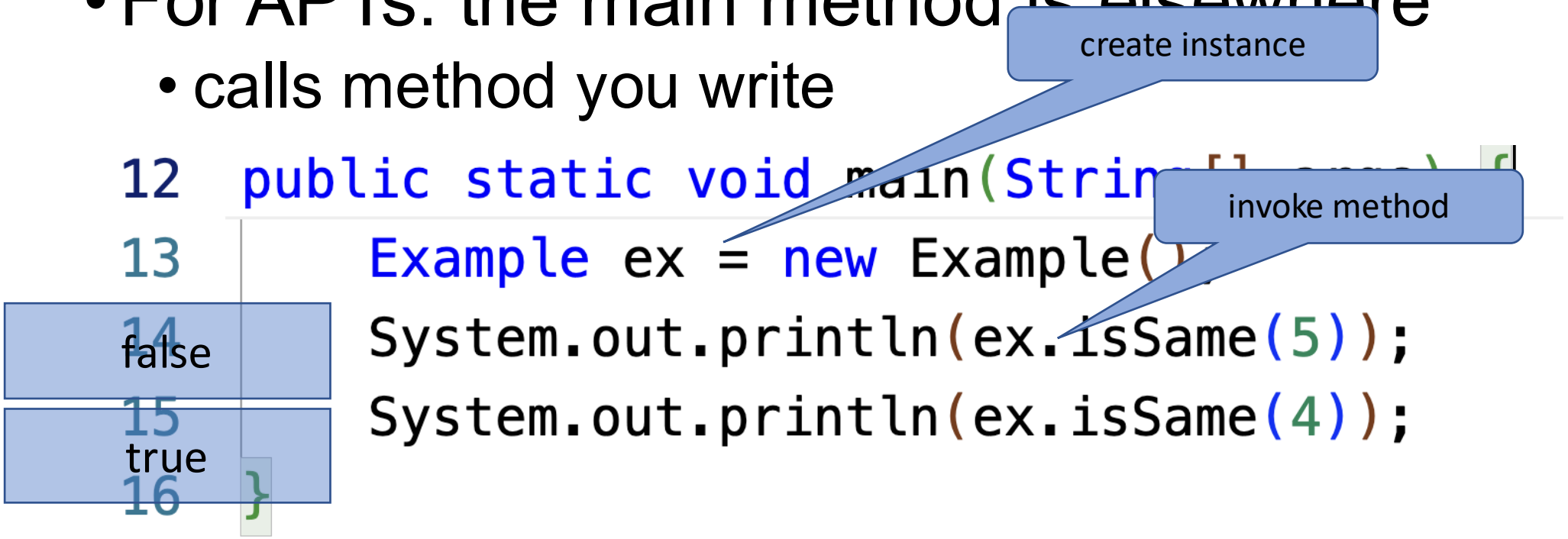
```
1  public class Example {
2
3      public int increase(int value){
4          return value * 2;
5      }
6      public int decrease(int value){
7          return value/2;
8      }
9      public boolean isSame(int value){
10         return value == increase(decrease(value));
11     }
```

