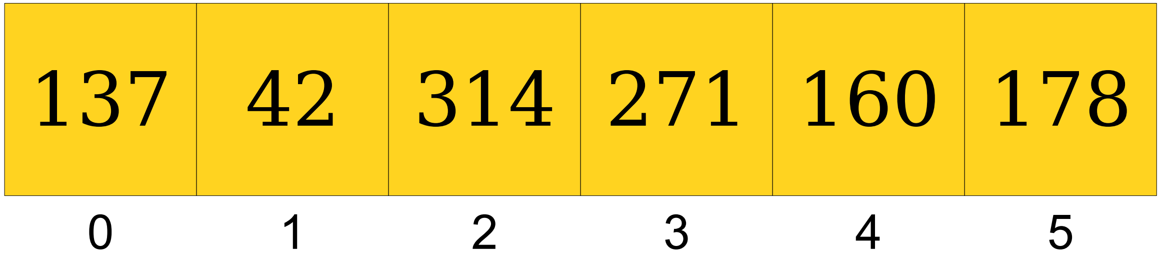
Multidimensional Arrays

Arrays



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

- 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.

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]
```

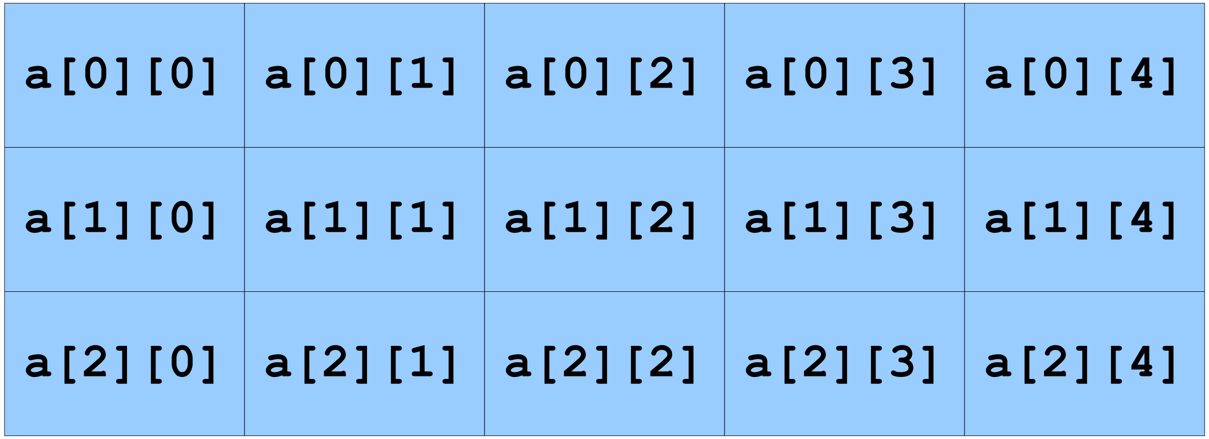
- To read the length of an array, you can read the `length` field (without parentheses):

```
arr.length
```

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[][] arr = new Type[3][5];
```

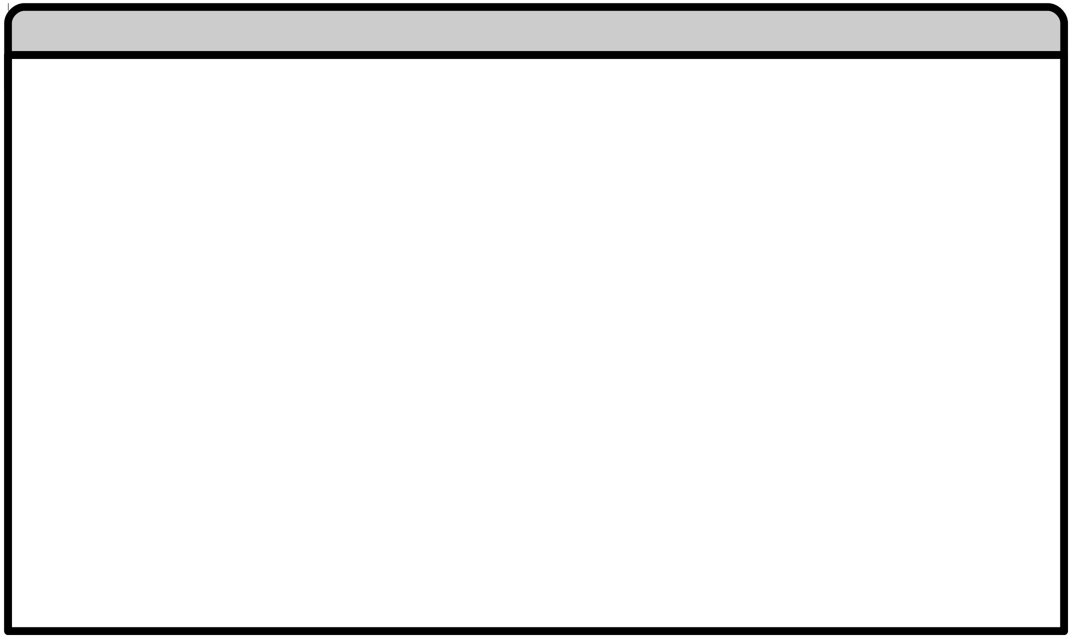
a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]

