Classes, Objects, APTs
P0: Person201

https://201wo.to

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

Owen Astrachan and Alex Steiger

ola@duke.edu, alex.steiger@duke.edu

C is for …

- ***Class***
  - Factory for creating objects
- ***Collections and Collection***
  - See java.util.* for details
- ***Collaboration***
  - WOTO when appropriate

Administrivia

- This week: discussion on 1/16
- Next week:
  - No class on MLK day 1/19, video instead
  - APTs due 1/22 (Friday)
- Following week: P0 due Tuesday 1/27
  - Early Engagement, late policy 48 hours
  - More APTs due Thursday 1/29

Plan for next three classes+discussion

- How to program in Java by example?
  - Review Java to be able to do P0
  - Understand the "object concept"
  - **`Person201`** for P0 and engagement

- Today: Objects, Classes, Strings
  - Avoiding duplicates in APT Starter
  - (after L02 wrap-up)

WOTO (L02 B): https://duke.is/cs201-pl

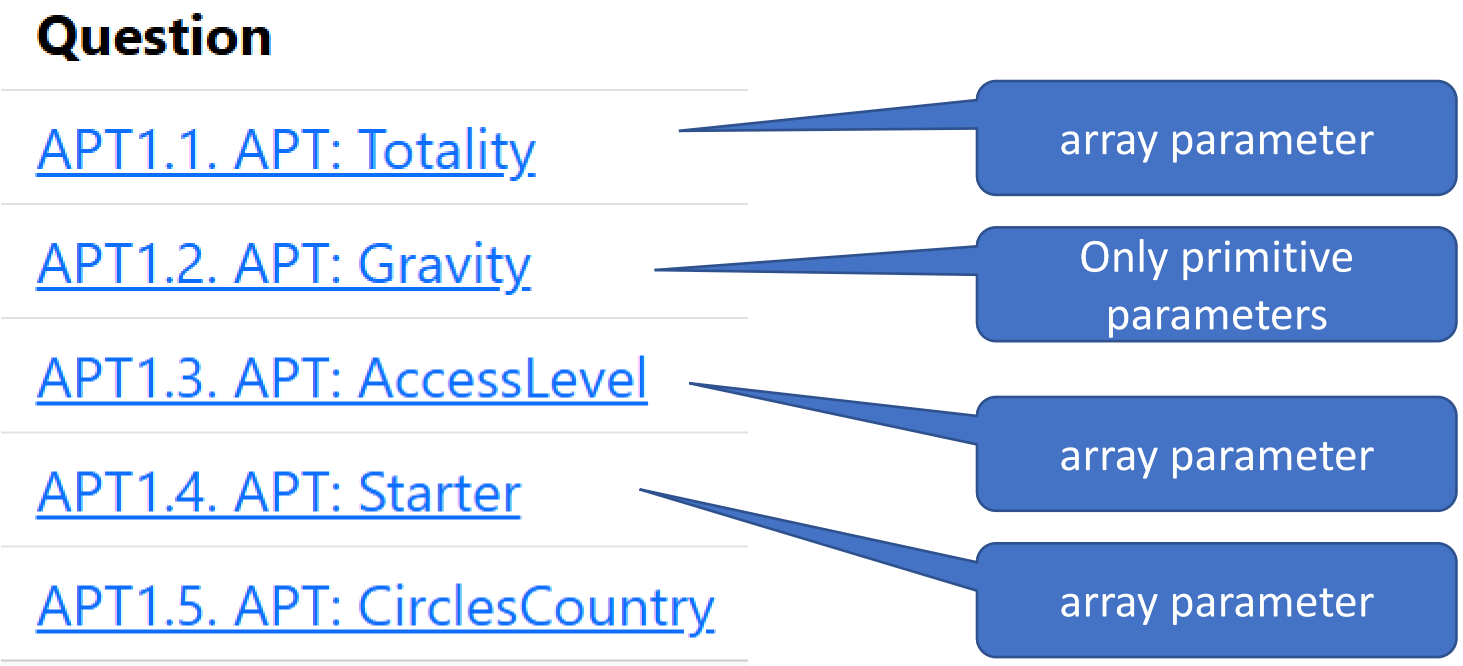
(Finish from last time)

Completing APTs

- Is getting all-green a requirement?
  - Can you do well without getting all green?

- http://thegreendance.com/

Problems in APT-1



Question
APT1.1. APT: Totality
APT1.2. APT: Gravity
APT1.3. APT: AccessLevel
APT1.4. APT: Starter
APT1.5. APT: CirclesCountry

array parameter

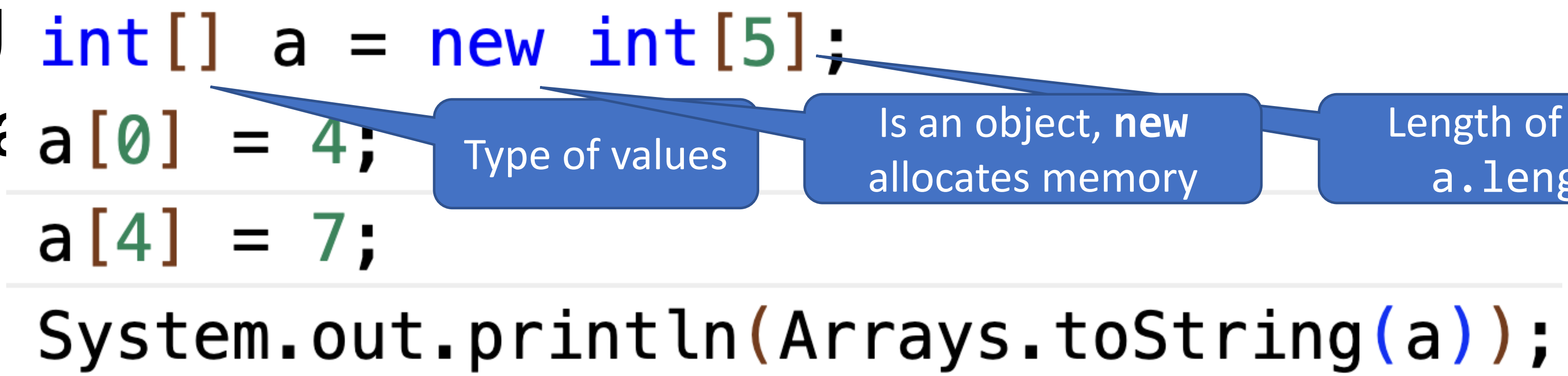
Only primitive parameters

array parameter

array parameter

From Strings to Arrays in Java

- Array: fixed size collection, random access
  - U
    va

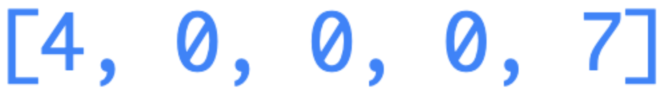


```
int[] a = new int[5];
a[0] = 4;
a[4] = 7;
System.out.println(Arrays.toString(a));
```

Type of values

Is an object, **new** allocates memory

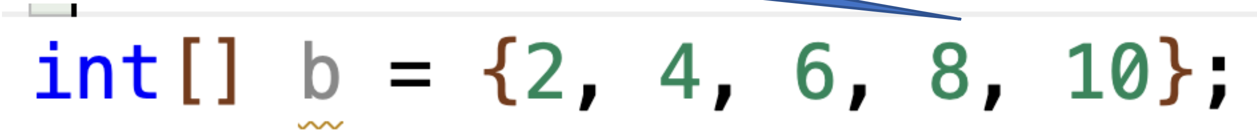
Length of array, `a.length`



```
[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

initialize, once

guard/test
boolean

update
at end of loop

