int x = 0;
x.toString()

PRMITIVE DATA TYPES (CH02)

- not objects
- numeric (integers)
  - `byte`, `short`, `int`, `long`
- floating-point (numbers with decimal points)
  - `float`, `double`
- T/F: `boolean`
- characters: `char` (single quotes)

10.2 0.1

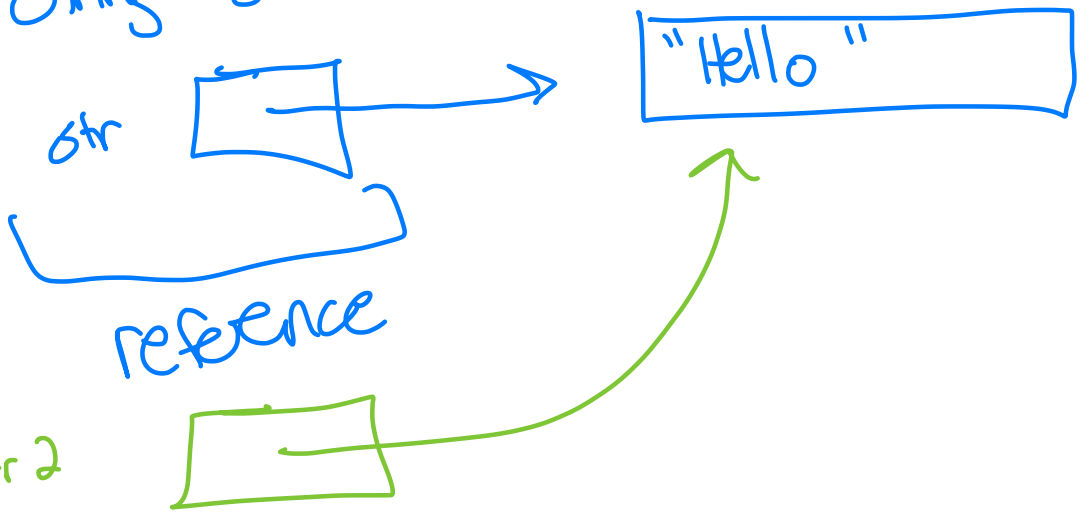
True False

NON-PRIMITIVE DATA TYPES

- objects
- can be objects from user defined classes or built-in
- Built-in
  - String `String str = "Hello";`
  - Wrapper for each primitive
  - Ex: `Integer`, `Boolean`, `Character`, `Double`
- variable is actually a reference

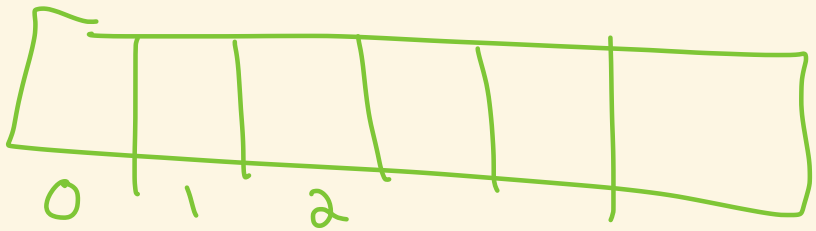
```
int x = 5;
```


```
String str = "Hello";
```