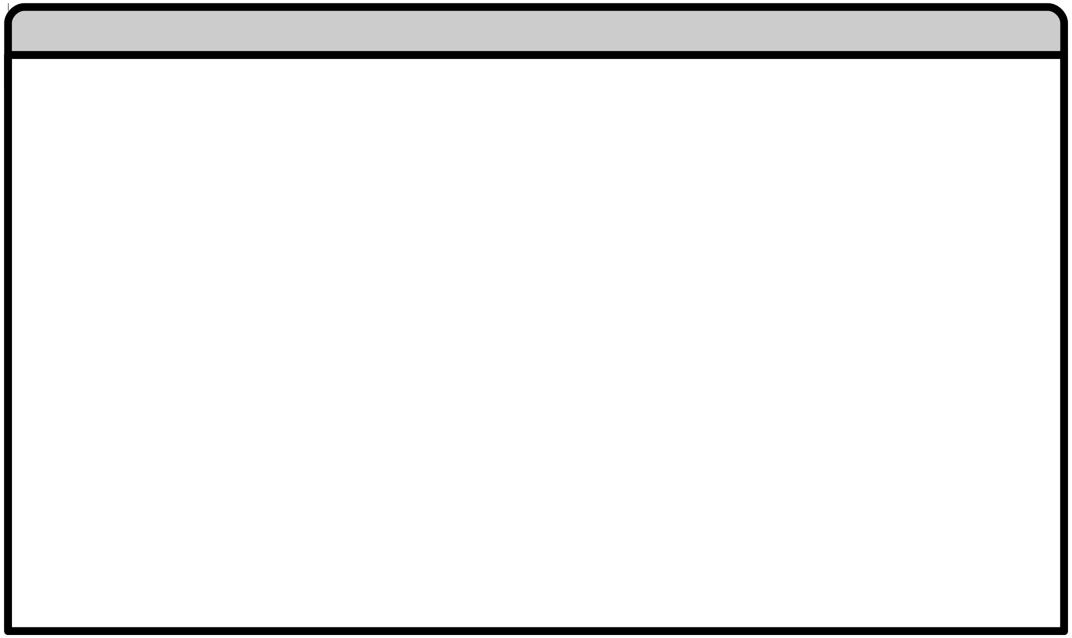
Multidimensional Arrays

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

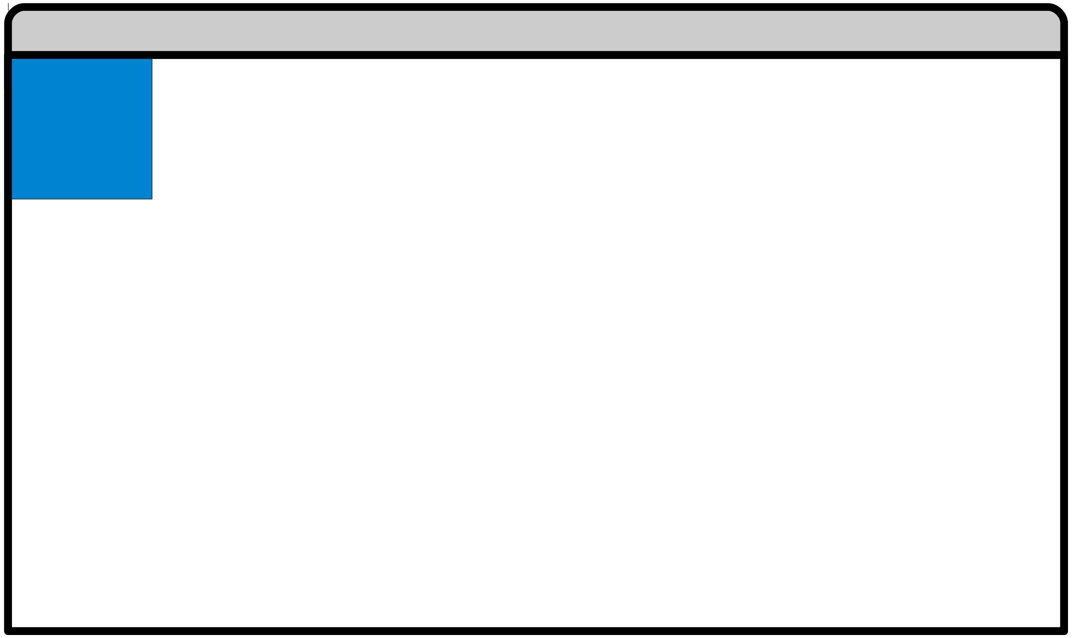
```
Type[][] arr = new Type[rows][cols];
```

