LEC 14

CSE 121

Reference Semantics

Questions during Class?

Raise hand or send here

sli.do #cse121

BEFORE WE START

Talk to your neighbors:

What's your favorite sport or competition to watch?

Music: CSE 121 26au Lecture Tunes

Instructor: Miya Natsuhara and Anjali Pal

TAs:

Samrutha	Tamsyn	Owen	Jennie
William	Cedric	Jameson	Anisha
Samvedha	Anum	Paul	TJ
Syreeta	Navya	Gurbax	Ivory
Ryan	Gene	Tony	Sarah W
Anant	Margot	Lucas	Avery
Henry	Sarah K	Alex	
Reese	Ethan	Jesse	
Savannah	Evelyn	Matthew	

Announcements, Reminders

- Computer Quiz 2 currently ongoing (11/12 – 11/18)
  - If you have not signed up for a timeslot yet, **do it today**
- Paper Quiz 2 scheduled for 11/17 in your quiz section
- Today is my last lecture!
  - I will still be around for the next week or so (probably), but will not be giving any more lectures
  - This means you *really really really* need to email us via the shared instructor email: cse121-instructors@cs.washington.edu
- Anjali will have bookable 1-on-1 slots starting next week!
- Winter 2027 new course preview: CSE 170!

(❄ Winter 2027) CSE 170

***CSE 170**: Principles, Applications, and Impacts of AI*

A new 4-credit course in Winter 2027 that provides a broad introduction to artificial intelligence (AI), focused on **critical AI literacy**

Open to all undergraduate students! No prior experienced required.

Meeting Times: MWF 2:30pm – 4:20pm

Instructor: James Weichert

cs.uw.edu/170

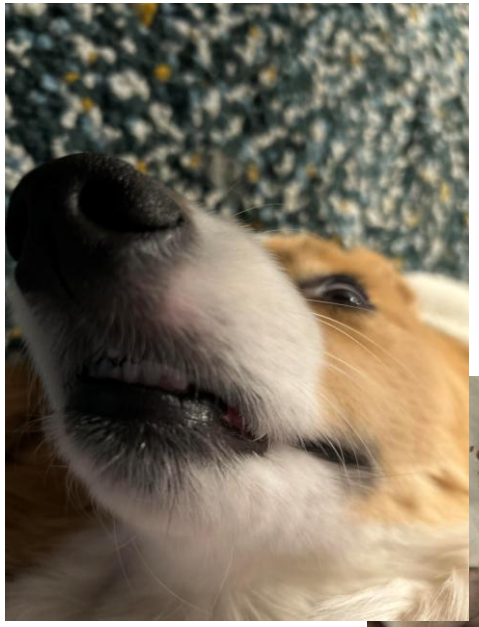
Feedback & Closing the Loop: The Good





- Pre-class materials & walkthrough videos
- Quiz sections and your TAs! (esp. walking through code)
- Practice quizzes
- Lectures and interactive activities!

Feedback & Closing the Loop: Suggestions





Suggestions that we're already working on:

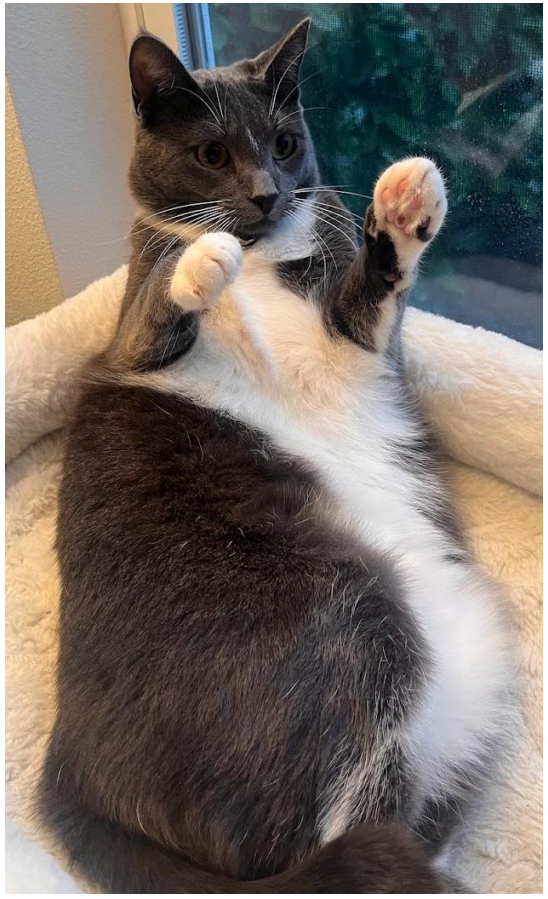
- Balance between "lecture" and "active learning" in class
- Frequency of quizzes
- More discussion of code quality and commenting in class
- More pet pics 🐾

