Arrays

John Nash of “A Beautiful Mind” sent a series of letters to the NSA in the 1950s detailing an encryption system of his own design. The design was rejected, but his theoretical discussion of secure cryptography is very similar to the theory employed by computer scientists today. The letters were recently declassified: January 27, 2012.

The correspondence is available online:

http://courses.csail.mit.edu/6.857/2012/files/H03-Cryptosystem-proposed-by-Nash.pdf

Friday Four Square!

Today at 4:15PM, outside Gates.

A Different Way to Store Data

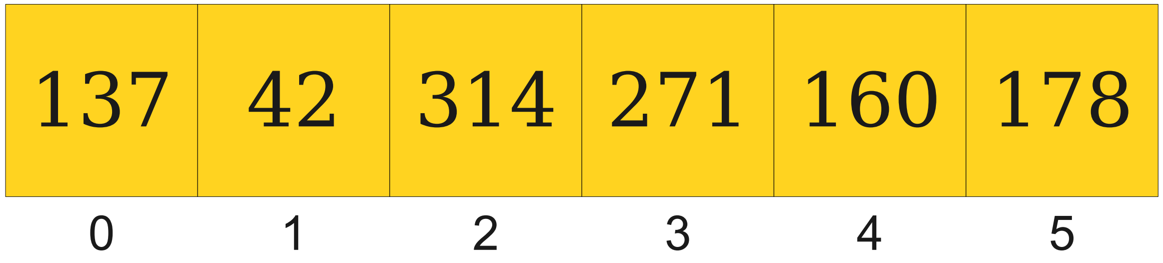
- Last time, we saw the **`ArrayList`** as a way to store lots of data.
  - Lines of text.
  - US cities!
- There is another Java concept called the **array** that can also be used to store data.

Recapping **`ArrayList`**

137	42	314	271	160	178
0	1	2	3	4	5

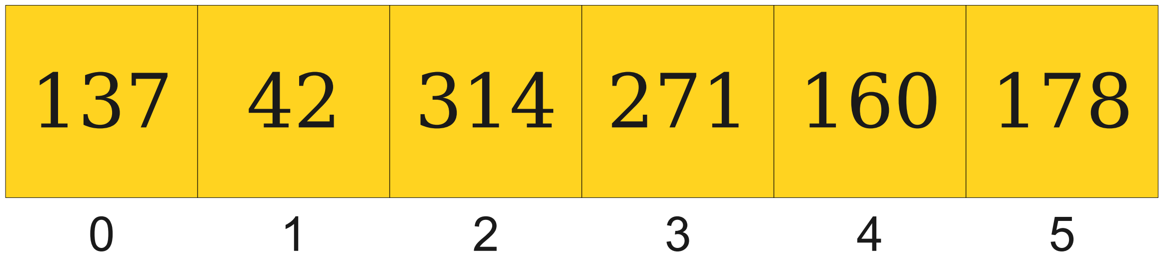
- An **`ArrayList`** stores a **sequence** of multiple objects.
  - Can access objects by index by calling **`get`**.
- All stored objects have the same type.
  - You get to choose the type!
- Must store objects; primitive types not allowed.
- Can grow as long as it needs.

Introducing Arrays



- An array stores a **sequence** of multiple objects.
  - Can access objects by index using [].
- All stored objects have the same type.
  - You get to choose the type!
- Can store *any* type, even primitive types.

Introducing Arrays



- An array stores a **sequence** of multiple objects.
  - Can access objects by index using [].
- All stored objects have the same type.
  - You get to choose the type!
- Can store *any* type, even primitive types.
- Size is fixed; cannot grow once created.

Default Values in Arrays

- When creating an array:
  - `int`, `double`, `char`, etc. default to 0,
  - `boolean` defaults to `false`, and
  - Objects default to `null`.

Basic Array Operations

- To create a new array, specify the type of the array and the size in the call to **`new`**:

```
Type[] arr = new Type[size]
```

- To access an element of the array, use the square brackets to choose the index:

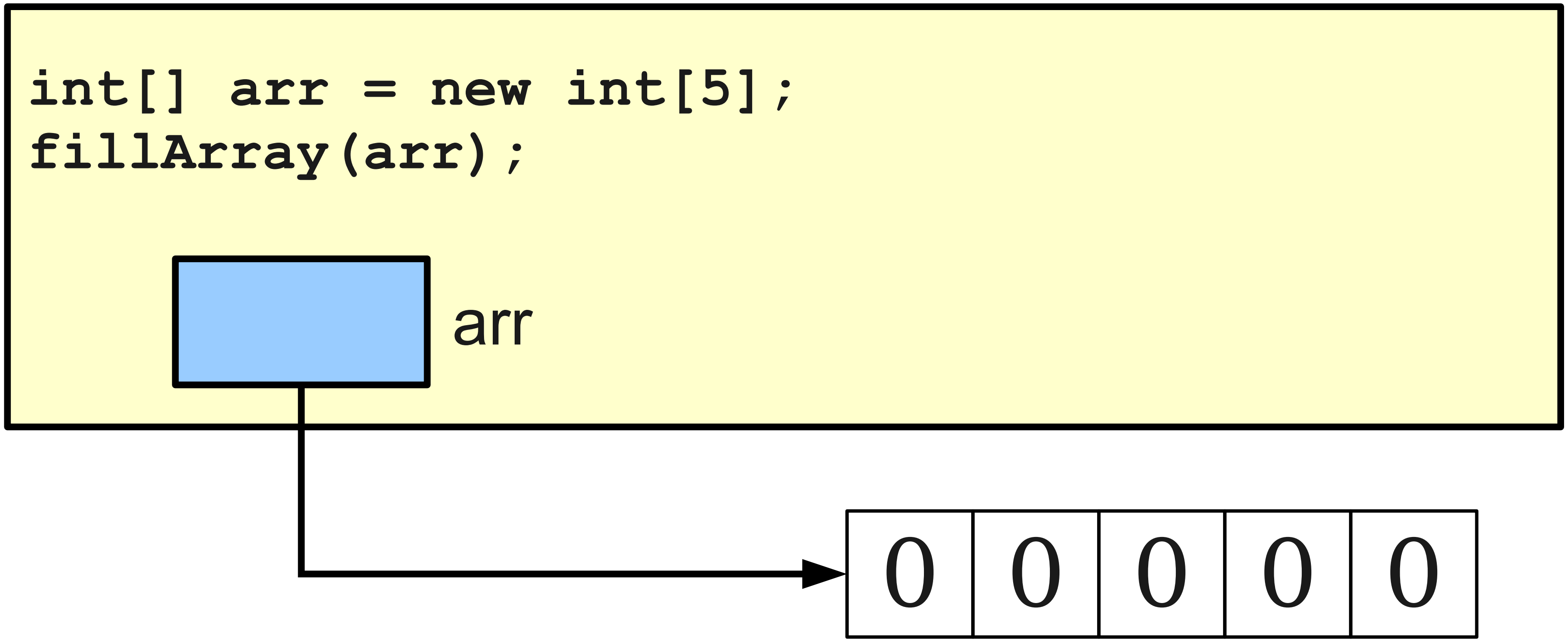
```
arr[index]
```

- To read the length of an array, you can read the **`length`** field:

```
arr.length
```

A Nuance of Pass-by-Reference

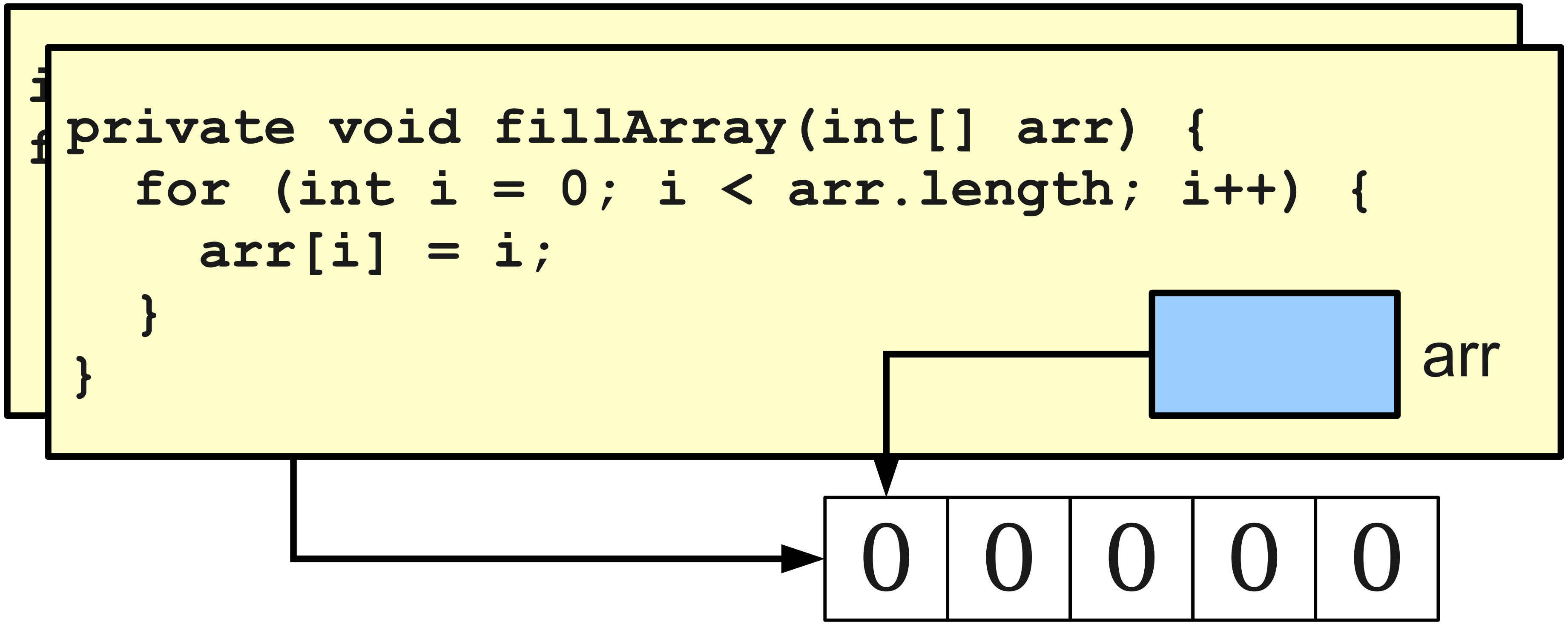
- Arrays are objects, so they are passed by reference.
- The **elements** of an array, like the fields of an object, can be modified inside of a method.



```
int[] arr = new int[5];
fillArray(arr);
```