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]

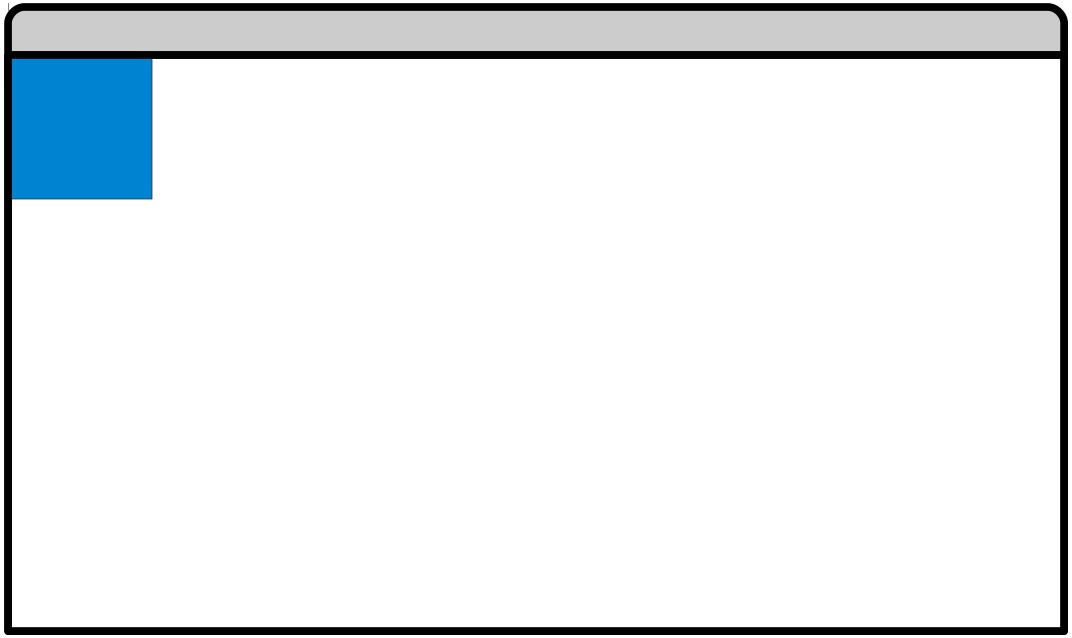




```
GRect box = new GRect(0, 0, BOX_SIZE, BOX_SIZE);
box.setFilled(true);
box.setFillColor(Color.BLUE);
add(box);
```