Other suggestions…

- Grading system (ESNU)
- Shorten project instructions
  - Miya's project from Microsoft

Feedback & Closing the Loop: Transfer

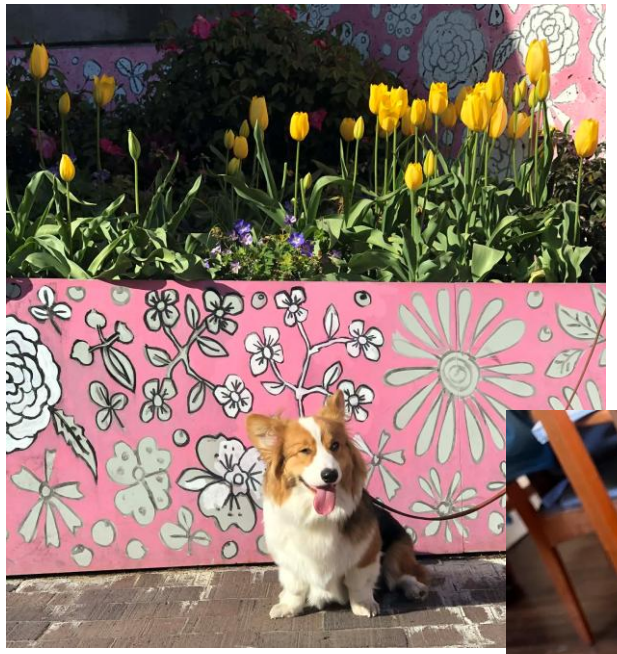




Transfer: how can you use your knowledge on new tasks (not ones that you've seen before)

There's a balance here – we want you to be able to to **transfer** knowledge to new problems, but we also want **alignment between lessons, practice, and assessment.**

Feedback & Closing the Loop: Reminders





- IPL and office hours are great places to get help on projects or review concepts!
- Practice problems
  - Section problems
  - "Extra Practice" Ed lessons
  - Section worksheets (linked from the course website)
- Recommendations for preparing for quizzes
  - Computer Quizzes vs. Paper Quizzes
  - Practice quiz should be your **last step!**



Practice: Think

sli.do #cse121

What would the array `arr` store at the end of this `arrayMystery` method if `{-20, 20, 26, 32, 50, 3}` was passed in?

```
public static void arrayMystery(int[] arr) {
  for (int i = arr.length - 1; i >= 1; i--) {
    if (arr[i] > arr[i - 1] + 10) {
      arr[i - 1] = arr[i - 1] + 5;
    }
  }
}
```

A. `{-20, 20, 26, 32, 50, 3}`

B. `{-15, 25, 31, 37, 55, 8}`

C. `{-15, 25, 31, 37, 50, 3}`

D. `{-15, 20, 26, 37, 50, 3}`



Practice: Pair

sli.do #cse121

What would the array `arr` store at the end of this `arrayMystery` method if `{-20, 20, 26, 32, 50, 3}` was passed in?

```
public static void arrayMystery(int[] arr) {
  for (int i = arr.length - 1; i >= 1; i--) {
    if (arr[i] > arr[i - 1] + 10) {
      arr[i - 1] = arr[i - 1] + 5;
    }
  }
}
```

