

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?*

Instructors: Hayden Feeney**TAs:** Anisha Savannah William
Cayden Tamsyn

Announcements, Reminders

- C3 releases today, due **Thursday, August 13th**
- R4 due next **Thursday, August 13th** (eligible: **C1**, P1, C2, P2)
 - C1 cycling out of eligibility after R4
- Quiz 2 on **Thursday, August 13th**
 - Conditionals, while loops, User Input (Scanner), Arrays
 - can't make it? email Hayden before your quiz!
- Final exam on **Friday, August 21st in class!**
 - Details on [final exam page](#)
 - More details to come!



Practice: Think

sli.do [#cse121](https://twitter.com/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;  
        }  
    }  
}
```

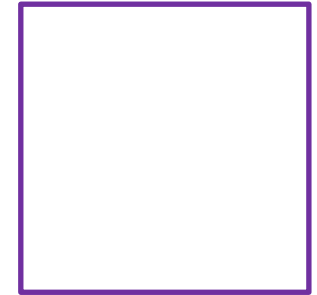
- A. `{-20, 20, 26, 32, 50, 3}`
- B. `{-15, 25, 31, 37, 55, 8}`
- C. `{-15, 25, 31, 37, 50, 3}`
- D. `{-15, 20, 26, 37, 50, 3}`

Tracing through arrayMystery

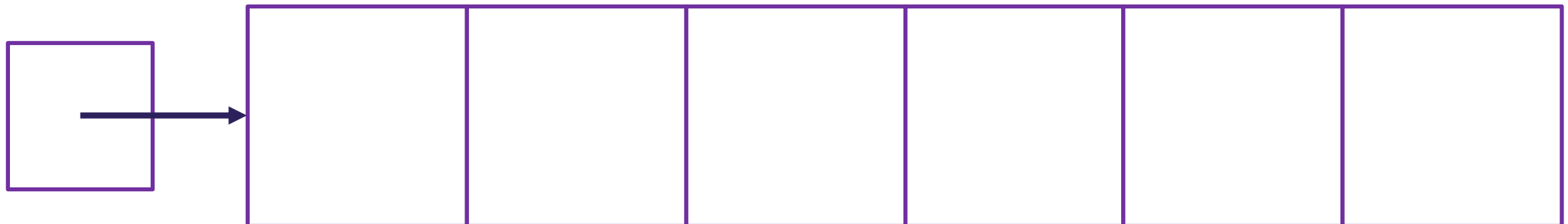
{-20, 20, 26, 32, 50, 3}

```
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

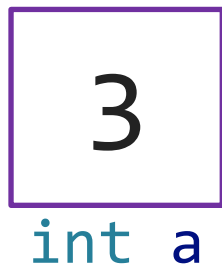


PCM Review: Value Semantics vs. Reference Semantics

Value Semantics

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

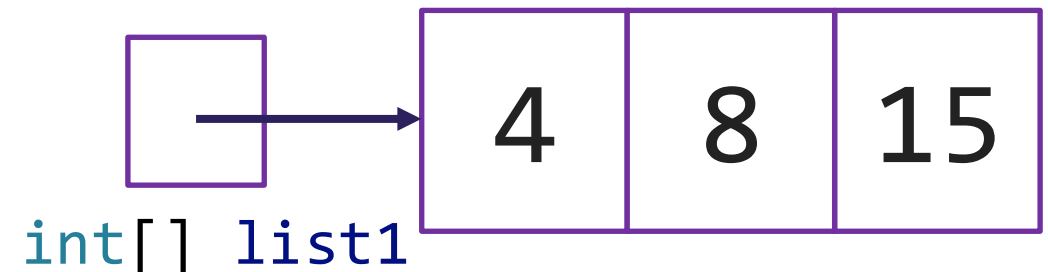
```
int a = 3;
```



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

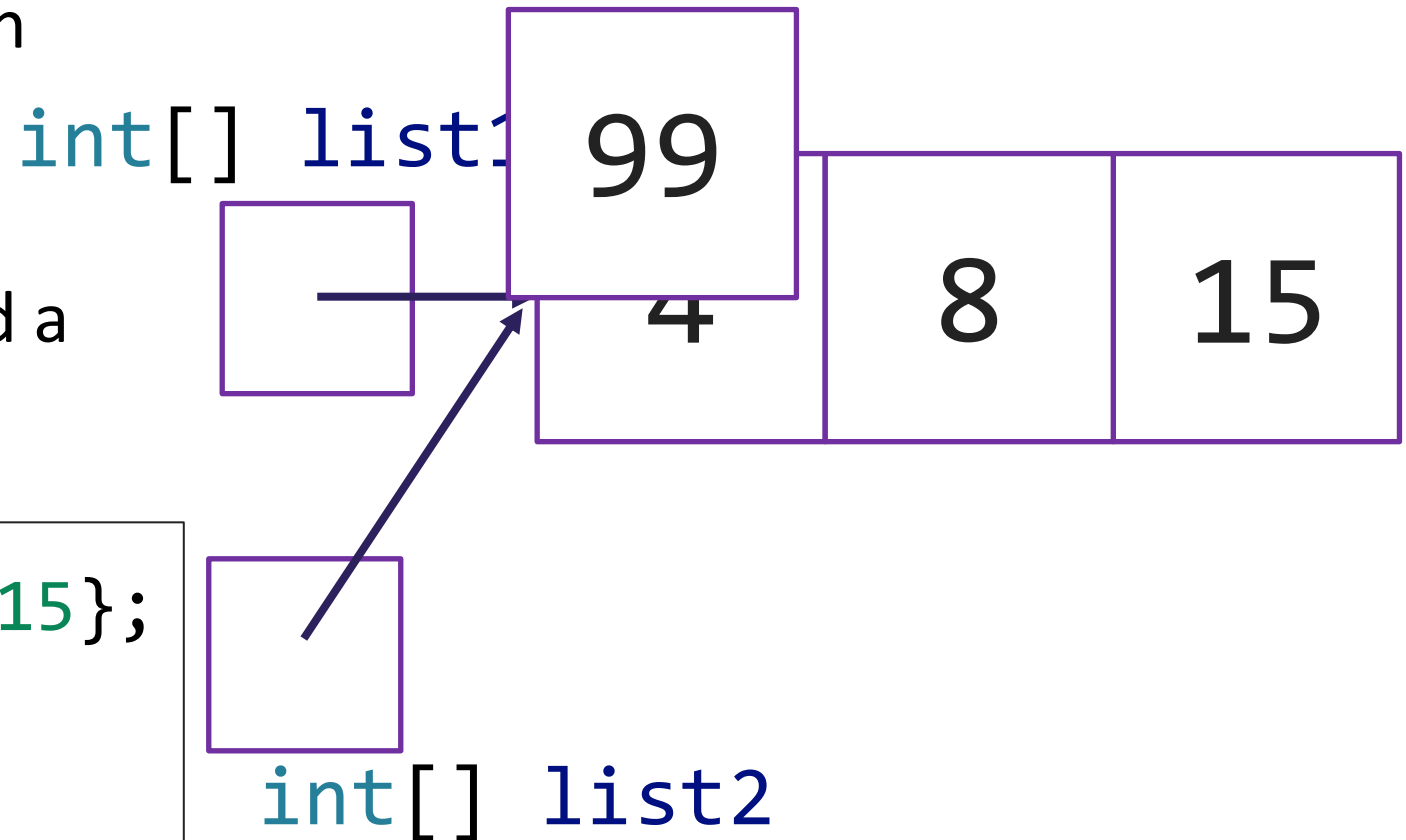
int b

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;
```



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.");  
}
```