Discussion 1
CS2112

August 30th / August 31st, 2022

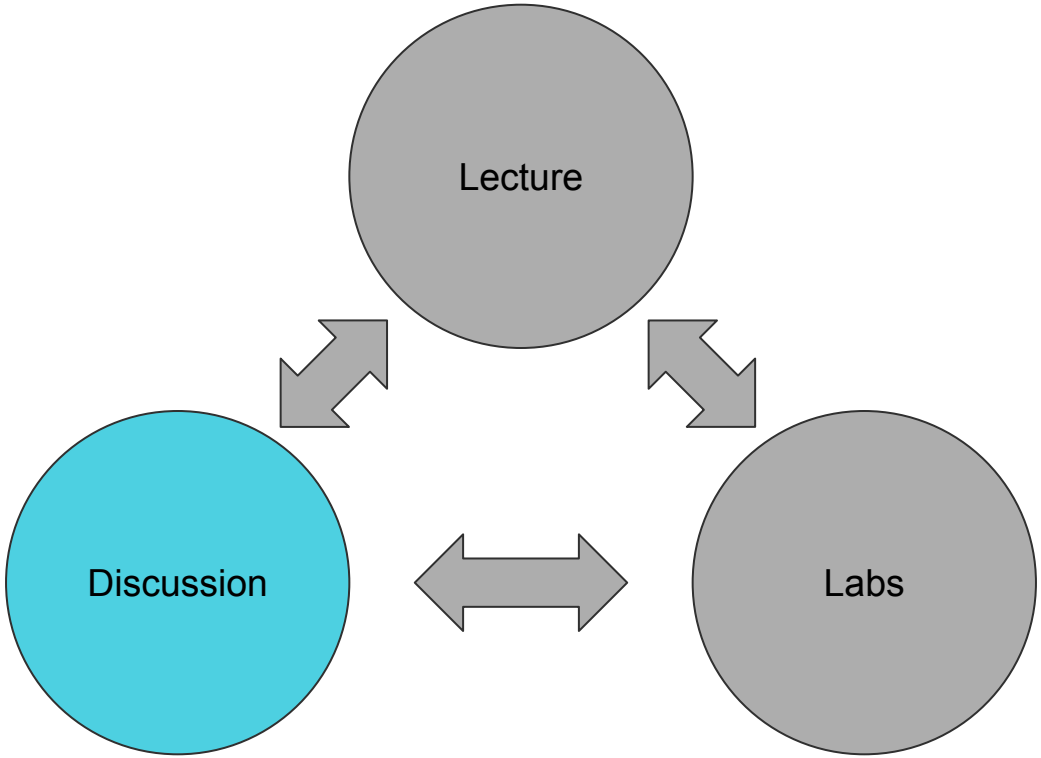
Discussion Staff

Tuesday	Wednesday
Jessica Cho	Vivian Ding
Kushal Kedia	Shiyuan Huang
Jerry Xu	Esther Wang

Reminders

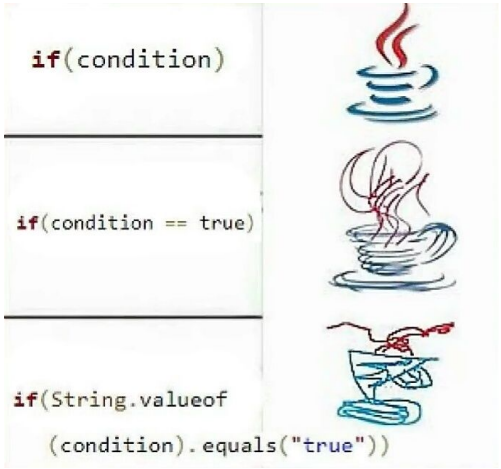
- A1 Out! Due September 8th, 11:59PM
- Class resources
  - Course website - https://www.cs.cornell.edu/courses/cs2112/2022fa/
  - Ed
  - CMSX
  - Add deadline coming up soon!
- Labs
  - Monday/Wednesday

What is discussion?



Java Execution Model

Arrays, Strings, Autoboxing



Names and Scope

Kind of name	Scope
Local variables	From when declared to end of block
Method names, field names	Class
Class and interface names	Program
Nested class	Containing class

General rule: find the definition of the name with the smallest scope that includes the use of the name

More on Scope

If a name is in the scope of two different declarations at once, the outer declaration is said to be **shadowed** by the inner one

Shadowing is sometimes illegal. Example:

```
int x = 2;
while (x != 0) {
    int x = 5;
    // both x's in scope here.
}
// only outer x in scope here.
```

More on Scope

But it's also definitely legal in other places.