```
for(int k=0; k < a.length; k++) {
    System.out.print(a[k]+" ");
}
System.out.println();
for(int v : a) {
    System.out.print(v+" ");
}
```

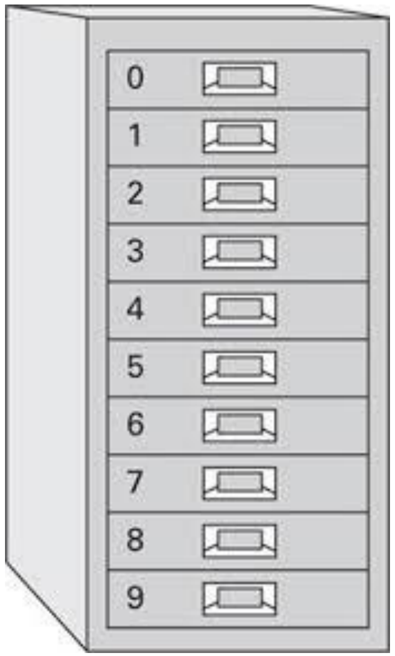
variabl

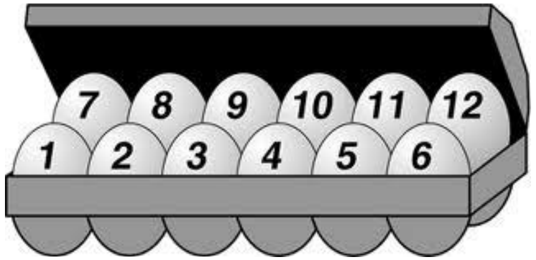
iterable
collection

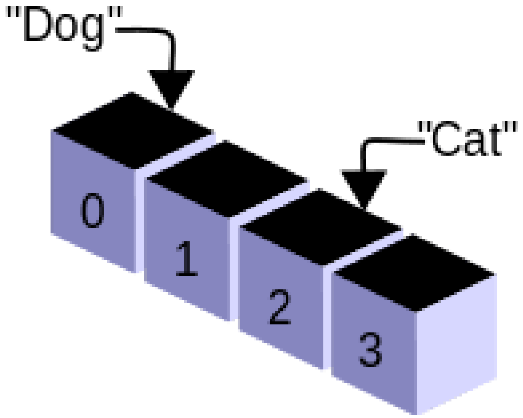
```
4 0 0 0 7
4 0 0 0 7
```

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

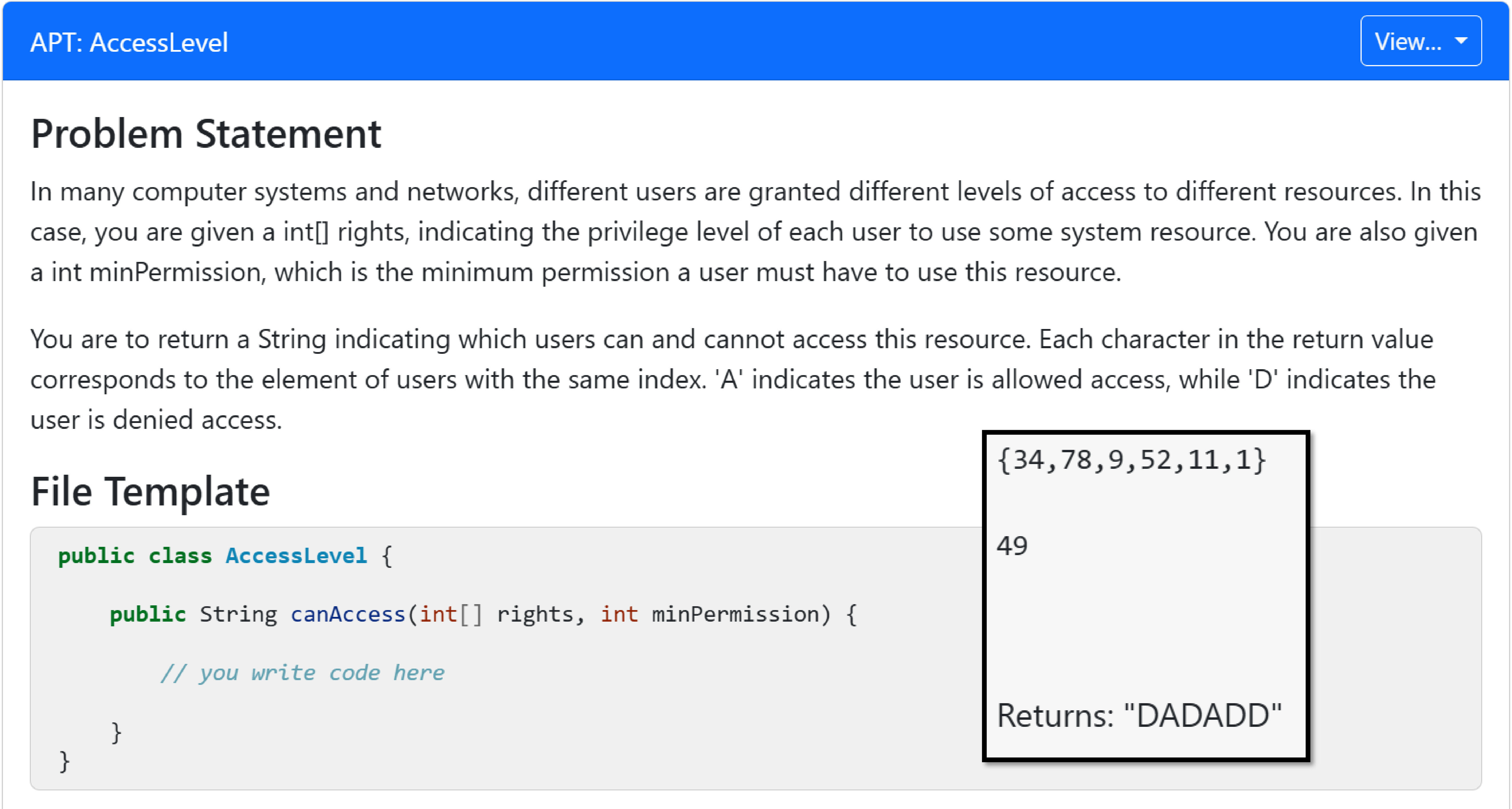
- More of this next class, not needed for APTs
  - Typically import java.util package to gain access

```
[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]
```

array

ArrayList

wrapper class



APT: AccessLevel

Problem Statement

In many computer systems and networks, different users are granted different levels of access to different resources. In this case, you are given a int[] rights, indicating the privilege level of each user to use some system resource. You are also given a int minPermission, which is the minimum permission a user must have to use this resource.

You are to return a String indicating which users can and cannot access this resource. Each character in the return value corresponds to the element of users with the same index. 'A' indicates the user is allowed access, while 'D' indicates the user is denied access.

File Template

```java
public class AccessLevel {

    public String canAccess(int[] rights, int minPermission) {

        // you write code here

    }
}
```

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

49

Returns: "DADADD"
```

Coding Interlude

- Working on AccessLevel APT in VSCode

WOTO (L02 B): https://duke.is/cs201-pl

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 …”

Data Structures, Algorithms, Java?

- Read and study example programs
  - Three from discussion
  - Several in class

- Use what you know, even if that's not Java
- Read resources
- Ask ChatGPT? No to code, Yes to questions

Classes in Java and in general

- Class named Foo stored in Foo.java
  - File+class names required by language

- Class is an ***object factory*** via new
- Class is blueprint for creating objects
  - Each ***object*** is different, ***blueprint*** same
  - String class: "hello", "Duke!"