A Nuance of Pass-by-Reference

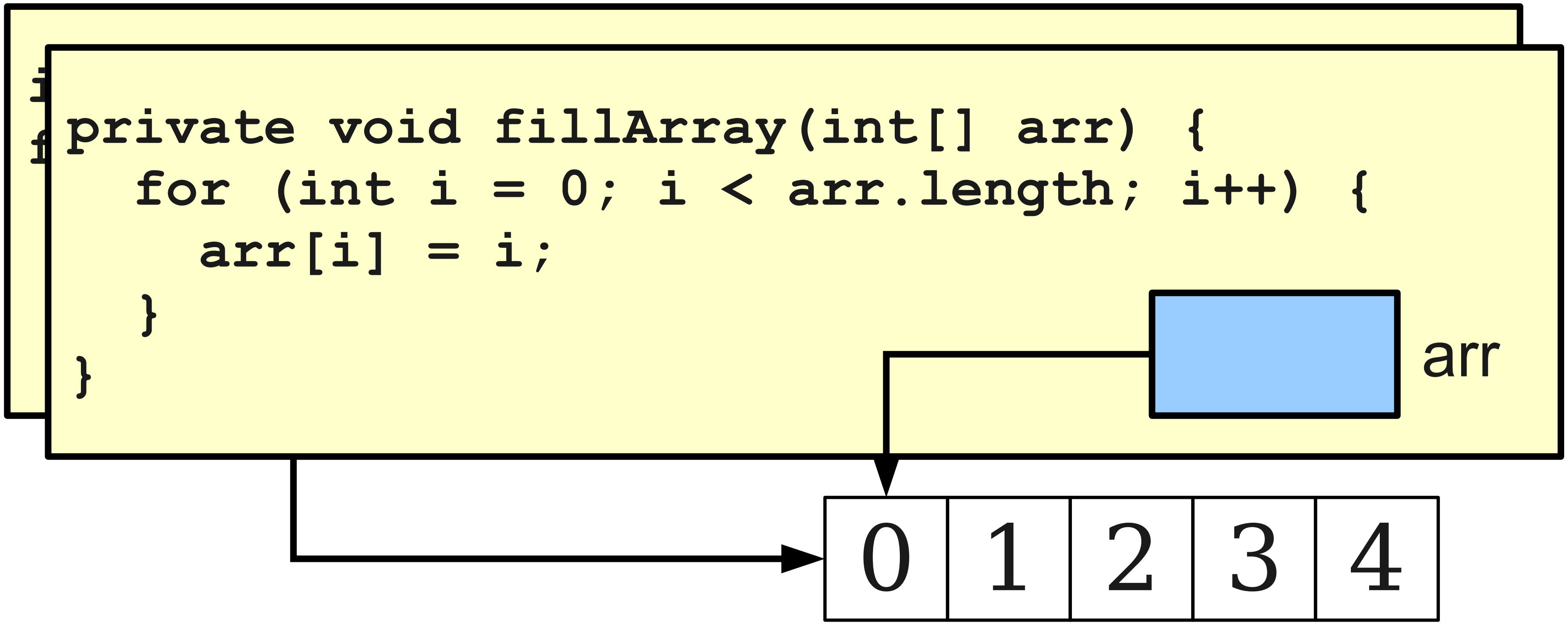
- Arrays are objects, so they are passed by reference.
- The **elements** of an array, like the fields of an object, can be modified inside of a method.



```
private void fillArray(int[] arr) {
  for (int i = 0; i < arr.length; i++) {
    arr[i] = i;
  }
}
```