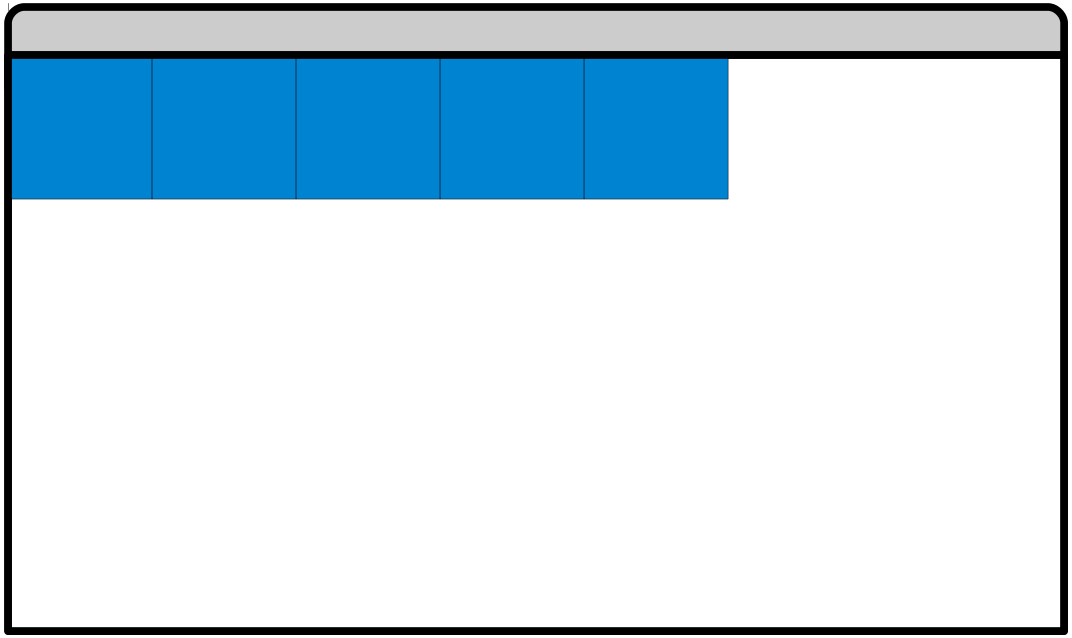


```
GRect box = new GRect(0, 0, BOX_SIZE, BOX_SIZE);
box.setFilled(true);
box.setFillColor(Color.BLUE);
add(box);
```



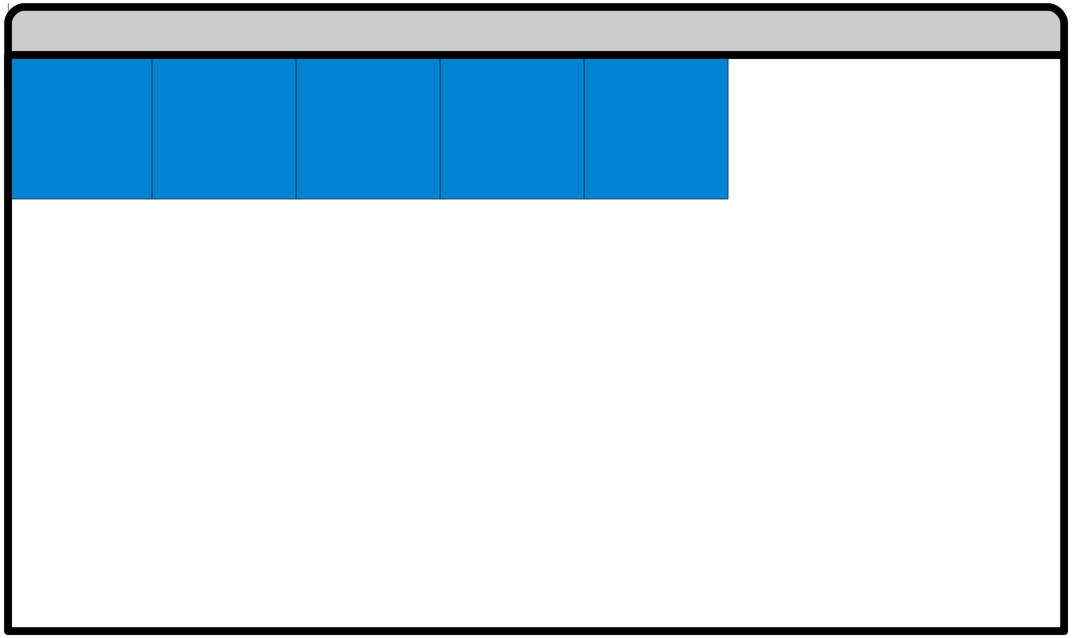
```
for (int j = 0; j < 5; j++) {
    double x = j * BOX_SIZE;


    GRect box = new GRect(x, 0, BOX_SIZE, BOX_SIZE);
    box.setFilled(true);
    box.setFillColor(Color.BLUE);
    add(box);
}
```