Concept of a class in every OO language

- Class encapsulates state/behavior
  - String is array of char values
  - Point in 2-dim is x,y double values
  - Person201 discussed soon …

- ***State*** is the "innards" of object
- ***Behavior*** is how to interact with it

String state: array of char values

String method

String method

```
[jshell> String s = "computer"
s ==> "computer"

[jshell> s.toCharArray()
$37 ==> char[8] { 'c', 'o', 'm', 'p', 'u', 't', 'e', 'r' }
```

String method

String method

- **`s.length() == 8`**
- **`s.charAt(1) == 'o'`**
- **`s.substring(3,6).equals("put")`**

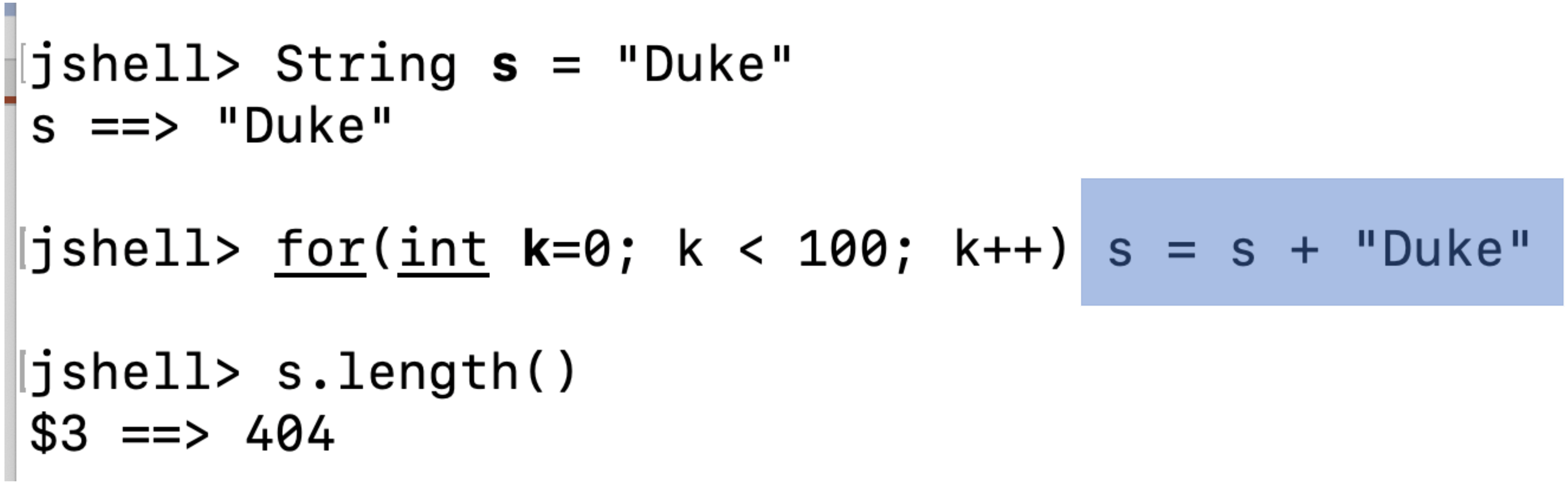
Immutable objects: String class

- String objects are ***immutable***
  - Once created can never change

- State can be accessed
  - `s.charAt(), s.startsWith()`
- State can't change: s references a new String
  - `s = "hello"; s = s + " world"`

How many Strings, how much memory?

- Final String, length can be found
- Total # chars in memory? 4 + 8 + … + 404
    - This is 4(1+2+…+101) = 4*(101*102)/2 = 20,604



```
jshell> String s = "Duke"
s ==> "Duke"

jshell> for(int k=0; k < 100; k++) s = s + "Duke"

jshell> s.length()
$3 ==> 404
```

String has getter methods

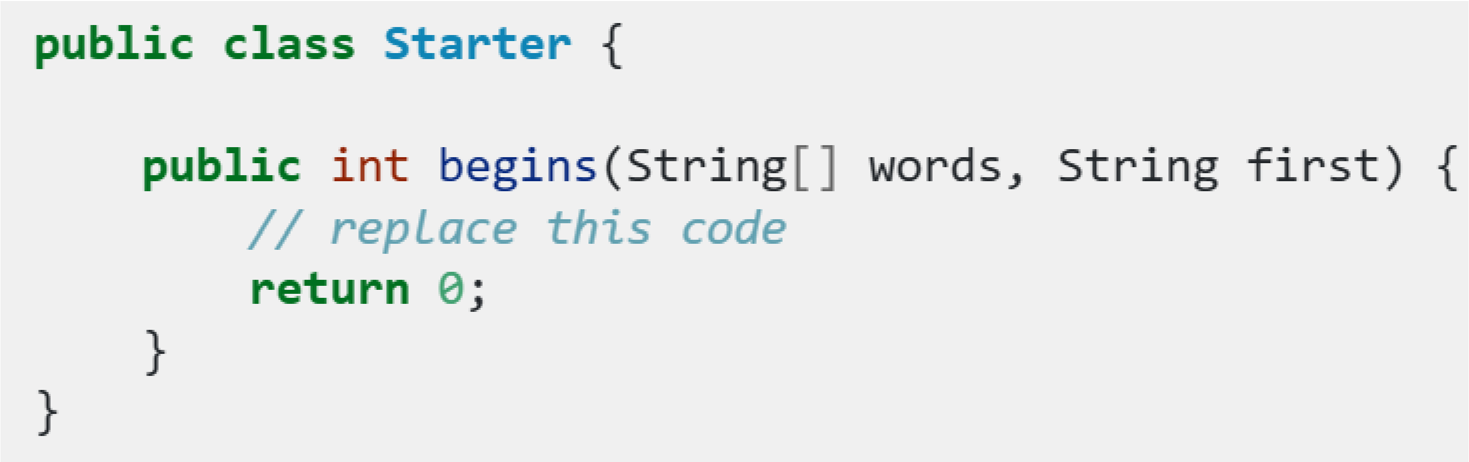
- Access state, different kinds of access
  - Length of string: `.length()`
  - Character at index: `.charAt(k)`
  - Substring: `.substring(first,last)`
  - Search for char or string: `.indexOf(x)`

- See Javadoc

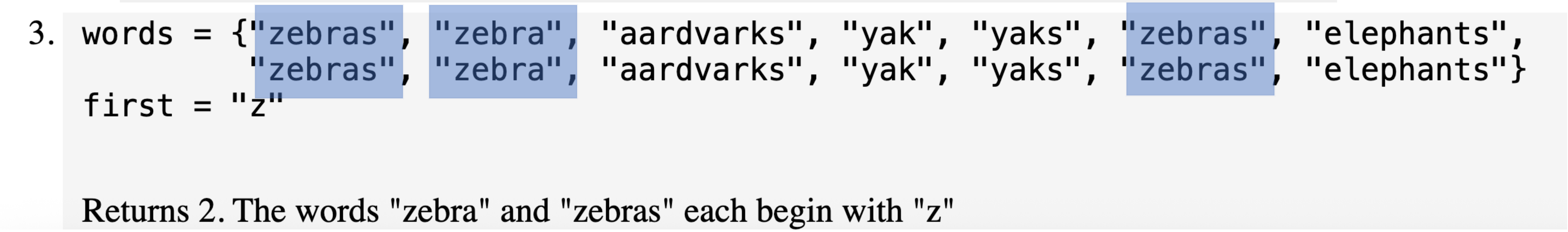
https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/lang/String.html

Starter APT

- How many Strings in `words` start with `first`?



```java
public class Starter {

    public int begins(String[] words, String first) {
        // replace this code
        return 0;
    }
}
```