A Nuance of Pass-by-Reference

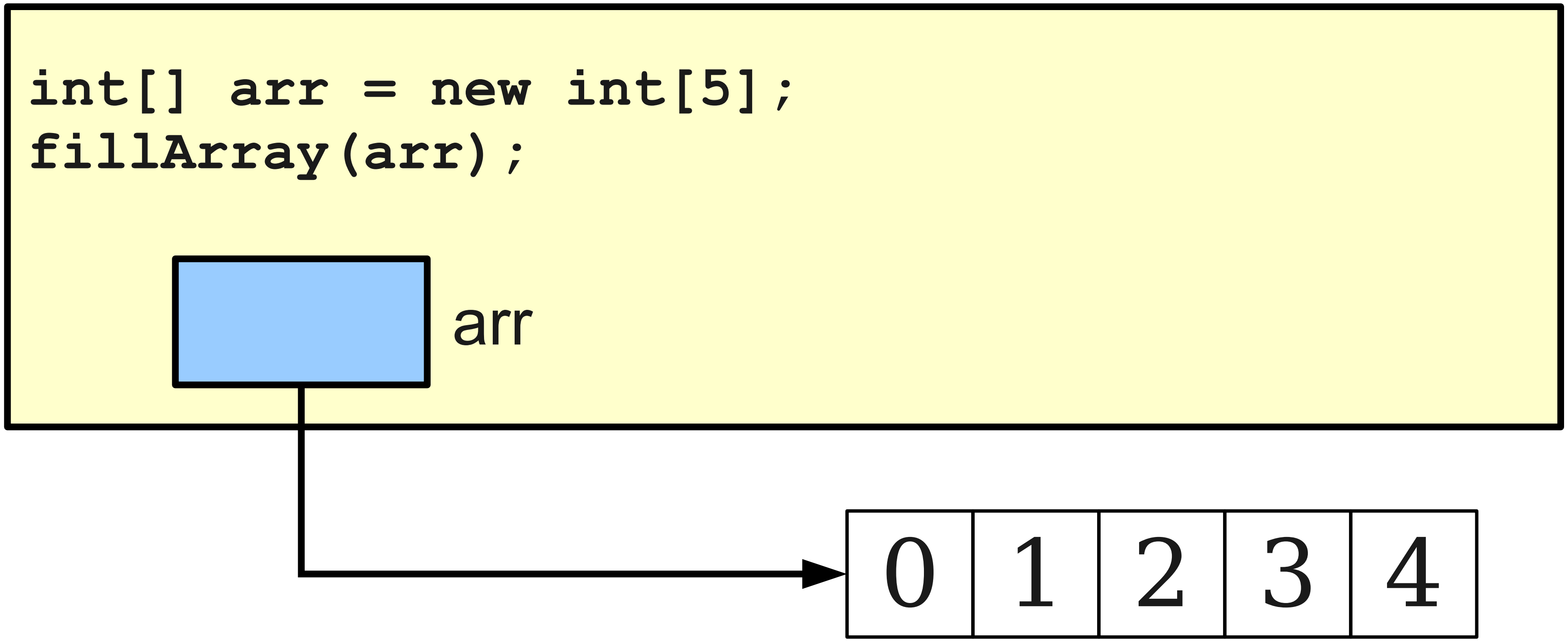
- Arrays are objects, so they are passed by reference.
- The **elements** of an array, like the fields of an object, can be modified inside of a method.



```
private void fillArray(int[] arr) {
   for (int i = 0; i < arr.length; i++) {
      arr[i] = i;
   }
}
```

A Nuance of Pass-by-Reference

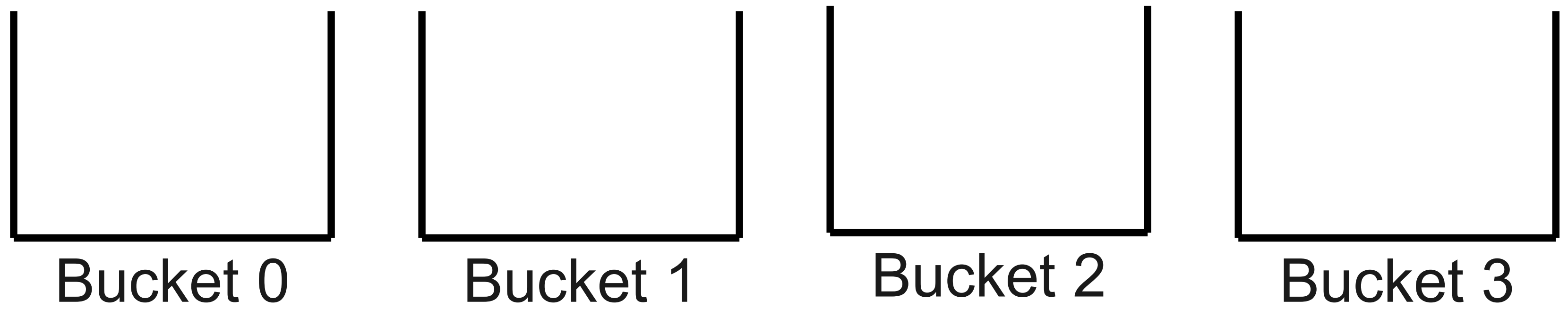
- Arrays are objects, so they are passed by reference.
- The **elements** of an array, like the fields of an object, can be modified inside of a method.



```
int[] arr = new int[5];
fillArray(arr);
```

Why Arrays?

- Arrays are excellent for representing a fixed-size list of **buckets**.
- We can store values in the appropriate bucket by looking up the bucket by index.