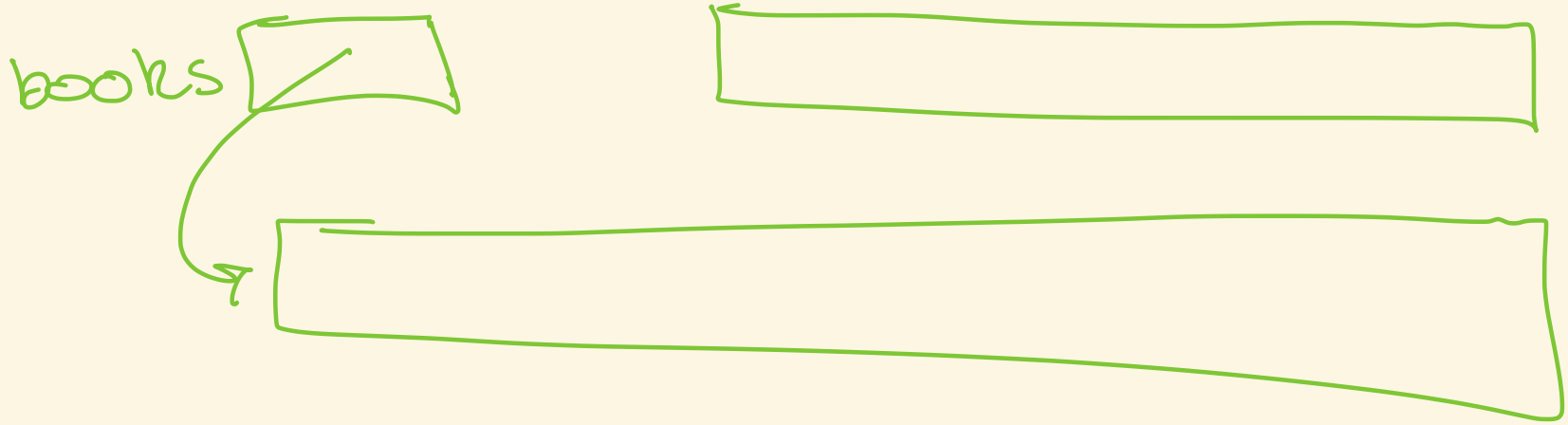



```
String str2 = str
```

ARRAYS



- List of values (ordered)
- Index: number corresponding to position
- Can store:
  - Primitive types
  - Objects
- Every element has to be the same type
- Fixed-size (size cannot change)



CREATING ARRAYS

```
int[] alphabet = new int[26];
double[] nums = new double[10];
Book[] books = new Book[100];
```

- the `new` instantiates an array object

books = new Book[200];

ARRAY INSTATIATION

- Makes space to hold either primitive types or reference to object
- Internal objects not created

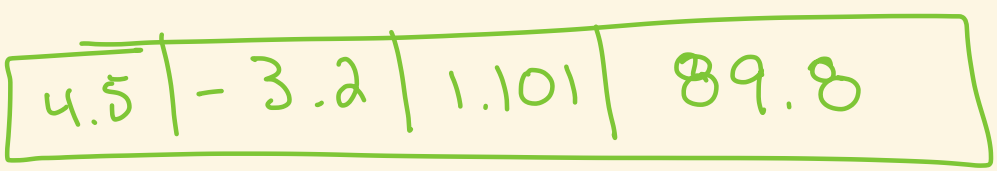
ARRAY INITIALIZATION

- Need to fill array
- Default depends on type
  - numerical
  - object
  - character
  - boolean

ARRAY INITIALIZATION

- Initializer list
  - don't use `new` or specify `size`
  - determined by number of items in initializer list

```
double[] nums = {4.5, -3.2, 1.101, 89.8};
```



ARRAYS

- Accessing elements `nums[1]`
- 0-based indexing
- Bounds checking:
  - Java will throw exception if index out of bounds
  - Practically, logic should prevent this from happening

TESTING

- Idea: should think about how you'll know when your code is correct
- really the thinking should happen *before* you code
- think about the different scenarios that can come up
- think about possible inputs and outputs -- what results should you expect

UNIT TESTING

- Testing each piece of small piece of code individually
- Means you can easily see if you've broken anything
- Not just about 1 unit test per method, about making sure that you fully test your method
- Edge cases: things that only come up some of the time
    - what happens if something is empty?
    - going out of bounds?
    - ...?

Book

BookTest

JUNIT

- A library for unit testing in java
- Create a class for testing (may have more than one depending on scope of project)
- Add a decorator `@Test` before each method that is a test

@Test (timeout =

ASSERTS

- tests typically return void
- asserts are ways of saying something has to be true (if not the test will fail)