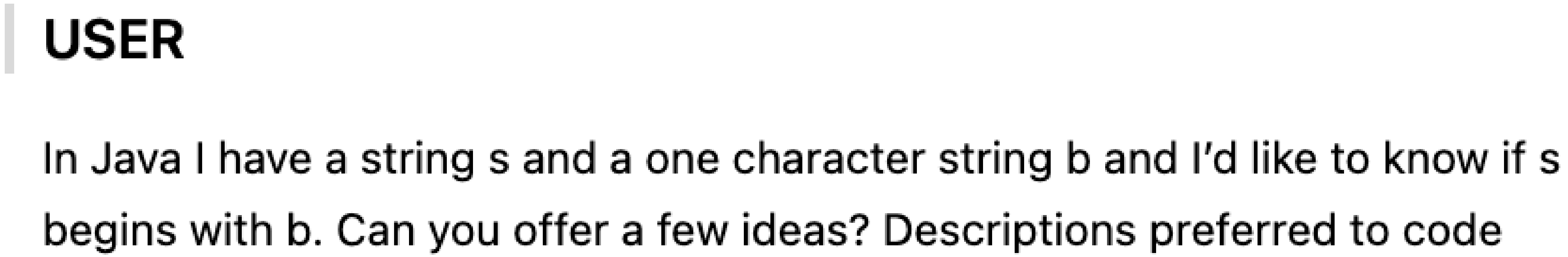


```
3.  words = {"zebras", "zebra", "aardvarks", "yak", "yaks", "zebras", "elephants",
             "zebras", "zebra", "aardvarks", "yak", "yaks", "zebras", "elephants"}
    first = "z"
```

Returns 2. The words "zebra" and "zebras" each begin with "z"

I don't know Java!

- Ask …

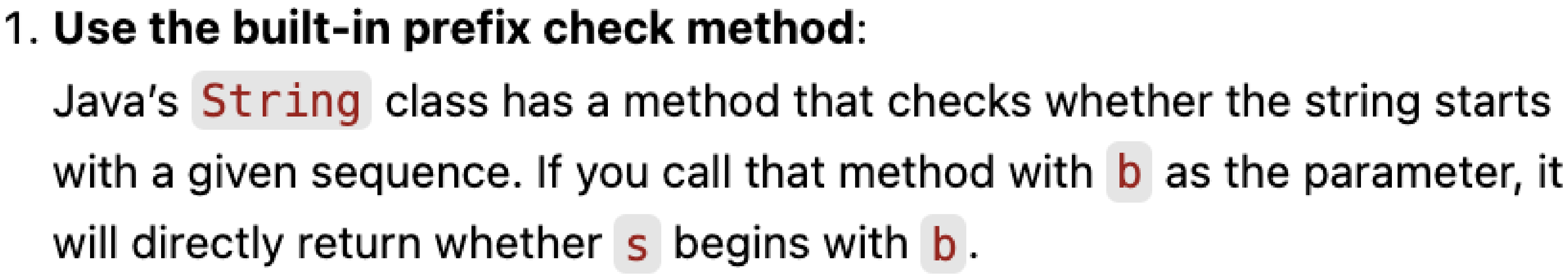


USER

In Java I have a string s and a one character string b and I’d like to know if s begins with b. Can you offer a few ideas? Descriptions preferred to code

Response from Chat/Duke GPT

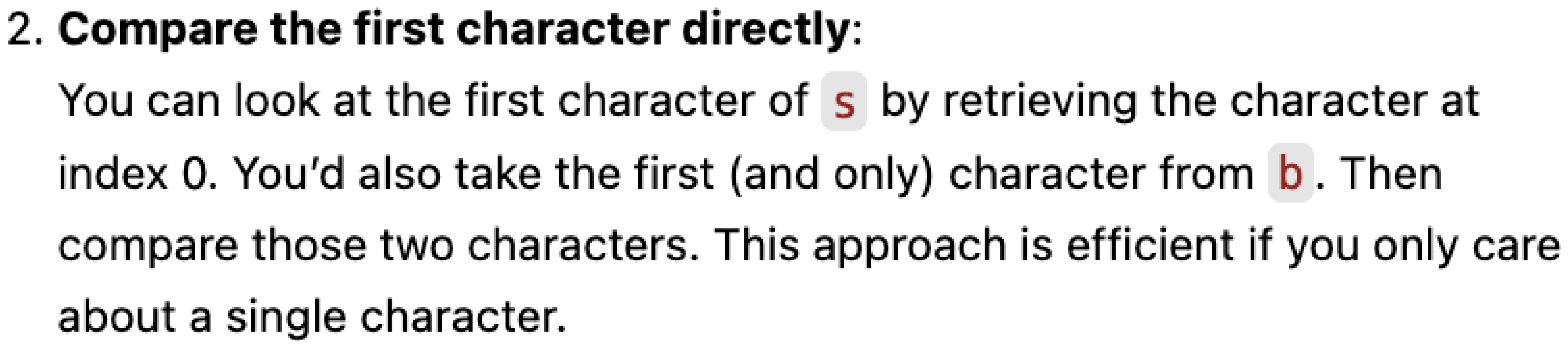
- Note: description, not code?



1. **Use the built-in prefix check method**:
   Java's `String` class has a method that checks whether the string starts with a given sequence. If you call that method with `b` as the parameter, it will directly return whether `s` begins with `b`.

Second response

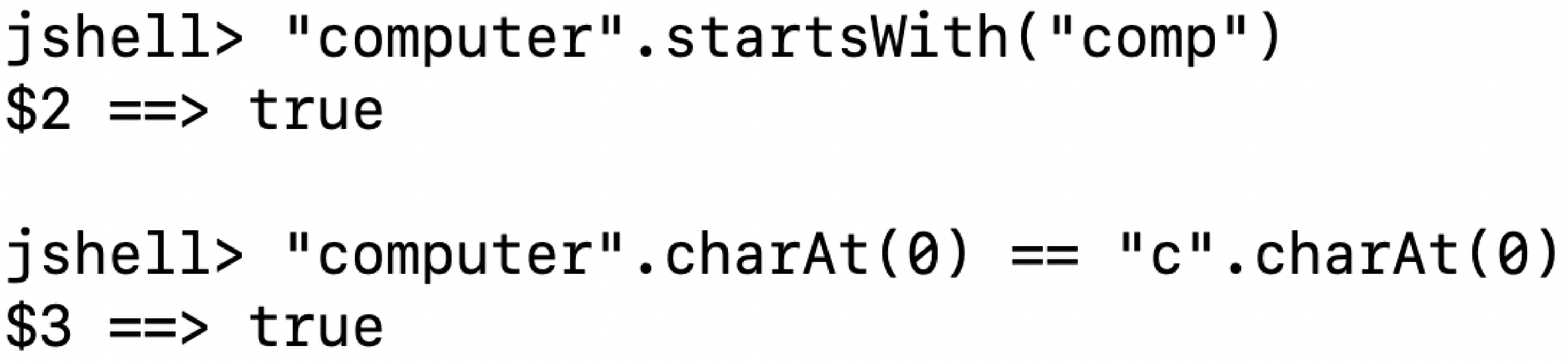
- Characters and Strings



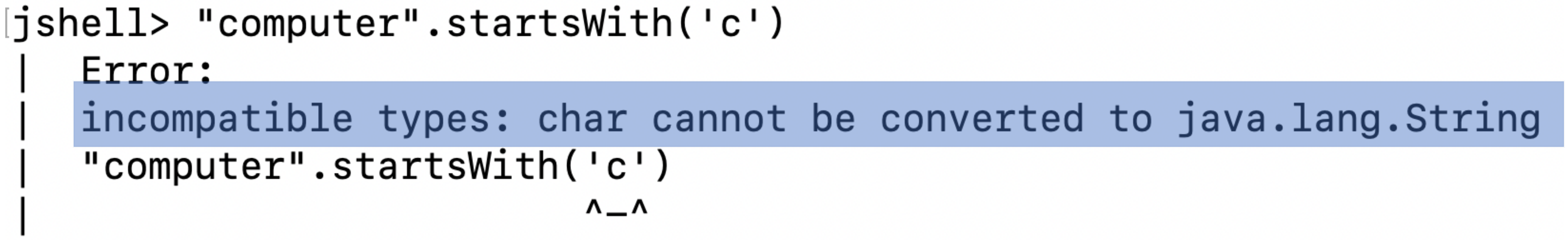
2. **Compare the first character directly**:
   You can look at the first character of `s` by retrieving the character at index 0. You’d also take the first (and only) character from `b`. Then compare those two characters. This approach is efficient if you only care about a single character.