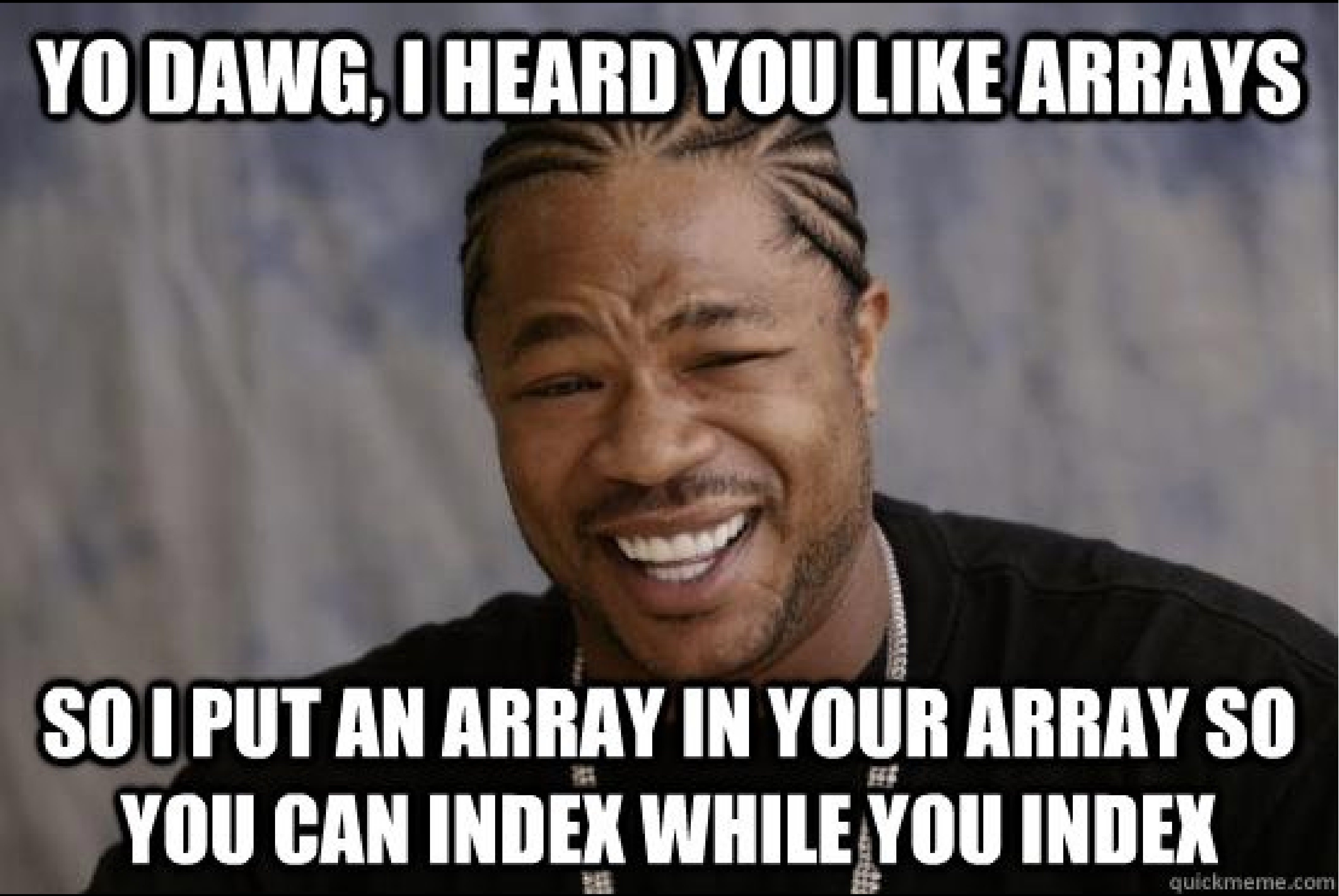


How many people need to be in a room before two of them will share a birthday?

The Birthday Paradox

- In a room of 23 people, there is a 50% chance that two of them have the same birthday.
- More generally, if you have an *n*-sided die, you only need to roll it around $\sqrt{2n}$ times before you have a 50% chance of getting the same outcome twice.

YO DAWG, I HEARD YOU LIKE ARRAYS
quickmeme.com


YO DAWG, I HEARD YOU LIKE ARRAYS
SO I PUT AN ARRAY IN YOUR ARRAY SO YOU CAN INDEX WHILE YOU INDEX
quickmeme.com

Multidimensional Arrays

- You can create **multidimensional arrays** to represent multidimensional data.

```
int[][] a = new int[3][5];
```

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]

