CSE 121 Lesson 14: More Arrays and Reference Semantics

Matt Wang & Brett Wortzman

Autumn 2024

sli.do #cse121

TAs:

Abby	Afifah	Ailsa	Alice	Aliyan	Arohan
Chloë	Christopher	Dalton	Derek	Elizabeth	Ethan
Hanna	Hannah	Heather	Hibbah	Janvi	Jasmine
Judy	Julia	Kelsey	Lucas	Luke	Mahima
Maitreyi	Maria	Merav	Minh	Neha	Ronald
Ruslana	Sahej	Sam	Samrutha	Sushma	Vivian
Yijia	Zachary				

Today's playlist:
121 24au lecture tunes

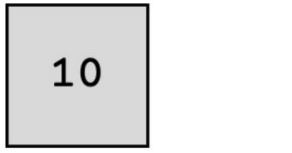
Reminders & Announcements

- C3 released tonight, due Tuesday, November 19th
- R4 closes tomorrow (last chance for C1)
- Quiz 2 next Thursday, November 21st
  - topics: everything up until arrays (incl. today's material)
  - Practice quiz out later this week
- Bonus resub for PSW coming later this week
- In the future: Final Exam (Wednesday, December 11th from 12:30 – 2:20 PM)
  - more logistical details coming soon!

(PCM) Value Semantics vs. Reference Semantics

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

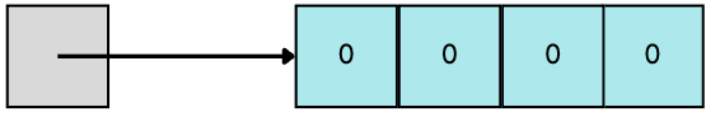
```
age = 10;
```



name: age (int)

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

```
int[] arr = new int[4];
```



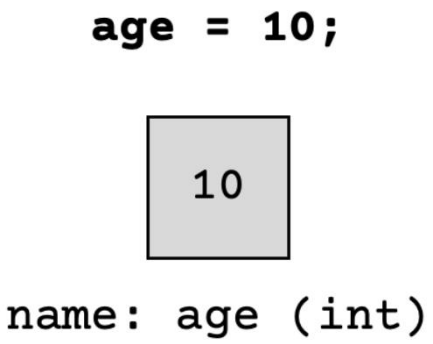
name: arr (int[])

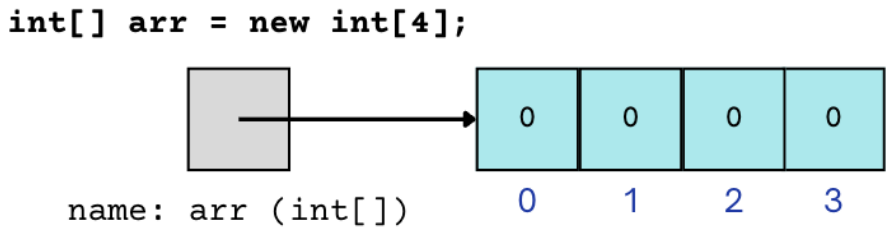
0 1 2 3

(PCM) Value Semantics vs. Reference Semantics

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

```
int[] list1 = {4, 8, 15, 16, 23};
int[] list2 = list1;
list1[1] = 99;
```





Poll in with your answer!

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

Poll in with your answer!

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

(PCM) Value Semantics vs. Reference Semantics

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

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

(PCM) `null`

The *absence* of a reference!

- Sort of like a "zero-equivalent" for references!
- Default value for “object types” (e.g. Random, Scanner…)

`NullPointerException``s` are an error that happen when you ask `null` to “do something”

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

`null`: 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) avoiding NullPointerException

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