

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 25au Lecture Tunes](#) 🎵

Instructors: Brett Wortzman & James Weichert

TAs:	Trey	Ava	Caleb	Elden	Anya
	Amogh	Reese	Anum	Suyash	Minh
	Samrutha	Hayden	Abdul	Sthiti	TJ
	Dalton	Aki	Janvi	Paul	Zach
	Ailsa	Spencer	Navya	Shayna	Cayden
	Ryan	Savannah	Sam	Jesse	Johnathan
	Anant	Tamsyn	Jessica	Nhan	

Announcements, Reminders

- C3 released tonight, due **Tuesday, November 18th**
- R4 due tomorrow (eligible: C1, P1, C2)
 - C1 cycling out of eligibility after R4
- Quiz 2 on **Thursday, November 20th**
 - Conditionals, while loops, User Input (Scanner), Arrays, Reference Semantics
 - can't make it? email Brett and James before your quiz!
- In the future: Thanksgiving week November 24-28
 - Almost all class activities cancelled

Feedback & Closing the Loop: The Good



- Pre-class materials and walkthrough videos
- Quiz sections and your TAs! (esp. practice problems)
- Project-based assignments
 - Including feedback!
- In-class active learning!
- Ed message board and IPL!

Feedback & Closing the Loop: Suggestions



- Section and exit ticket solutions
 - Already available!
- One place to view all grades
 - In progress; maybe this week? 🙌
- More paper practice in section
 - On it's way!

Other suggestions...

- More and less instruction in section
- More practice quizzes

Feedback & Closing the Loop: Transfer



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

Tricky to balance **alignment** (lessons, practice, assignments connecting to each other) and **transfer** (assessing your ability to apply skills to novel problems).

But *some* transfer is always necessary.



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;  
        }  
    }  
}
```

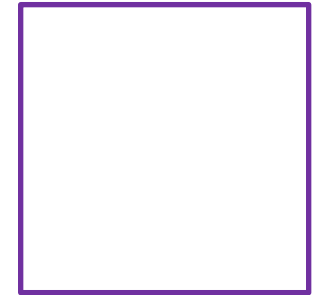
- A. `{-20, 20, 26, 32, 50, 3}`
- B. `{-15, 25, 31, 37, 55, 8}`
- C. `{-15, 25, 31, 37, 50, 3}`
- D. `{-15, 20, 26, 37, 50, 3}`

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;  
        }  
    }  
}
```

i



a

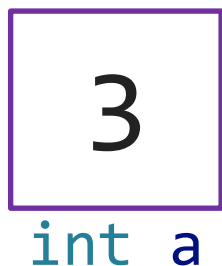


PCM Review: Value Semantics vs. Reference Semantics

Value Semantics

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

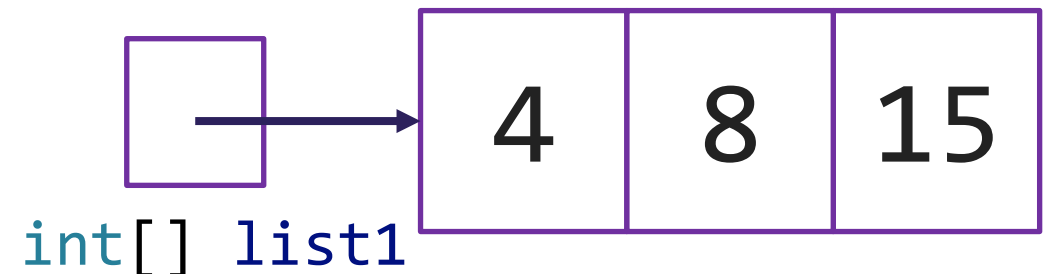
```
int a = 3;
```



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;
```

int a

99

int b

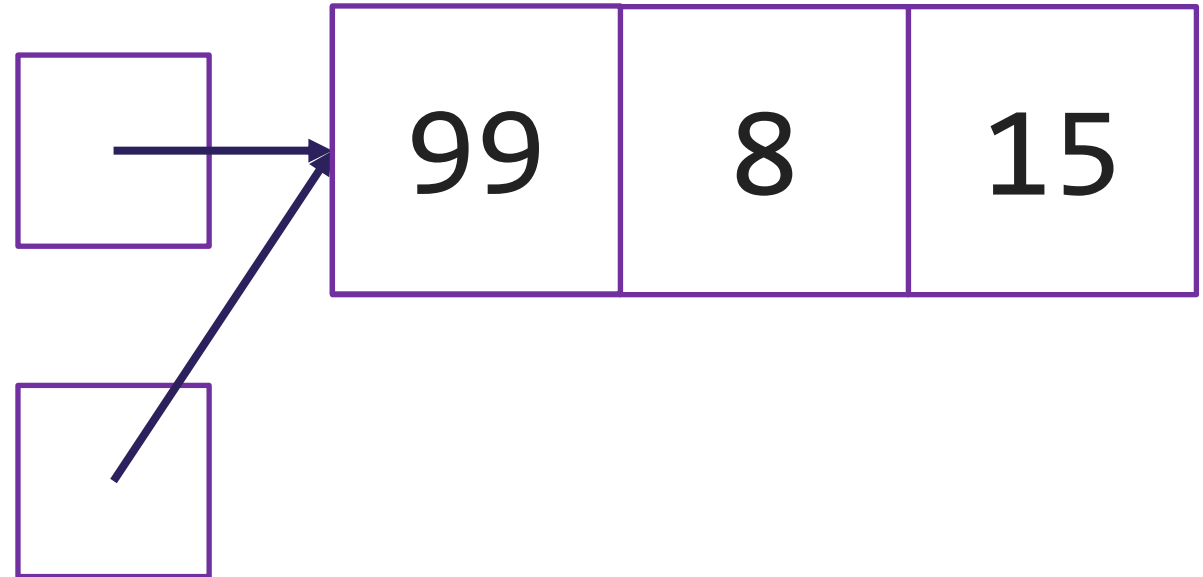
3

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;
```

int[] list1



int[] list2

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.");  
}
```