CSE 121 Lesson 15:
Arrays and Reference Semantics

Matt Wang
Spring 2024

TAs: Andy, Anju, Archit, Arkita, Autumn, Christian,
Hannah H, Hannah S, Heather, Hibbah, Janvi, Jessie,
Jonus, Julia, Luke, Maria, Mia, Ritesh,
Shayna, Simon, Trey, Vidhi, Vivian, Gumball?

sli.do #cse121-15

Today's playlist: CSE 121 lecture beats 24sp
Reminders & Announcements

• C3 released Wednesday, due Tuesday May 21
• R5 released yesterday, due Thursday May 23rd (last chance for P1)
• Quiz 2 next Thursday, May 23rd
  • topics: everything up until Arrays on Wednesday (i.e. not today’s material)
  • see: Ed post on practice resources!
• In the future: Final Exam (Wednesday June 5th from 2:30 – 4:20 PM)
  • more logistical details coming soon!
Poll in with your answer!

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\}
(PCM) Value Semantics vs. Reference Semantics

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

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

```java
age = 10;
name: age (int)
```

```java
int[] arr = new int[4];
name: arr (int[])
```
Value Semantics vs. Reference Semantics

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

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

####缺点
- 使用值传递效率更高
- 适用于变量和小值
- 简化代码

####优点
- 使用引用传递效率更低
- 适用于大值和数组
- 适用于变量和小值的结合
Without knowing what `someMethod` does, what are the possible values of `num`?

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

A. anything!
B. just 42
Poll in with your answer!

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

```java
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];
    }
}
null

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

NullPointerExceptions 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

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