Methods and algorithms

- Does a String start with …



```
jshell> "computer".startsWith("comp")
$2 ==> true

jshell> "computer".charAt(0) == "c".charAt(0)
$3 ==> true
```



```
jshell> "computer".startsWith('c')
|  Error:
|  incompatible types: char cannot be converted to java.lang.String
|  "computer".startsWith('c')
|                        ^-^
```

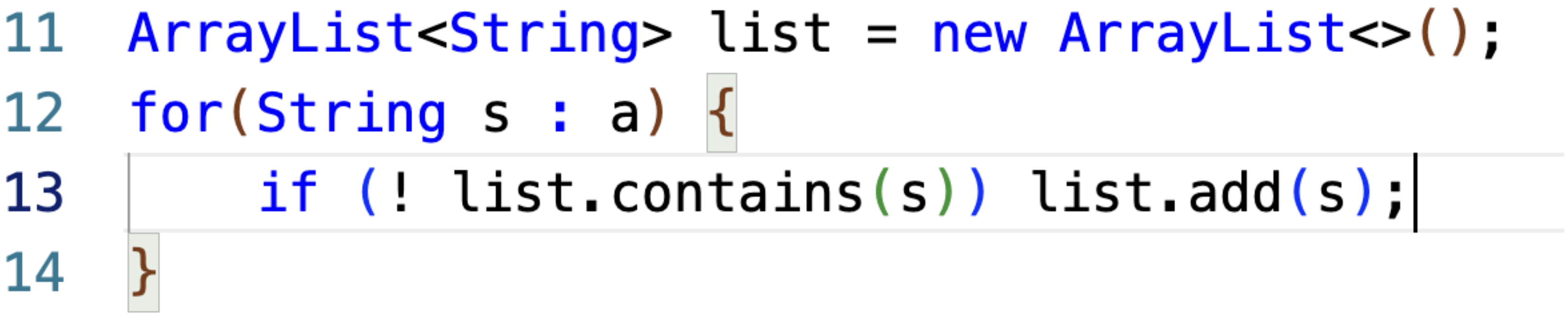
How to avoid duplicates

- Problem: Count each word once
  - How to avoid duplicates?

- Transform input: No duplicates
- Alternative: First occurrence only

Transform input array: no duplicates

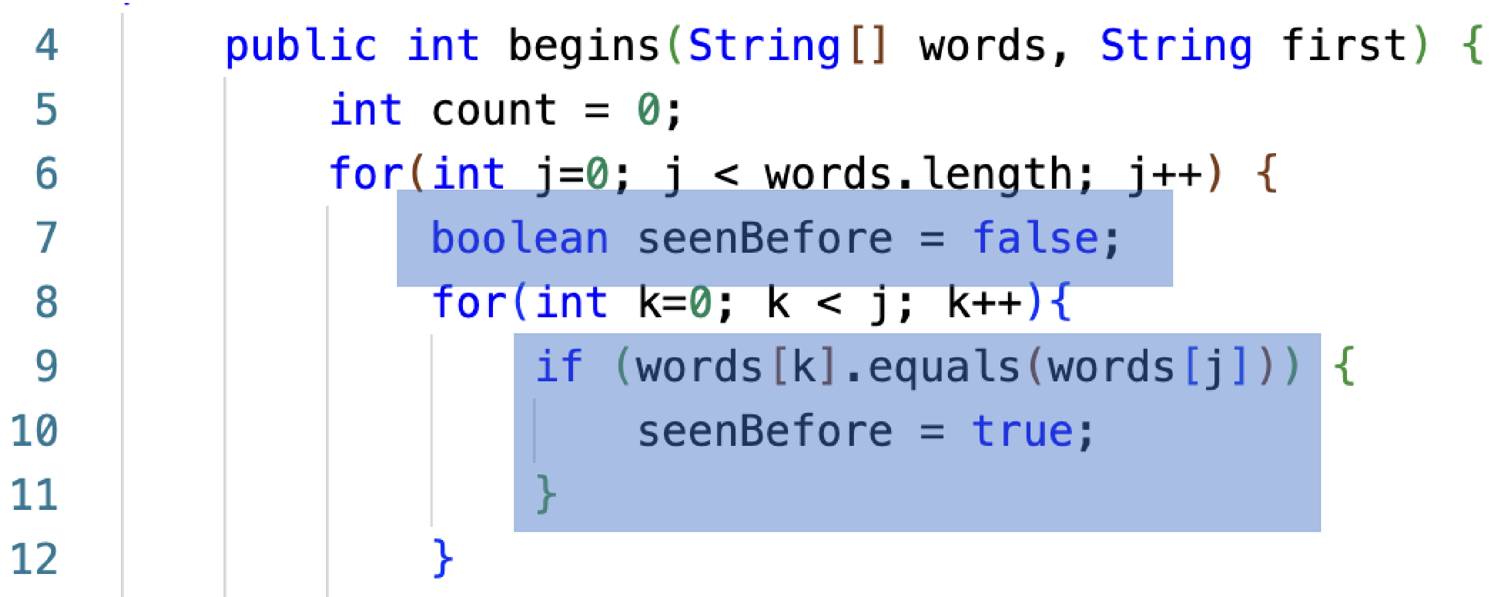
- Create list with no duplicates
  - Why ArrayList and not array?
  - What about HashSet? (good alternative)



```java
ArrayList<String> list = new ArrayList<>();
for(String s : a) {
    if (! list.contains(s)) list.add(s);
}
```

Alternative: only count "first occurrence"

- On line 13, if **`seenBefore == true`**?
  - We've already processed **`words[j]`**



```
 4      public int begins(String[] words, String first) {
 5          int count = 0;
 6          for(int j=0; j < words.length; j++) {
 7              boolean seenBefore = false;
 8              for(int k=0; k < j; k++){
 9                  if (words[k].equals(words[j])) {
10                      seenBefore = true;
11                  }
12              }
```

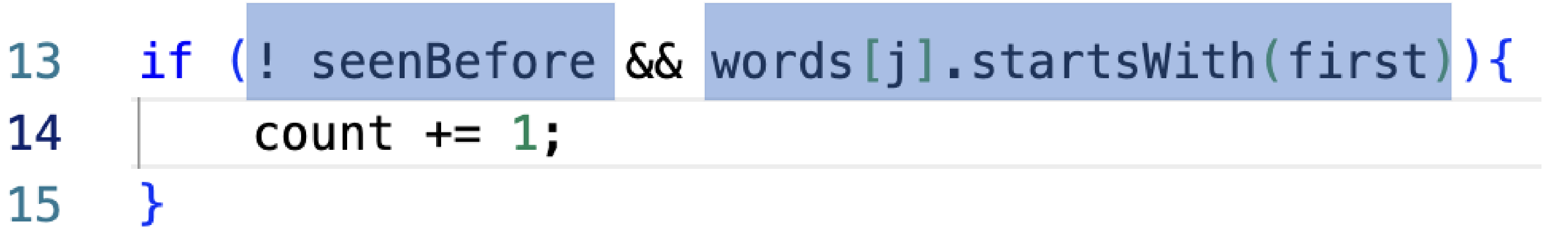
Boolean "flag" variable

- Value of variable determines "state"
  - Word has been seen before
- Initialize to false before loop, change value inside loop
  - Once true, can "break" out of loop
- Use value after loop with code

Check flag variable after loop

- After loop, if words[j] hasn't been seen before? Check first letter
  - Reason about initial/before loop value
  - Add "break"?