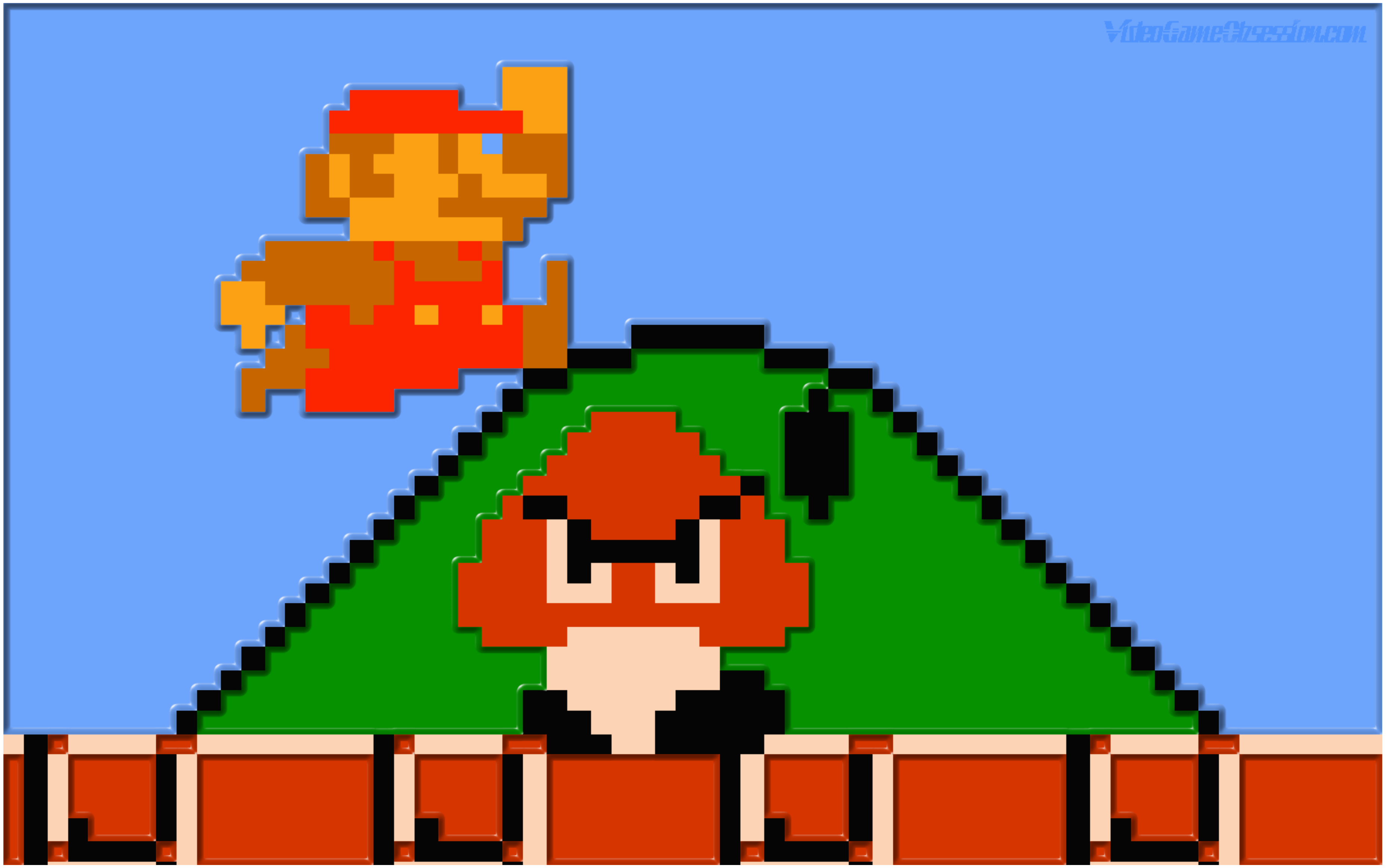
Multidimensional Arrays

- You can create **multidimensional arrays** to represent multidimensional data.

```
Type[][] name = new Type[rows][cols];
```