class name/file name

method: return/parameters

Create instance, call methods

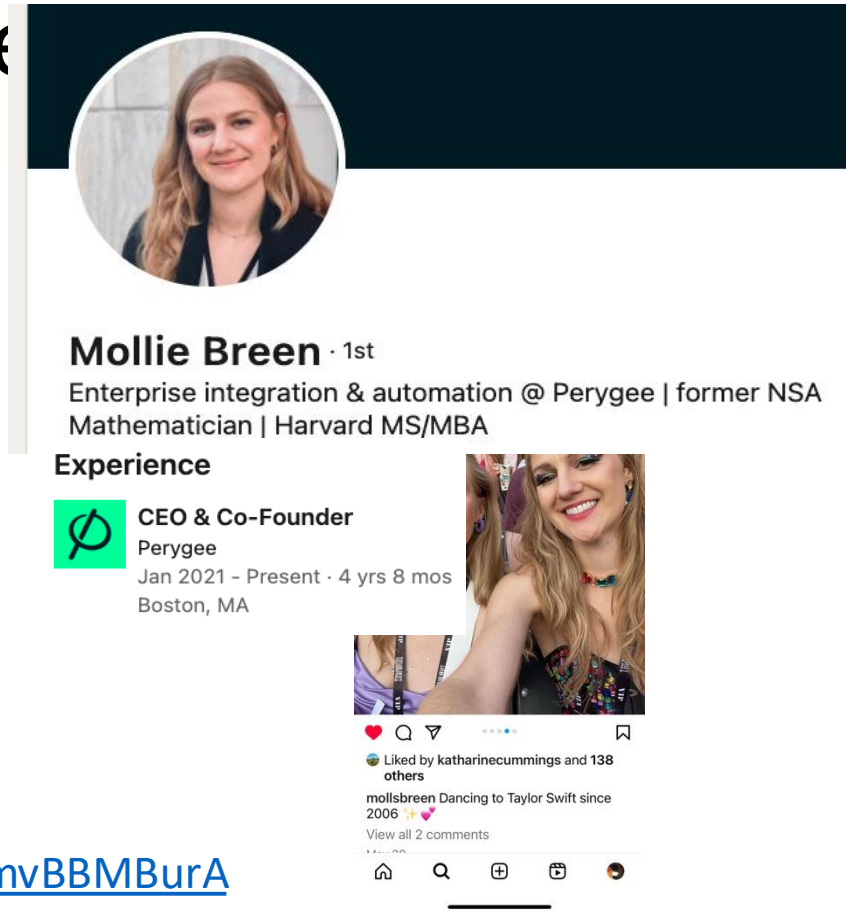
- For APTs: the main method is elsewhere
  - calls method you write



Completing the Gravity APT

- Is getting all-green a requireme
  - Can you do well without getting
  - http://thegreendance.com/

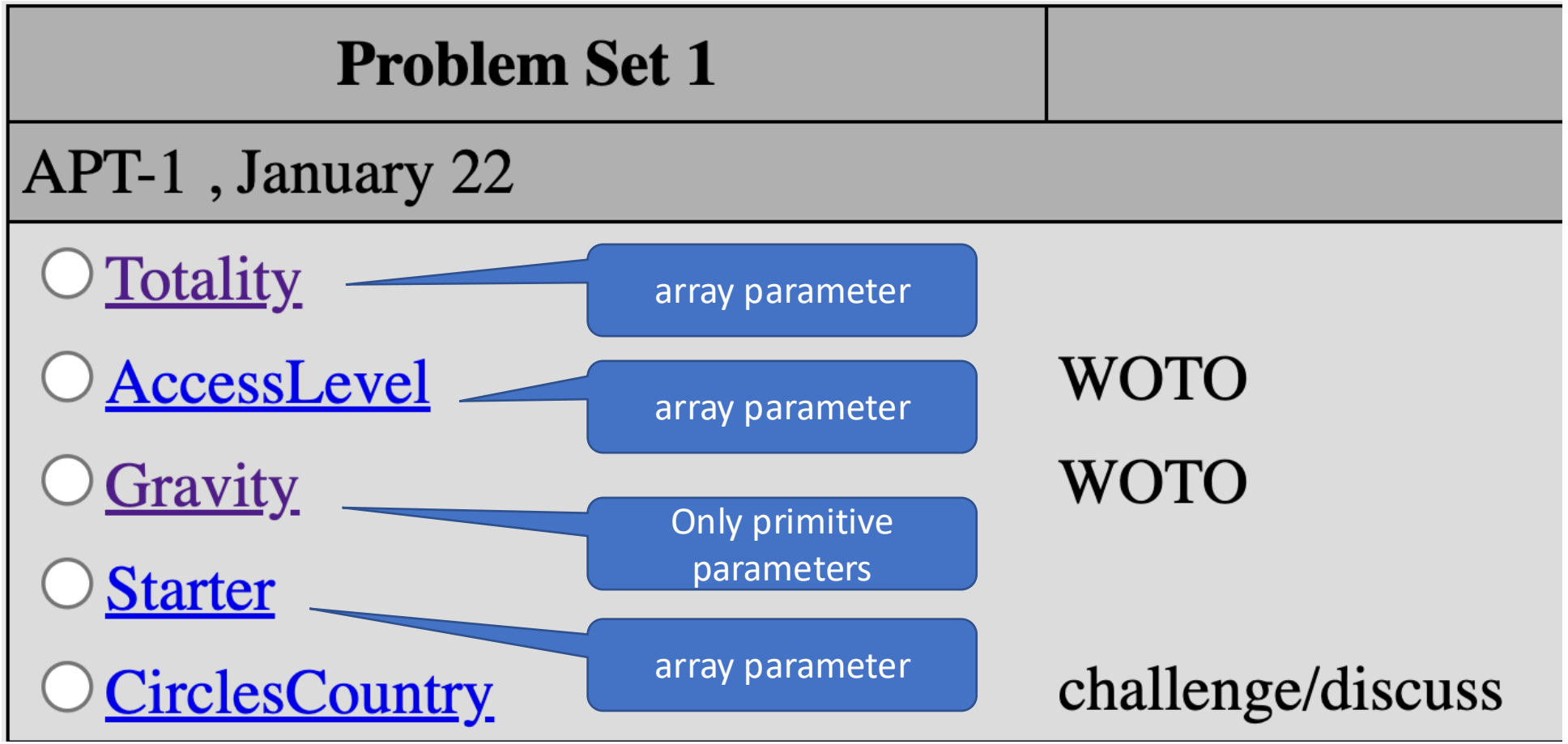
https://www.youtube.com/watch?v=1QmvBBMBurA