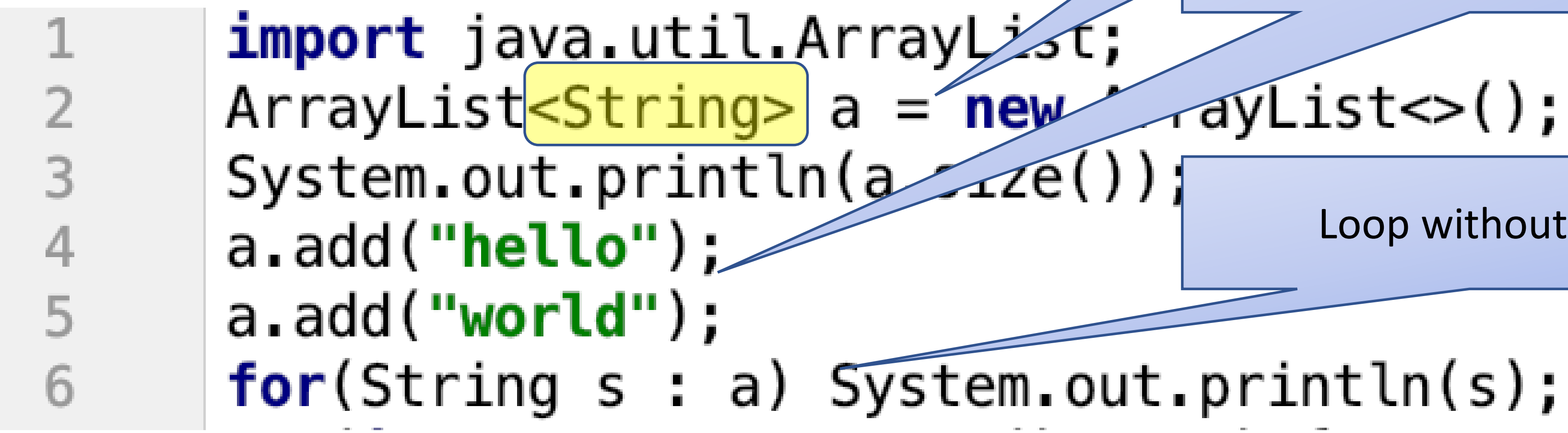
only checked if ! seenBefore



```
if (! seenBefore && words[j].startsWith(first)){
    count += 1;
}
```

ArrayList Example I

- Package for ArrayList class
  - Declaration/Definition
  - Adding
  - For-each loop



```java
import java.util.ArrayList;
ArrayList<String> a = new ArrayList<>();
System.out.println(a.size());
a.add("hello");
a.add("world");
for(String s : a) System.out.println(s);
```

import class to use in code

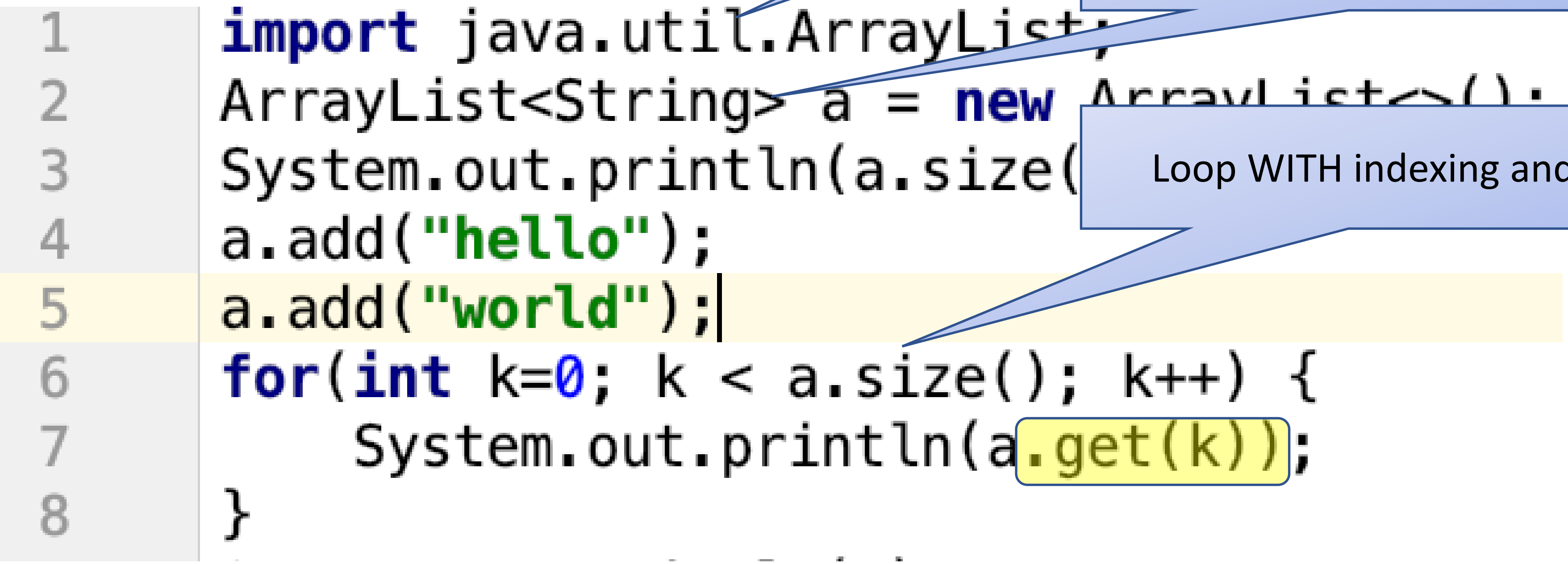
<Generic> lhs/rhs

add at end, grow if needed

Loop without indexing

ArrayList Example II, indexing

- Using indexing loop
  - Must use `.get()` method



```
import java.util.ArrayList;
ArrayList<String> a = new ArrayList<>();
System.out.println(a.size(
a.add("hello");
a.add("world");
for(int k=0; k < a.size(); k++) {
    System.out.println(a.get(k));
}
```

import class for use

<Generic> lhs/rhs

Loop WITH indexing and .get

java.util.ArrayList

- Like an array, but can "grow"
  - other methods:**`.indexOf(x)`**, **`.contains(x)`**, **`.clear()`**, more

- Homogeneous collection supporting random access, but …
  - ***Cannot store primitive types***
  - Cannot use **`a[k]`** to access elements