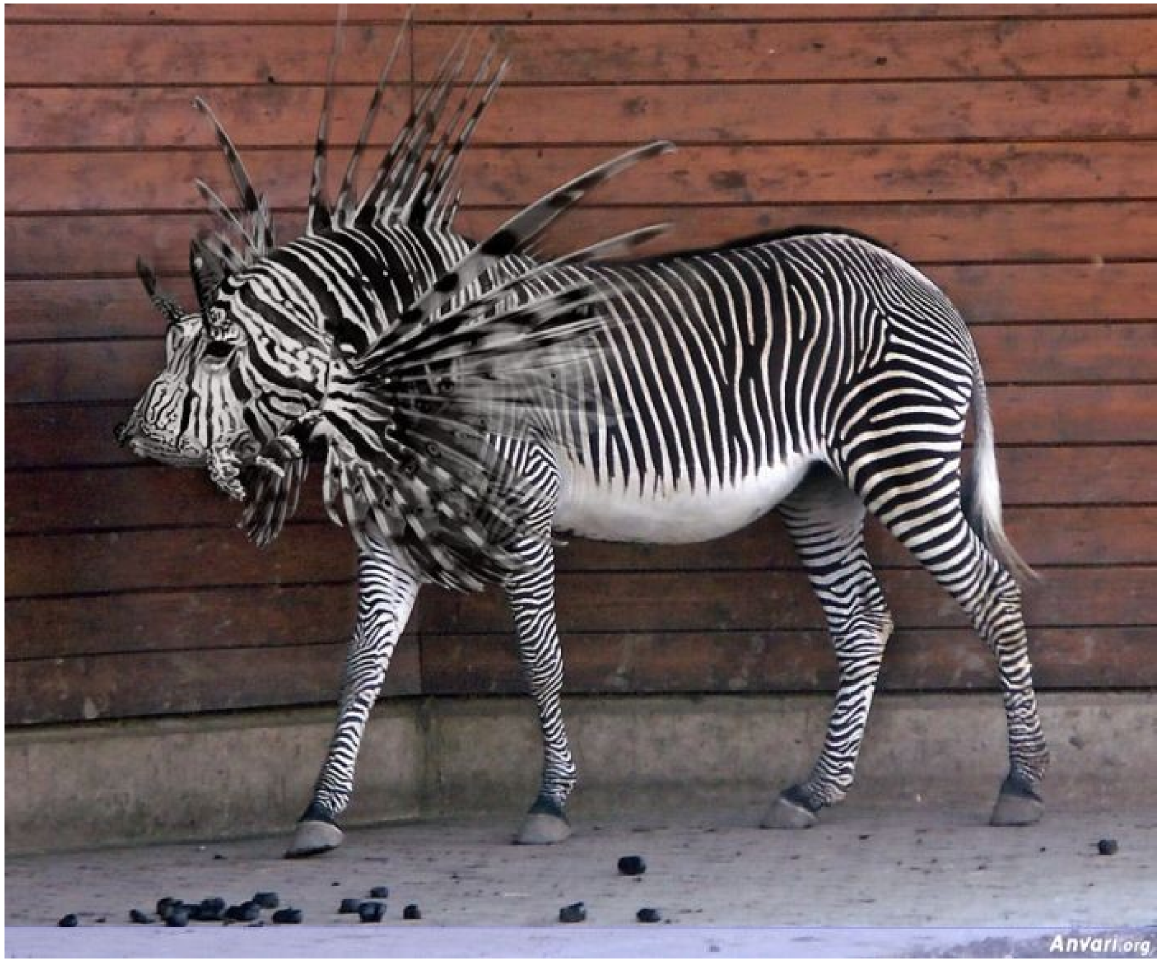
a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]

Working with Images



http://www.videogameobsession.com/videogame/wallpapers/VGO-NES-SuperMario_03-vgo.png