Green Dance, Compsci 101, Fall 2012

https://www.youtube.com/watch?v=YBAju5SE7q0&t=9s

Problems in APT-1



From Strings to Arrays in Java

- Array: fixed size collection, random access
  - Use index to store/get values, initial values: zero

```
int[] a = new int[5];
a[0] = 4;
a[4] = 7;
System.out.println(Arrays.toString(a));
```

Type of values

Is an object, **new** allocates memory

Length of array, `a.length`

```
[4, 0, 0, 0, 7]
```

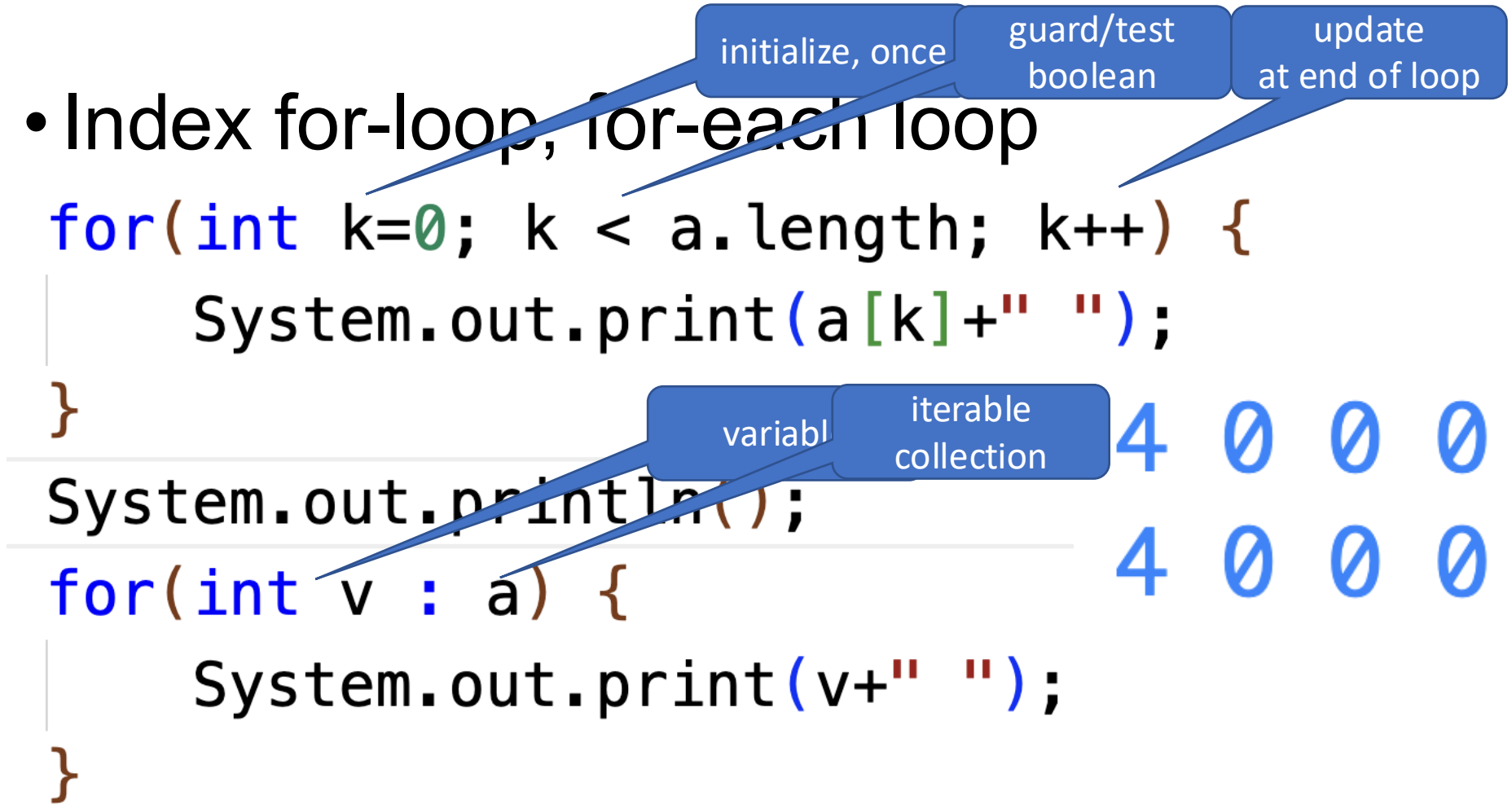
```
0  1  2  3  4
```

inline initializer

```
int[] b = {2, 4, 6, 8, 10};
```

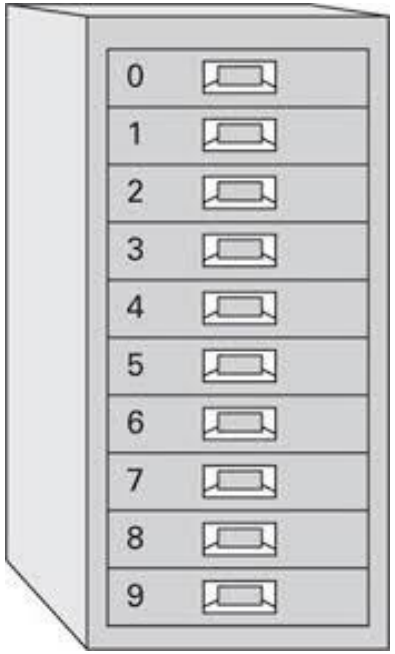
Looping to access array values

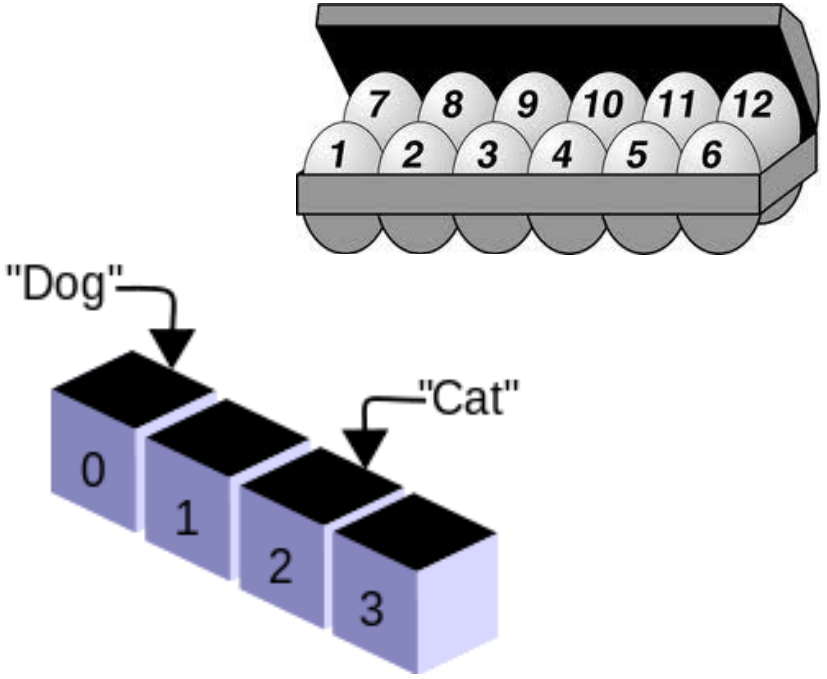
- Index for-loop, for-each loop



Metaphors and imagery

- What is an array?
  - homogeneous collection, random access





Arrays and ArrayLists (Preview)