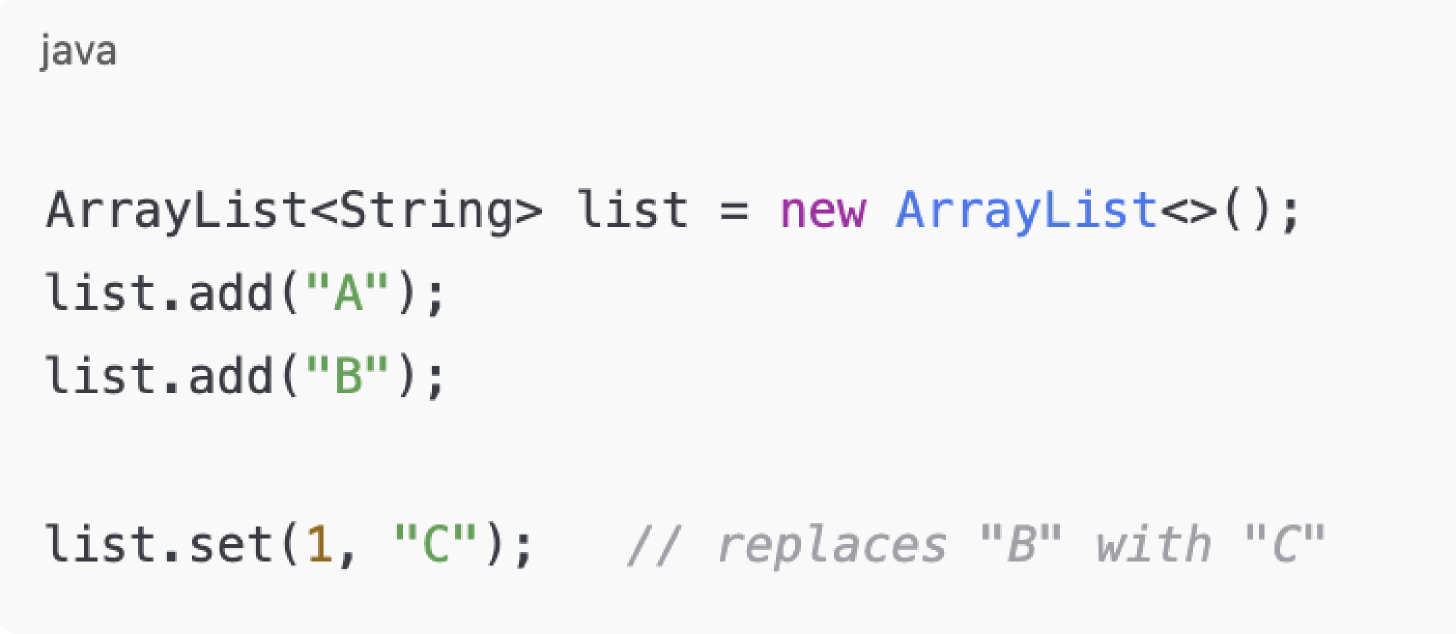
ArrayList methods reference

Method	Notes
`add(element)`	Appends `element` to end of list
`get(index)`	Returns the `index` position element (starting with 0)
`contains(element)`	Searches list, returns `true` if `element` is in the list, else `false`.
`size()`	Returns the (integer) number of elements in the list
`set(index, element)`	Assigns `element` to the `index` position (starting at 0), overwriting the previous value.
`remove(index)`	Remove the `index` position element

See the full ArrayList documentation, or ask ChatGPT?

Dialog with an LLM

- In ArrayList, how to set value at a specific index?

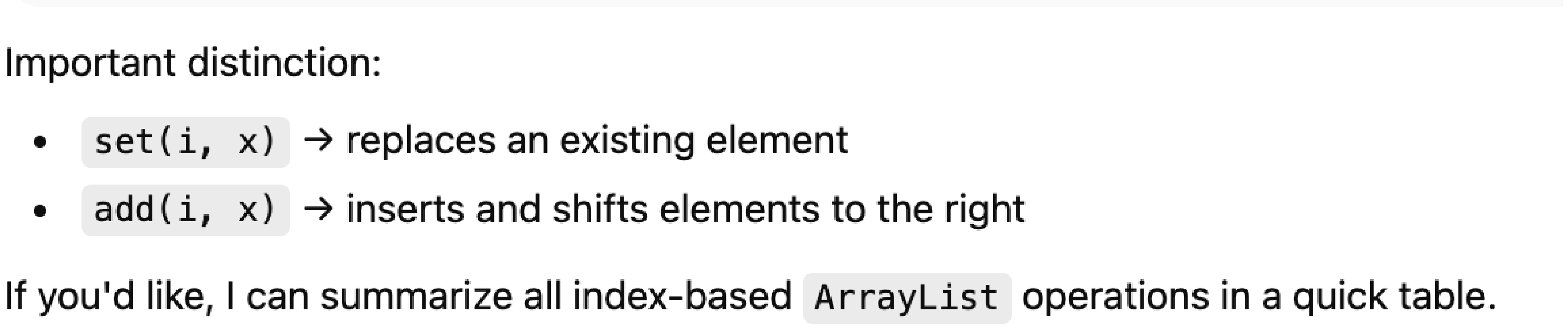


java

```java
ArrayList<String> list = new ArrayList<>();
list.add("A");
list.add("B");

list.set(1, "C");   // replaces "B" with "C"
```

Does LLM help too much?

- Is the LLM smart?
  - Absolutely not
- Does it sometimes appear that way?
  - YMMV



Important distinction:

- `set(i, x)` → replaces an existing element
- `add(i, x)` → inserts and shifts elements to the right

If you'd like, I can summarize all index-based `ArrayList` operations in a quick table.

WOTO (L03 A): https://duke.is/cs201-pl

P0: Person201 class and objects

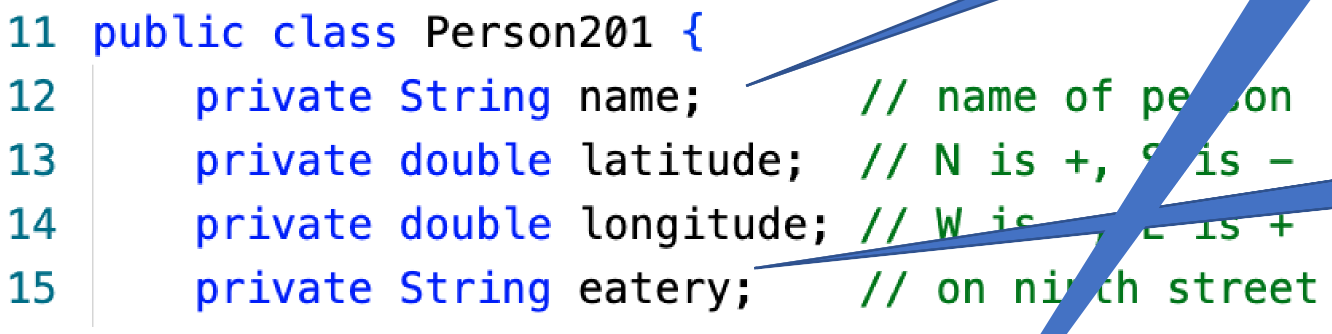
- Name, latitude, longitude, eatery
  - will collect data from class
    - privacy and anonymity an issue?

- Illustrate multi-class project
  - Read files, URLS, JSON format
  - Practice with Git, Gradescope, coding

Class encapsulates state and behavior

per object

- State specified by *instance variables*



```
public class Person201 {
    private String name;       // name of person
    private double latitude;   // N is +, S is -
    private double longitude;  // W is -, E is +
    private String eatery;     // on ninth street
```

per object

- Specific to each object *instance*
  - claire from Oakland, ricardo from Nairobi

```
Person201 a = new Person201("claire",37.8044,-122.2712,"Vin Rouge");
Person201 b = new Person201("ricardo",-1.2921, 36.8219,"Elmo's Diner");
```

Immutable classes

- String and Person201 are *immutable*
  - Once created can never change

- State can be accessed for **`Person201 p`**
  - **`p.name(), p.latitude()`**

- Create new String? new Person201? YES!!
- Change a string? Change Person201? NO!!

