LEC 14

CSE 121

Reference Semantics

Questions during Class?

Raise hand or send here

sli.do #cse121

BEFORE WE START

Talk to your neighbours:

What do you do to destress?

Music: 121 25wi lecture playlist

Instructor: Matt Wang

TAs: Ailsa, Alice, Chloë, Christopher, Ethan, Hanna, Hannah, Hibbah, Janvi, Judy, Julia, Kelsey, Lucas, Luke, Maitreyi, Merav, Ruslana, Samrutha, Sam, Shayna, Sushma, Vivian

Announcements, Reminders

- C3 released tonight, due **Tuesday, Mar 4th**
- R4 due tomorrow (eligible: **C1**, P1, C2)
- Quiz 2 on **Thursday, Mar 6th**
  - includes everything up to today's lecture
  - can't make it? email me before your quiz!
- Announcement explaining extra resub (from PSWs) is out!
- In the future: final exam on **Tue, March 18th from 12:30-2:20 PM**
  - many, many more details coming after Quiz 2

A bit on structured group work in section

- thanks for your continued feedback & thoughts!
  - from poll: entire class >> yourself/pairs/small group
  - “People tend to not talk to each other or answer questions in quiz section. I'm not sure [whether] it's because they fear being wrong or they're just not interested. Maybe putting us in pairs/small groups could take off a bit of the pressure.”
- after much discussion and work, expect to see…
  - more paper worksheets (building mental models + finals prep)
  - more opt-in pair/small-group section activities
  - section problems with a tighter/timeboxed scope

Practice: Think

sli.do #cse121

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

```
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;
    }
  }
}
```

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

```
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;
    }
  }
}
```