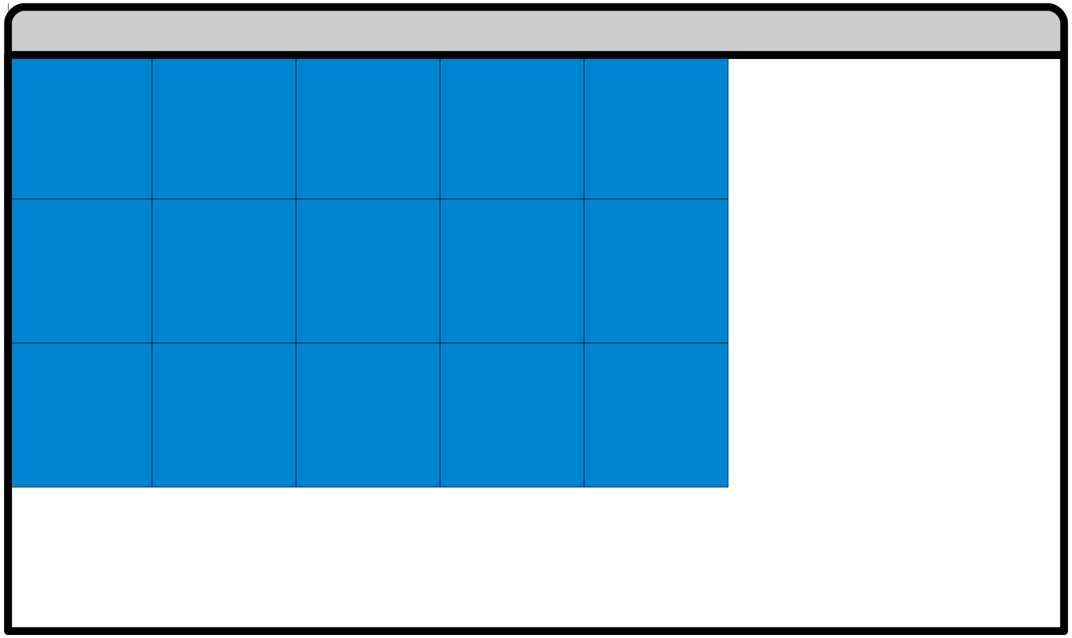
```
for (int j = 0; j < 5; j++) {
    double x = j * BOX_SIZE;


    GRect box = new GRect(x, 0, BOX_SIZE, BOX_SIZE);
    box.setFilled(true);
    box.setFillColor(Color.BLUE);
    add(box);
}
```