Constructor

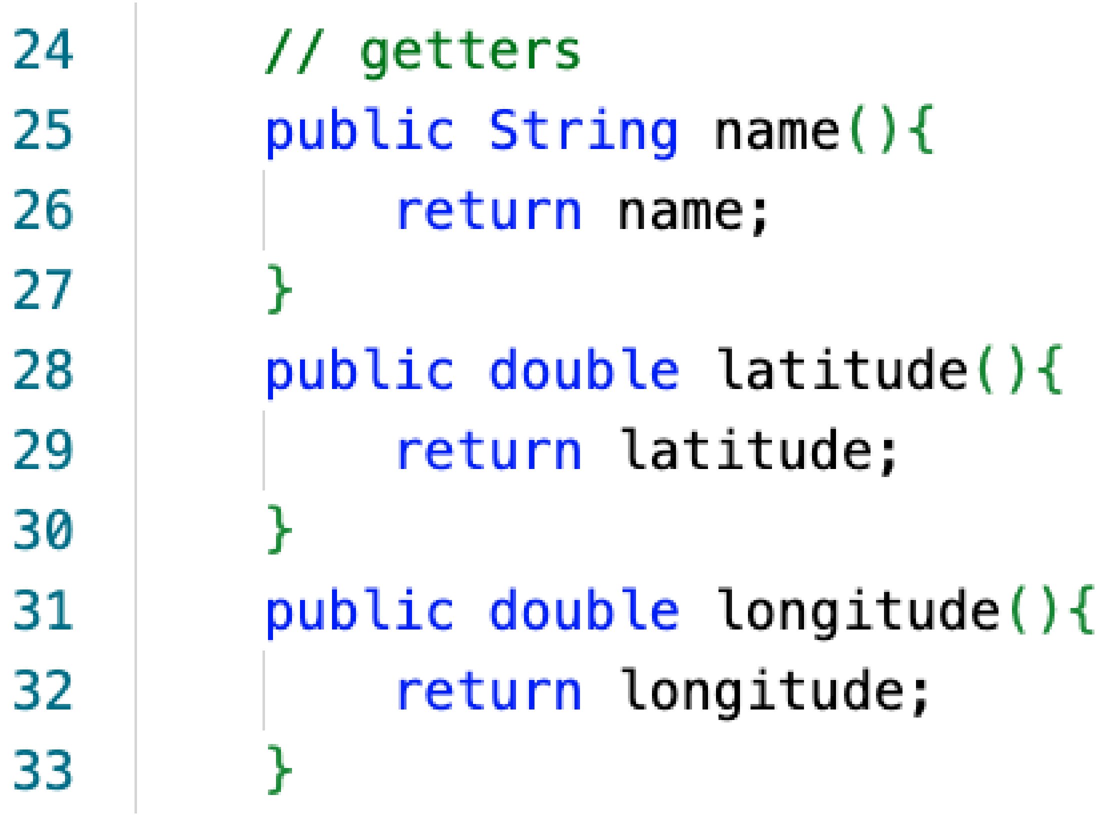
- Same name as class
  - No return type
- **`Person201(String, double,double,String)`**
  - Initialize all instance variables
  - *Default constructor* has no parameters
    - Who will this person be? How to initialize?
    - **`Person201 p = new Person201();`**

Person201 behavior/methods

- Cannot access private **`p.name`** in client code

getter method

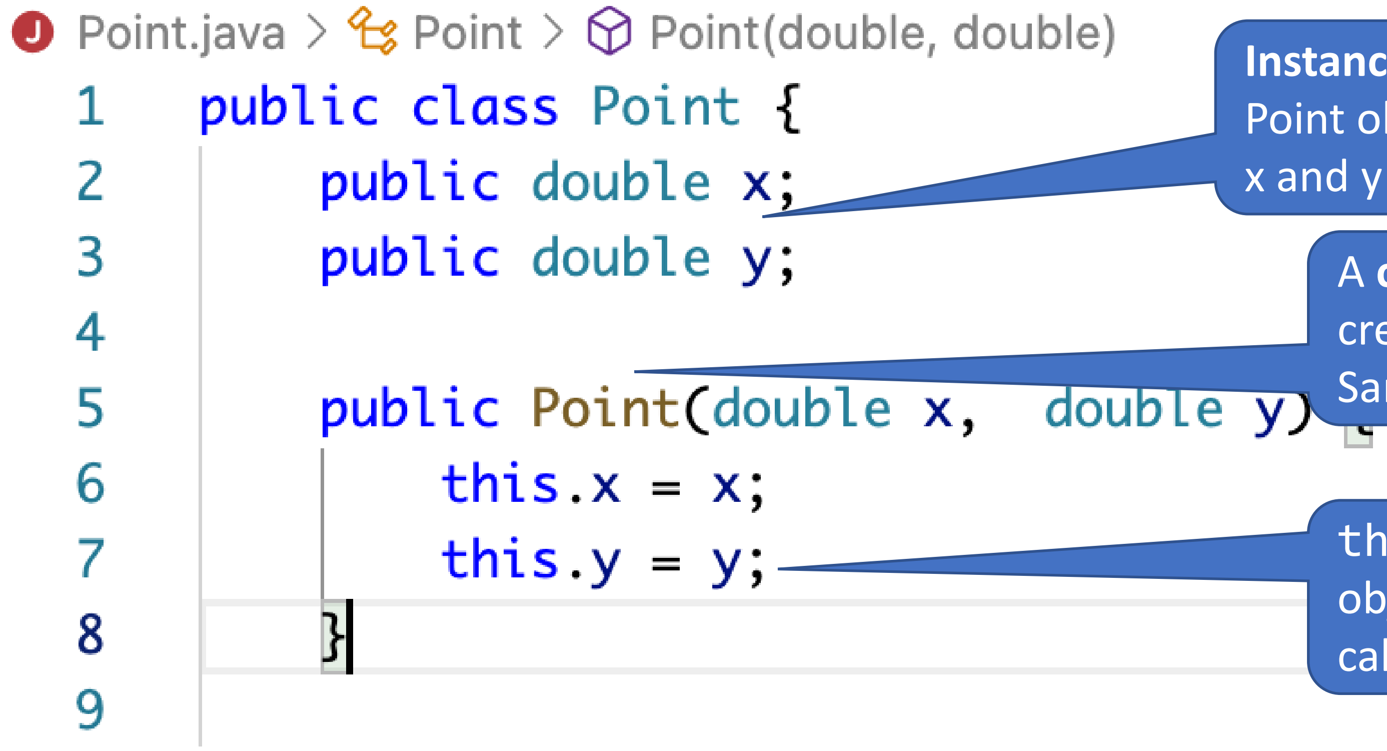
getter method



```
    // getters
    public String name(){
        return name;
    }
    public double latitude(){
        return latitude;
    }
    public double longitude(){
        return longitude;
    }
```

Example of Point class

`Point p = new Point(3.0, 6.0)`



```java
public class Point {
    public double x;
    public double y;

    public Point(double x,  double y) {
        this.x = x;
        this.y = y;
    }

```

Instance variables. Each Point object has its own x and y value.

A **constructor** specifies how to create a new Point object. Same name as class.

`this` keyword refers to object on which method is called.

Why is state private?

- Best practice that state is private
  - Cannot modify in an immutable class
  - Can only modify intentionally otherwise

- Immutable objects important concept
  - Performance gains, correctness gains

- Carries over, implementation hidden

P0: Workflow and Details

- Main learning objectives
  - Understand project workflow: git, commit, push
  - Understand project grading: gradescope
  - Understand projects: Analysis Questions

- Simple Java classes and interactions
  - Augmented with JSON and post/get code

Dr. Joy Buolamwini

- Founded Algorithmic Justice League
  - Rhodes Scholar, Anita Borg Scholar
  - TedX: Fighting Algorithmic Bias
  - Facial Recognition Bias

And so in exploring this [facial recognition], I could have viewed my face not being consistently detected as, “Oh, this is a *technical* challenge” — but being in the space of the Center for Civic Media definitely orients me to [say], “This is not just a technical challenge … this is as much a reflection of society as other spaces where you see inequities that need to be addressed.”