- Arrays are fixed in size, cannot grow
- Arrays can store primitive and object types

- The class java.util.ArrayList can and cannot …
  - can grow dynamically, supports random access
  - cannot store primitives, only objects

java.util.ArrayList

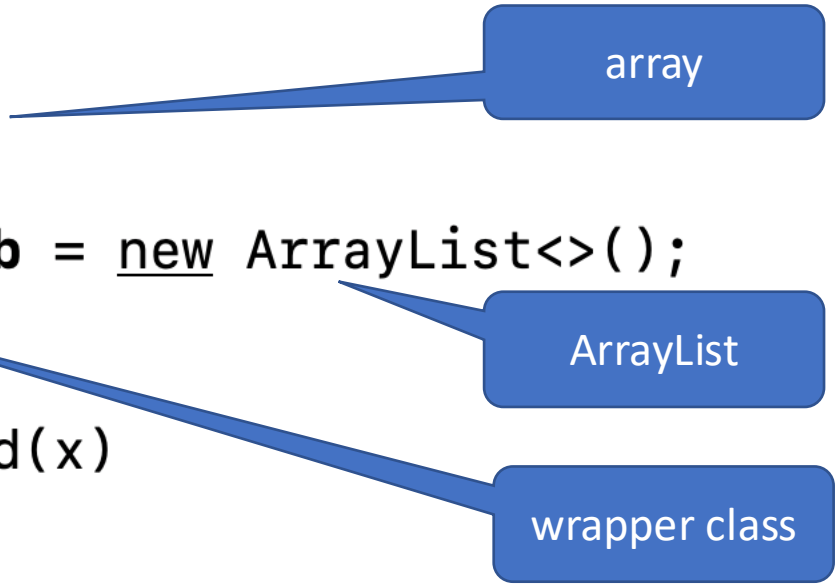
- More of this next class, not needed for APTs
  - Typically import java.util package to gain access

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

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

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

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



APT: AccessLevel (seen in Discussion)

- https://www2.cs.duke.edu/csed/newapt/accesslevel.html

```
public class AccessLevel {
   public String canAccess(int[] rights, int minPermission) {
          // fill in code here
   }
}
```

```
{34,78,9,52,11,1}

49


Returns: "DADADD"
```

- Think for a minute, talk, then …
  - Please, help me write the code

Coding Interlude

- Working on AccessLevel APT in VSCode

+

$\mathbb{Z}$

+

{…-1,0,1,2,…Integer.MAX_VALUE}

+

WOTO: L02-B on PrairieLearn

https://pl.cs.duke.edu

Fred Brooks: Computer Scientist

Turing Award '99, UNC/CS '64, Mythical Man-Month '75, Duke '53

The most important single decision I ever made was to change the IBM 360 series from a 6-bit byte to an 8-bit byte, thereby enabling the use of lowercase letters. That change propagated everywhere.

- "Fred Brooks" by Copyright owned by SD&M (www.sdm.de) - Request for picture sent by email to Fred Brooks by uploader (Mark Pellegrini; user:Raul654) Fred sent this photo back, along with contact information for Carola Lauber at SD&M, who gave copyright permission.. Licensed under CC BY-SA 3.0 via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:Fred_Brooks.jpg#/media/File:Fred_Brooks.jpg

Why is programming fun? FB says …

- First is the sheer joy of making things
- *Second is the pleasure of making things that are useful*
- Third is the fascination of fashioning complex puzzle-like objects of interlocking moving parts
- *Fourth is the joy of always learning*
- Finally, there is the delight of working in such a tractable medium. The programmer, like the poet, works only slightly removed from pure thought-stuff.

Learning is continual

Programming then is fun because it gratifies creative longings built deep within us and delights sensibilities we have in common with all …

ola learns from Kemeny