ARRAYS

- List of values (ordered)
- Index: number corresponding to position
- Can store:
  - Primitive types
  - Objects
- 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

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};
```

ARRAY INITIALIZATION

- Makes space to hold references
- Internal objects not created

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

MULTIDIMENSIONAL ARRAYS

- An array of arrays
- Ex: `int[][] arr = new int[5][3]`

MULTIDIMENSIONAL ARRAYS

- Like 1D arrays can hold primitives, objects, etc.
- Purposes:
  - Representing grids (ex: game boards)
  - Representing tables/matrices
    - Physical simulations
    - ML/DM (representing samples and features)
    - images

MULTIDIMENSIONAL ARRAYS

- Not limited to 2D
- Can have 3D, 4D, etc.
- Ex: 3D array is just an array of 2D arrays
- Uses:
  - movie = sequence of images
  - color image can be thought of as N x M x 3 array

JAGGED ARRAYS

- `int[][] arr = new int[5][3];`
  - array of 5 arrays of size 3
- could create N-D array differently

```
int[][] arr = new int[2][];
```

- need to then create each internal array separately