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

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

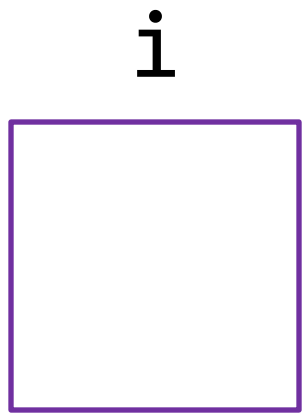
C. `{-15, 25, 31, 37, 50, 3}`

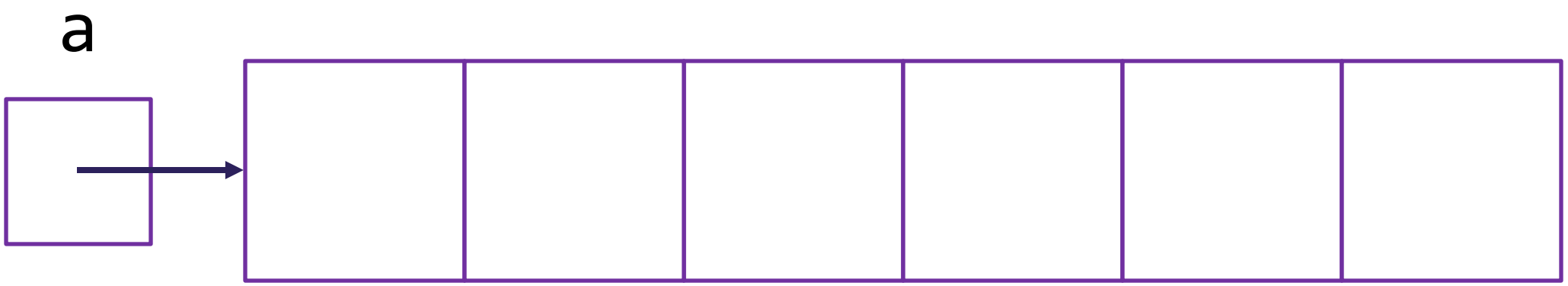
D. `{-15, 20, 26, 37, 50, 3}`

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;
    }
  }
}
```



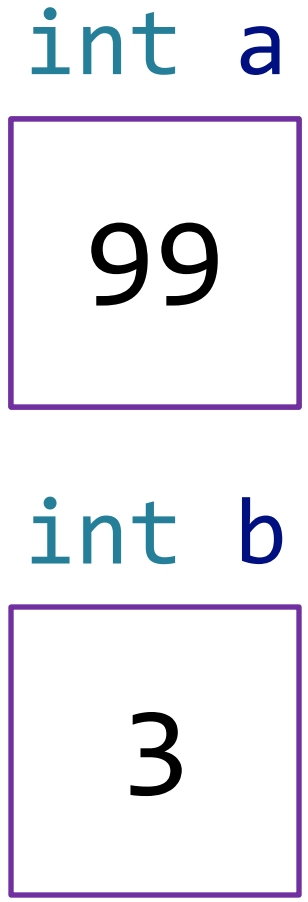


PCM Review: Value Semantics

Our "default" model for variables:

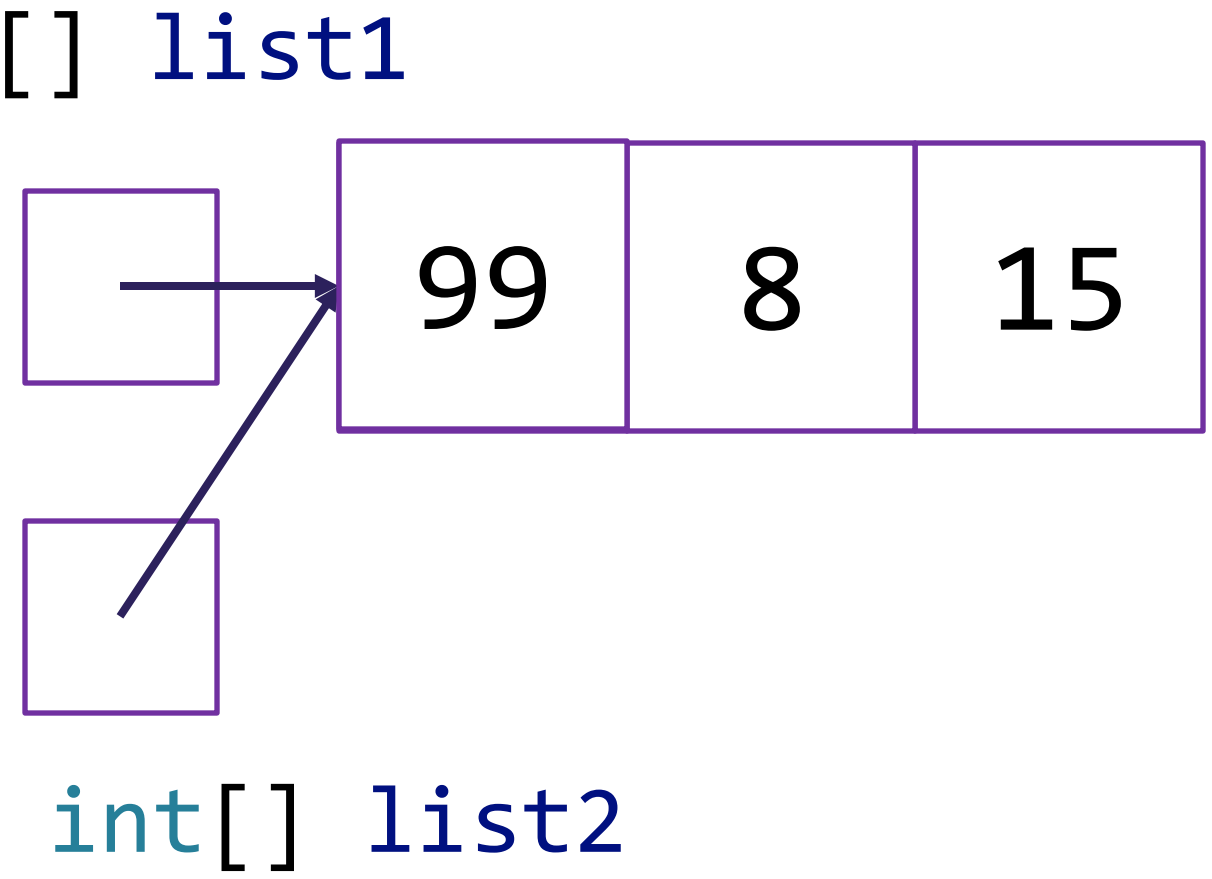
- applies to primitive types
- variables/parameters hold a *copy* of the actual value

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



PCM Review: Reference Semantics

- applies to object types
  - importantly, arrays!
- variables/parameters hold a *reference* to the object




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

Practice: Think

sli.do #cse121

Without knowing what `someMethod` does, what are the possible values of `num`?

```
int num = 42;
someMethod(num);
System.out.println(num);
```

A. anything!

B. just 42

Practice: Think

sli.do #cse121

Without knowing what `anotherMethod` does, what are the possible values of `nums[0]`?

```
int[] nums = {42, 43, 44};
anotherMethod(nums);
System.out.println(nums[0]);
```

A. anything!

B. just 42

Practice: Pair

sli.do #cse121

Without knowing what `someMethod` or `anotherMethod` do, what are the possible values of `num` and `num[0]`?

```
int num = 42;
someMethod(num);
System.out.println(num);
```

```
int[] nums = {42, 43, 44};
anotherMethod(nums);
System.out.println(nums[0]);
```

A. num: anything
   num[0]: anything

B. num: 42
   num[0]: anything

C. num: anything
   num[0]: 42

D. num: 42
   num[0]: 42

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

Aside: the "billion dollar mistake"

From Sir Tony Hoare ("inventor" of `null`, Turing award winner):

"I call it my billion-dollar mistake… […]

But I couldn't resist the temptation to put in a null reference, simply because it was so easy to implement. This has led to innumerable errors, vulnerabilities, and system crashes, which have probably caused a billion dollars of pain and damage in the last forty years." (quote from 2009 talk)

PCM Review: avoiding NullPointerException

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

Announcements, Reminders (again)

- C3 released tonight, due **Tuesday, Mar 4th**
- R4 due tomorrow (eligible: **C1**, P1, C2)
- Quiz 2 on **Thursday, Mar 6th**
  - includes everything up to today's lecture
  - can't make it? email me before your quiz!
- Announcement explaining extra resub (from PSWs) is out!
- In the future: final exam on **Tue, March 18th from 12:30-2:20 PM**
  - many, many more details coming after Quiz 2