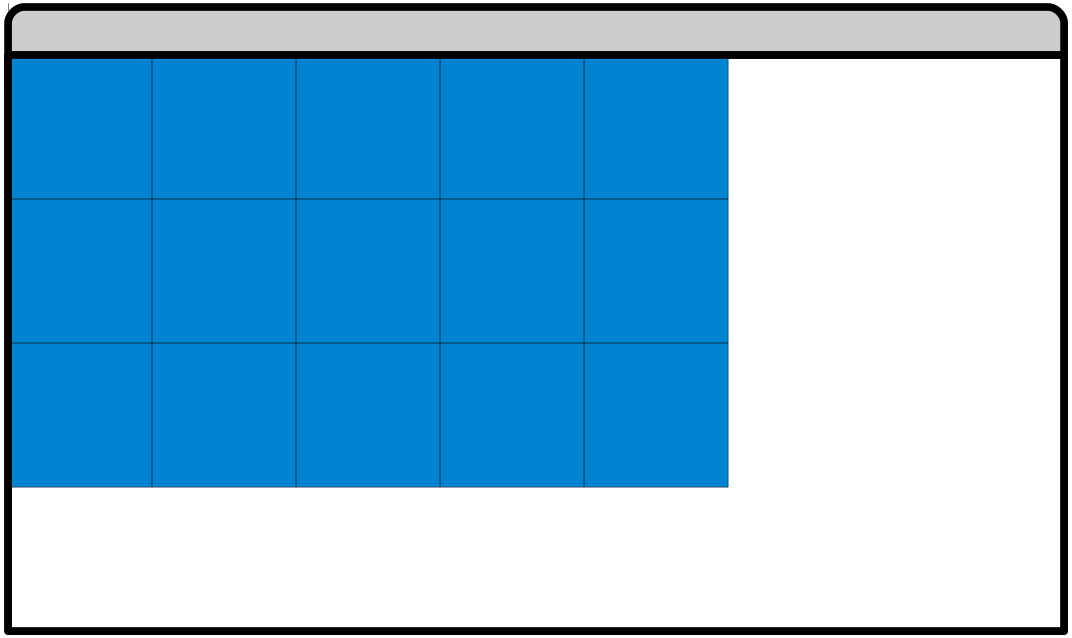
```
for (int i = 0; i < 3; i++) {
   for (int j = 0; j < 5; j++) {
       double x = j * BOX_SIZE;
       double y = i * BOX_SIZE;

       GRect box = new GRect(x, y, BOX_SIZE, BOX_SIZE);
       box.setFilled(true);
       box.setFillColor(Color.BLUE);
       add(box);
   }
}
```



```
for (int i = 0; i < 3; i++) {
   for (int j = 0; j < 5; j++) {
       double x = j * BOX_SIZE;
       double y = i * BOX_SIZE;

       GRect box = new GRect(x, y, BOX_SIZE, BOX_SIZE);
       box.setFilled(true);
       box.setFillColor(Color.BLUE);
       add(box);
   }
}
```



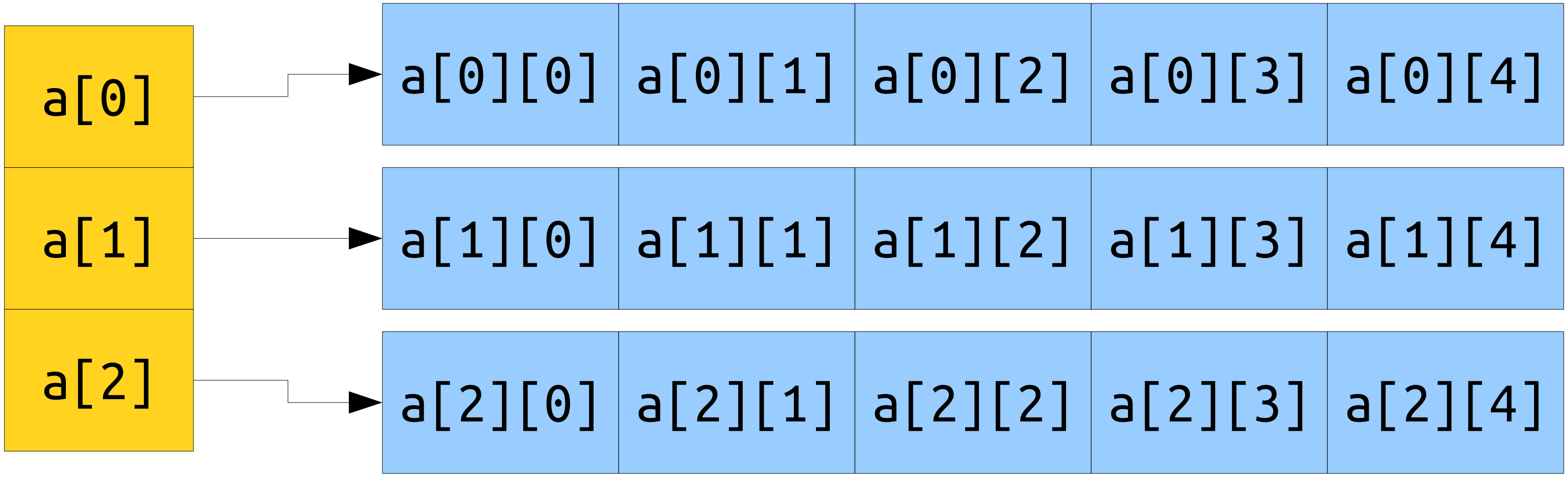
```
for (int i = 0; i < 3; i++) {
   for (int j = 0; j < 5; j++) {
       double x = j * BOX_SIZE;
       double y = i * BOX_SIZE;

       GRect box = new GRect(x, y, BOX_SIZE, BOX_SIZE);
       box.setFilled(true);
       box.setFillColor(Color.BLUE);
       add(box);
   }
}
```

Intuiting Double `for` Loops

- There are two main ways to think about a double `for` loop.
- **As a Unit:**
  - A double `for` loop is a way of saying “iterate over a two-dimensional space.”
- **As a Loop in a Loop:**
  - A double `for` loop is a normal `for` loop wrapped up inside of a second `for` loop.