A. `{-20, 20, 26, 32, 50, 3}`

B. `{-15, 25, 31, 37, 55, 8}`

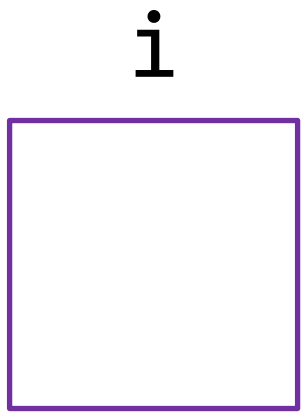
C. `{-15, 25, 31, 37, 50, 3}`

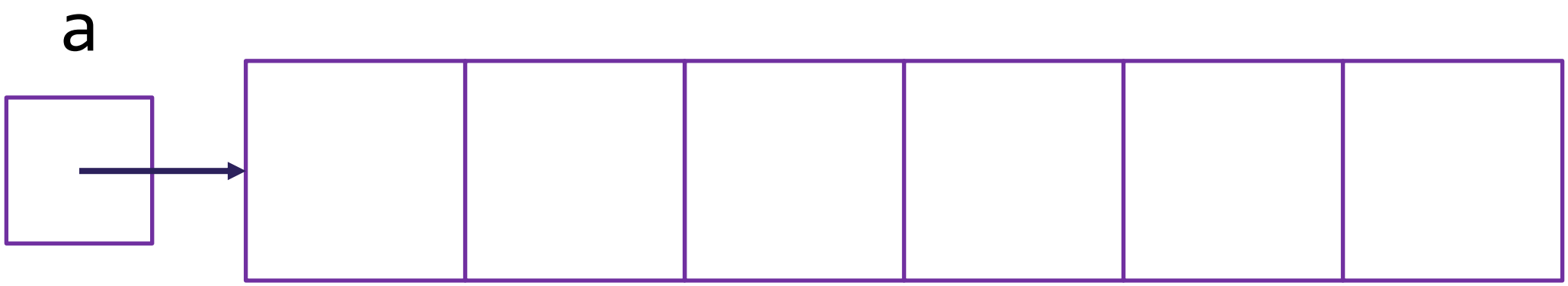
D. `{-15, 20, 26, 37, 50, 3}`

Tracing through arrayMystery

{-20, 20, 26, 32, 50, 3}

```
public static void arrayMystery(int[] a) {
  for (int i = a.length - 1; i >= 1; i--) {
    if (a[i] > a[i - 1] + 10) {
      a[i - 1] = a[i - 1] + 5;
    }
  }
}
```



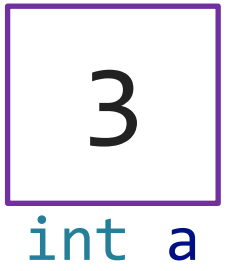


PCM Review: Value Semantics vs. Reference Semantics

Value Semantics

- Applies when working with primitive types
- Variables/parameters hold a *copy* of the actual value

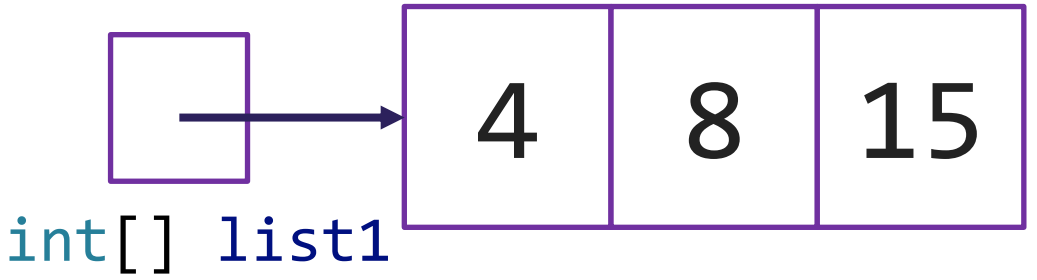
```
int a = 3;
```



Reference Semantics

- Applies when working with objects
- Variables/parameters hold a *reference* to the object