Commonly shows up in constructors:

```
class Point {
    int x, y;
    Point(int x, int y) {
        // locals x and y shadow instance variables x and y
        this.x = x;
        this.y = y;
    }
}
```

This shadowing only occurred because we named the constructor parameters the same thing as the instance variables

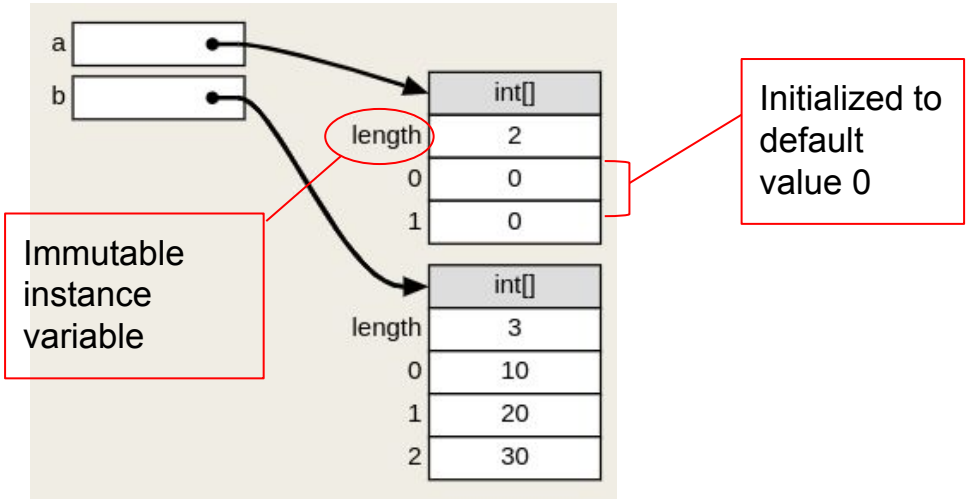
Keyword **this** can be used within instance methods to refer to instance variables (not static ones)

Arrays

- Arrays are reference types in Java

```
int[] a = new int[2];
int[] b = new int[] { 10, 20, 30 };
```

- Variables `a` and `b` contain references to arrays rather than the arrays themselves

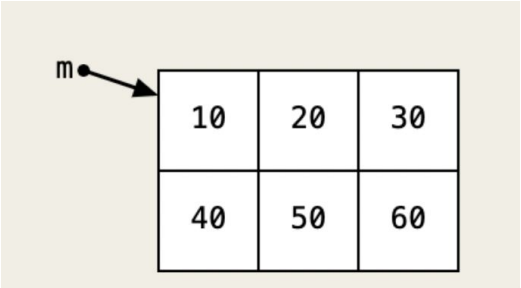


Multidimensional Arrays

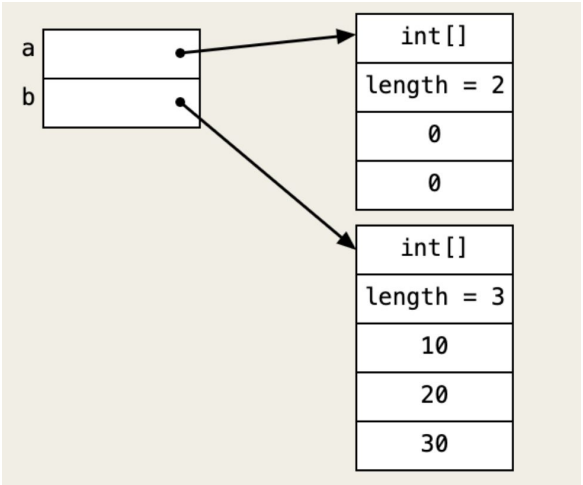
Arrays within an array! For example, a 2-D array consists of 1-D arrays

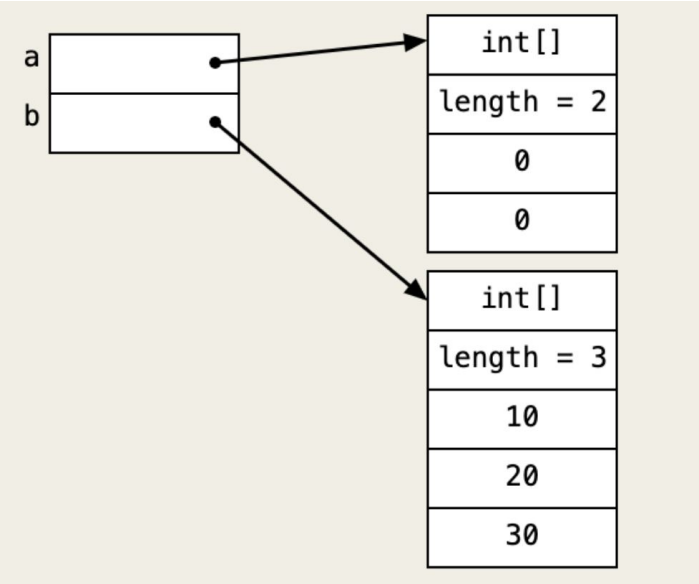
```
int[][] m = {{10, 20, 30}, {40, 50, 60}};
```

Can be thought of as a matrix….