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]



a[0]

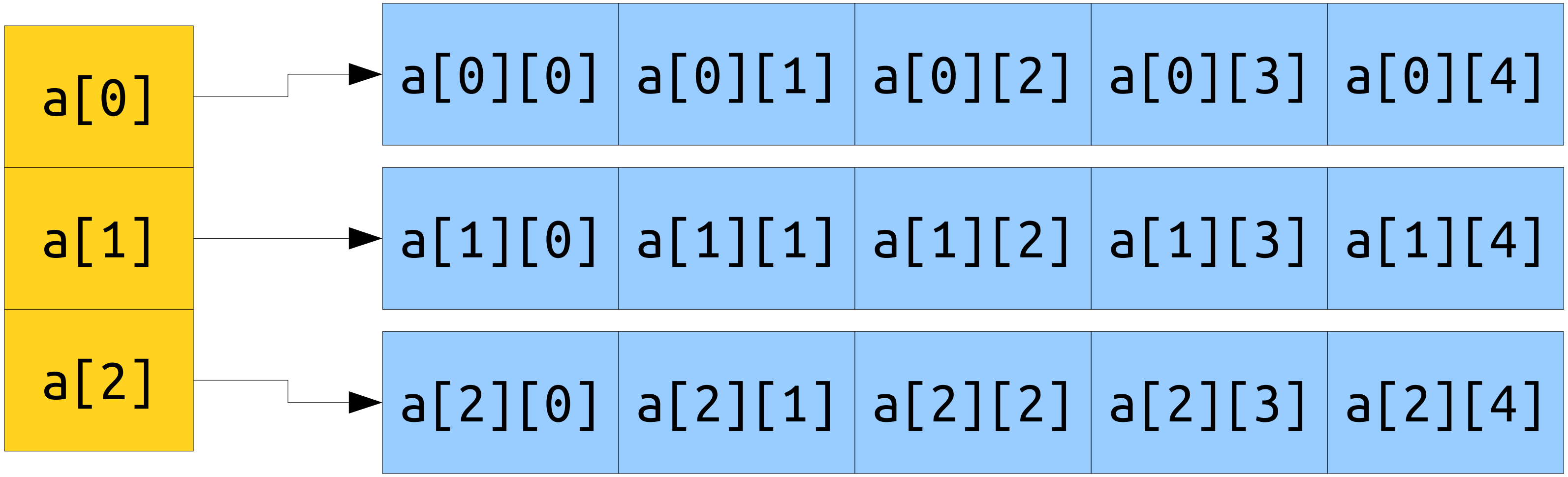
a[1]

a[2]

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]



a[0] a[1] a[2]

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]

`arr[i][j] = (arr[i])[j]`

Intuiting Multidimensional Arrays

- There are two main ways of intuiting a multidimensional array.
- **As a Unit:**
  - A multidimensional array represents a 2D grid.
- **As an Array of Arrays:**
  - A multidimensional is an array whose elements are themselves arrays.

Loops and Multidimensional Arrays

- The canonical way to loop over a multidimensional array is with a double **for** loop:

```
Type[][] arr = /* … */

for (int row = 0; row < arr.length; row++) {
  for (int col = 0; col < arr[row].length; col++) {
     /* … access arr[row][col] … */
  }
}
```