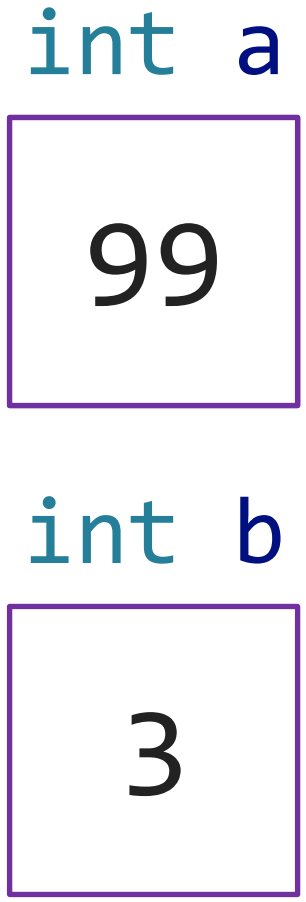
```
int[] list1 = {4, 8, 15};
```



PCM Review: Value Semantics

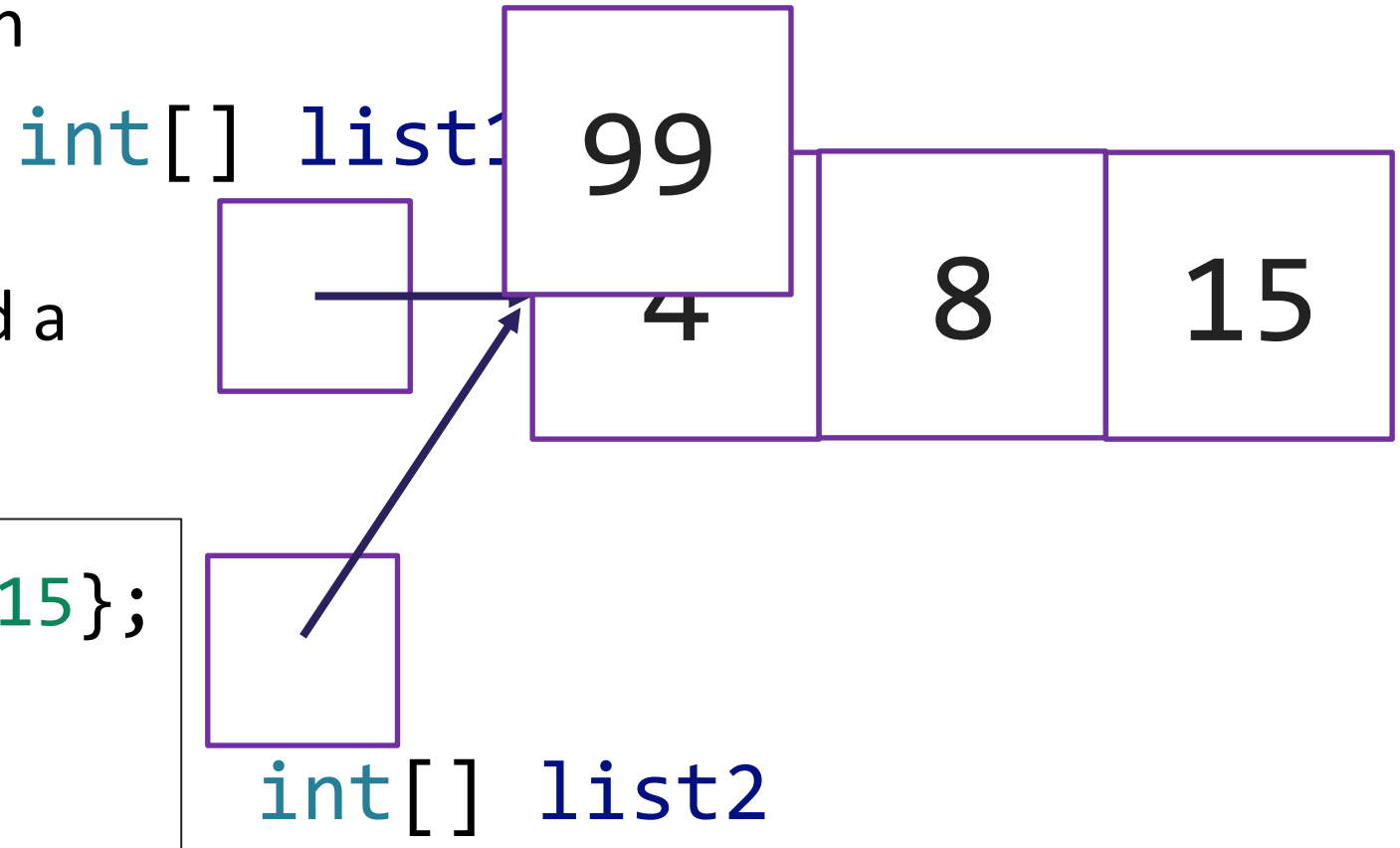
- Applies when working with primitive types
- Variables/parameters hold a *copy* of the actual value

```
int a = 3;
int b = a;
a = 99;
```



PCM Review: Reference Semantics

- Applies when working with objects
  - Including arrays!
- Variables/parameters hold a *reference* to the object




```
int[] list1 = {4, 8, 15};
int[] list2 = list1;
list1[0] = 99;
```

Value Semantics & Methods

```
boolean test = true;
flipValue(test);

public static void flipValue(boolean b) {
  b = !b;
}
```

Reference Semantics & Methods

```
boolean[] tests = {true, false, false, false};
flipValues(tests);

public static void flipValues(boolean[] b) {
  for (int i = 0; i < b.length; i++) {
    b[i] = !b[i];
  }
}
```

PCM Review: `null`

`null` is the absence of a reference!

- sort of the "zero" for references
- default value for object types (e.g. `Random`, `Scanner`, and `String`)

A **`NullPointerException`** is an error that happens when you ask `null` to "do something", which includes:

- calling `.toUpperCase()` on `null`? **`NullPointerException!`**
- calling `.nextInt()` on `null`? **`NullPointerException!`**
- many, many more

PCM Review: avoiding NullPointerException

```
if (strs[i] != null) {
  System.out.println(strs[i].toUpperCase());
} else {
  System.out.println("element " + i + " is null.");
}
```