…but internally represented by 3 objects





The rows can alias each other, i.e, represent the same object

```
m[1] = m[0];
```

Also, Java does not enforce that the elements in an array have the same dimension

For example, consider the 2 lines of code:

```
int[][] m = {{10, 20, 30}, {40, 50, 60}};
```

```
m[1] = new int[1];
```

This means that the index 1 row of the matrix m has a dimension of 1, whereas the index 0 row has a dimension of 3

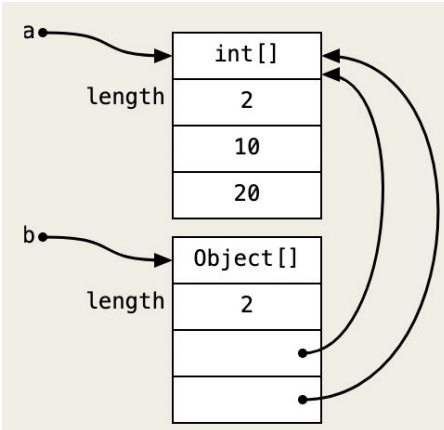
Arrays as Objects

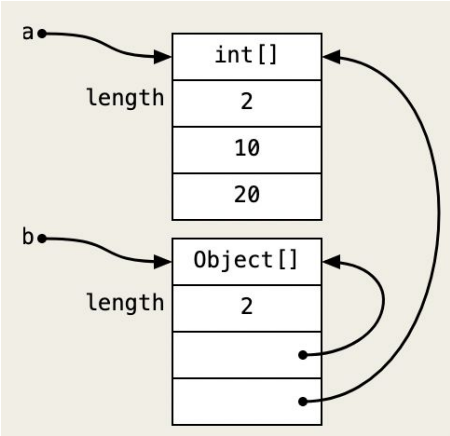
Arrays are objects, so the following is legal:

```
Object a = new int[] {10, 20};
```

Is the following legal?

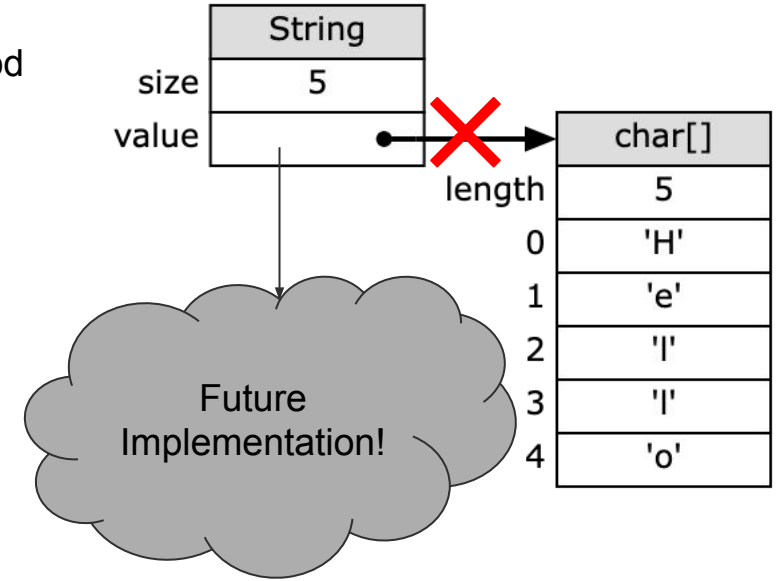
```
Object[] b = new Object[] {a, a};
b[0] = b; // !
```





Strings

- Objects!
  - Implemented as character arrays under the hood
- Useful methods
  - `==` operator vs `.equals()` method
  - `.charAt()`
- Abstracting implementation away!



Autoboxing

primitive type	corresponding reference type
int	Integer
boolean	Boolean
short	Short
byte	Byte
char	Char
float	Float
double	Double
long	Long

Idea. Object can refer to any object, but primitive values are not objects. Sometimes, we expect a reference type somewhere but want to store an integer.

Solution. autoboxing!

Example. Use HashMap<Integer, Boolean> rather than HashMap<int, boolean>

More on Autoboxing

```
Integer i = 200;
Object l = i;
int j = i;
Object k = j;
System.out.println(i == j); // true
System.out.println(i == l); // true
System.out.println(j == k); // static error: can't compare Object and int.
System.out.println(l == k); // false!
System.out.println(l.equals(k)); // true
```

Static type of k is an Object, whereas j is an int. These cannot be directly compared.

l and k represent the same number, but they are different objects. This comparison says l and k are stored at different memory locations. Use .equals() to compare the value!

Questions?