Manipulating Images

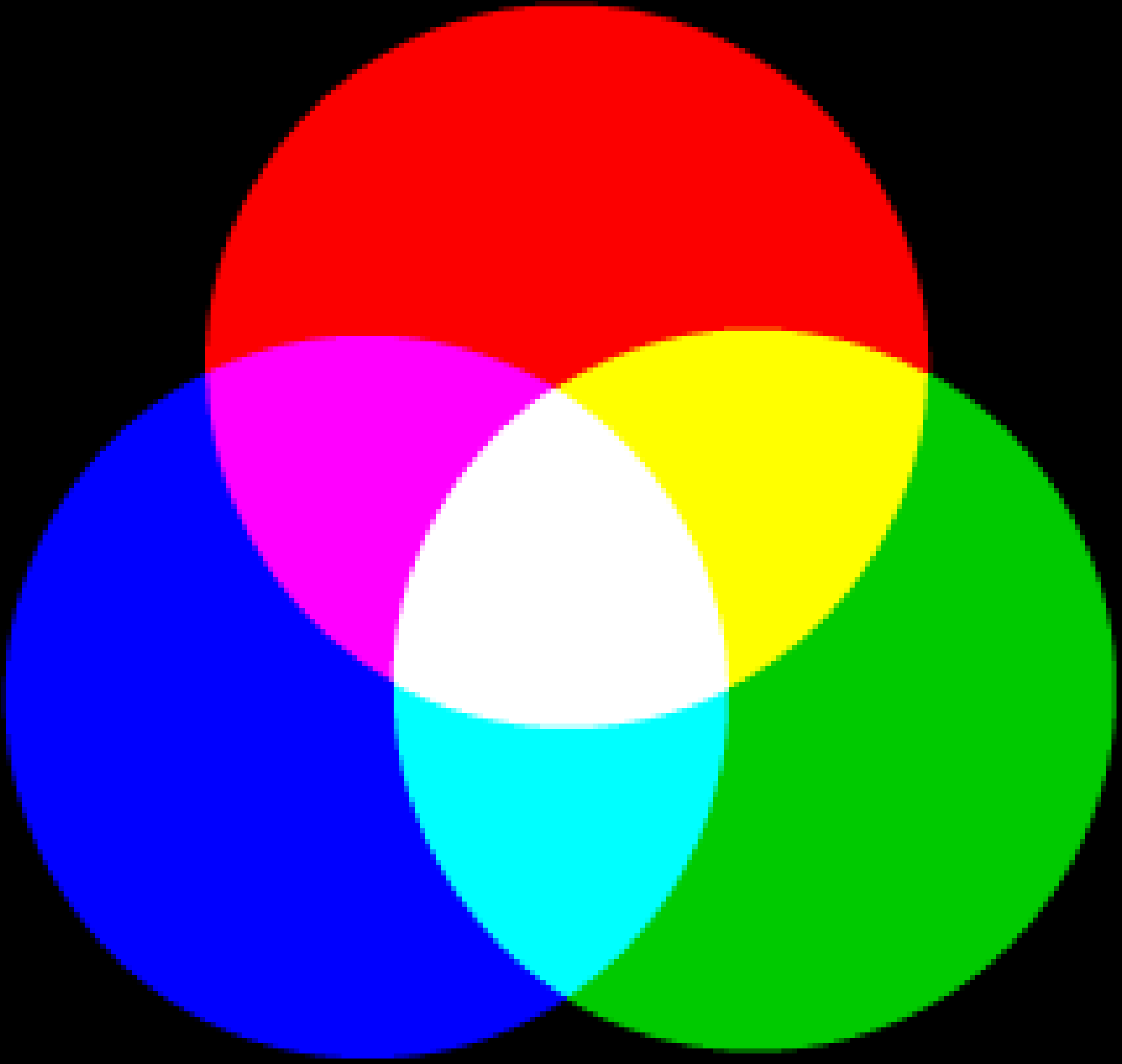


Manipulating Images

- You can obtain an array of pixels from a **`GImage`**.
- You can construct a **`GImage`** from an array of the pixel values.

Representations of Color

- One color format: RGB
  - The human eye has three different types of color receptors that pick up colors (close to) red, green, and blue.
  - Describe the intensity (from 0 to 255) of the red, green, and blue channels.

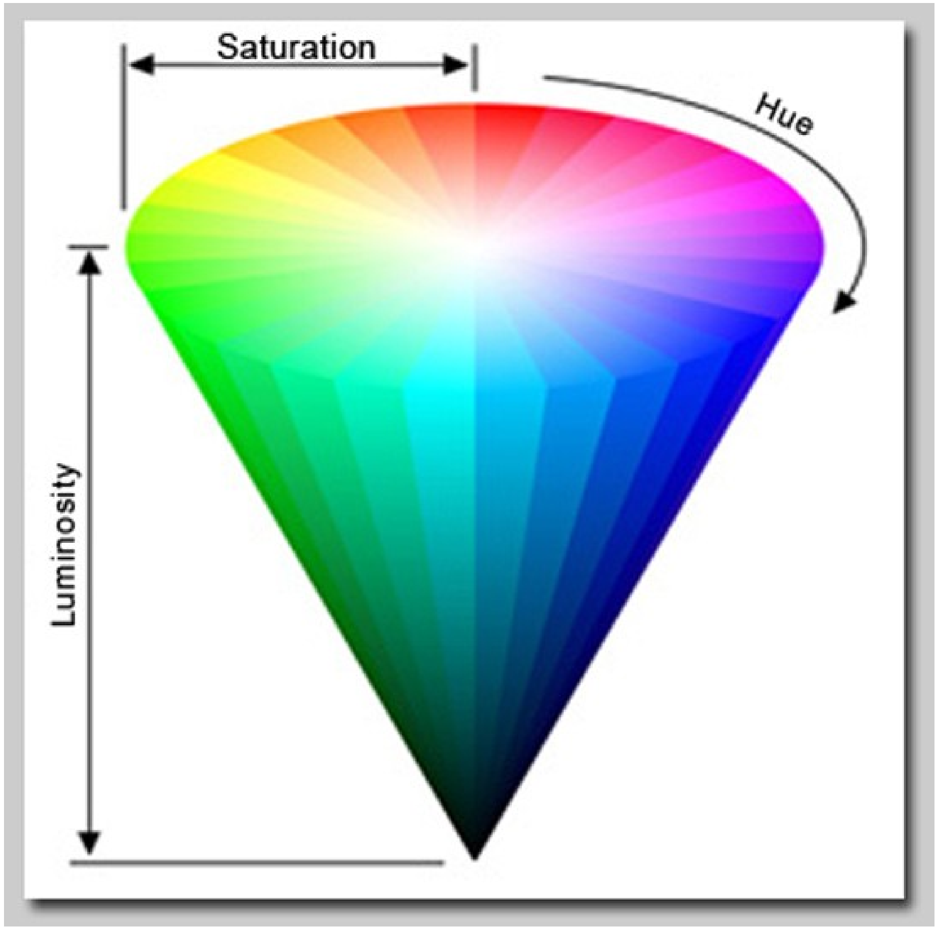


Transforming RGB Colors

- Changing the red, green, and blue channels can warp the color of an image.
- Increasing or decreasing the red, green, and blue values changes the brightness of the image.

Representations of Color

- Another format: HSB:
  - Choose the **hue** (what color), **saturation** (how intense), and **brightness** (absolute brightness).
  - Each choice in the range from 0.0 to 1.0.



Transforming HSB Colors

- Changing the hue changes which color (yellow, green, blue, etc.) is displayed.
- Changing the saturation changes how vibrant the colors are (gray versus colorful)
- Changing the brightness changes how bright the image is.