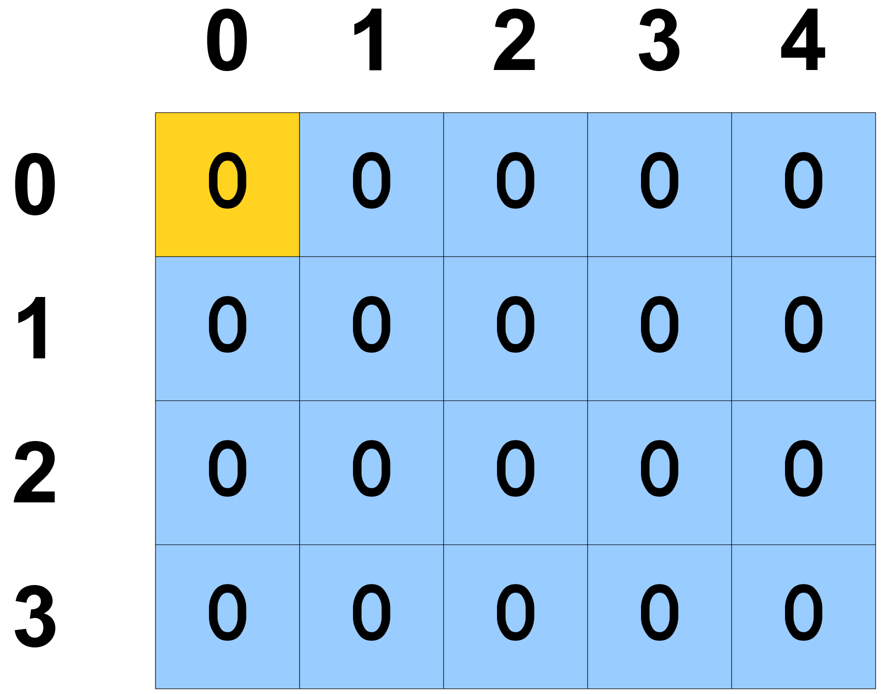
Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
for (int row = 0; row < arr.length; row++) {
  for (int col = 0; col < arr[row].length; col++) {
     arr[row][col] = row + col;
  }
}
```

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

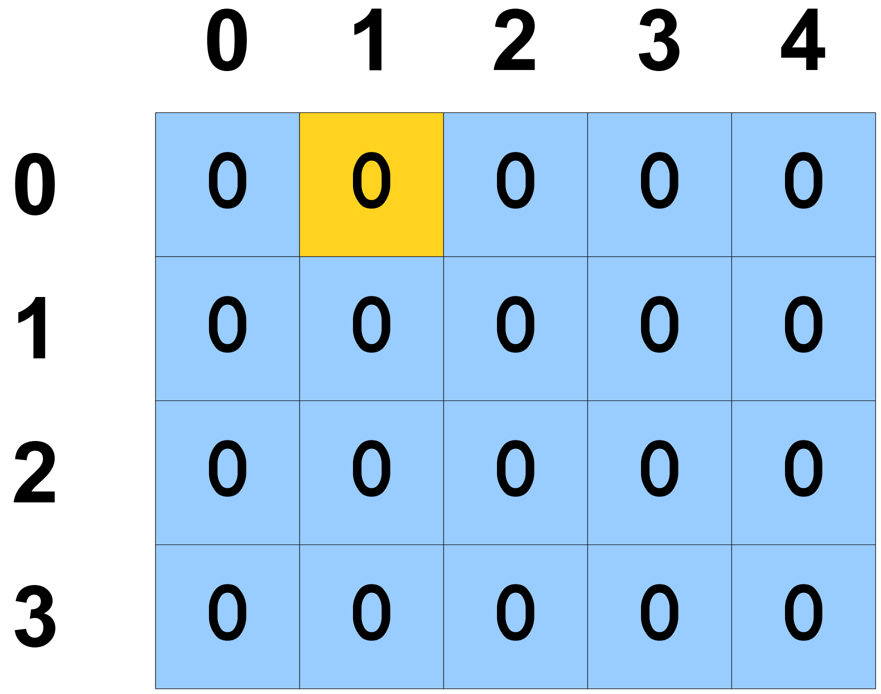
```
int[][] arr = new int[4][5];
for (int row = 0; row < arr.length; row++) {
 for (int col = 0; col < arr[row].length; col++) {
   arr[row][col] = row + col;
 }
}
```



	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
for (int row = 0; row < arr.length; row++) {
  for (int col = 0; col < arr[row].length; col++) {
     arr[row][col] = row + col;
  }
}
```



	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
for (int row = 0; row < arr.length; row++) {
  for (int col = 0; col < arr[row].length; col++) {
     arr[row][col] = row + col;
  }
}
```

	0	1	2	3	4
0	0	1	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

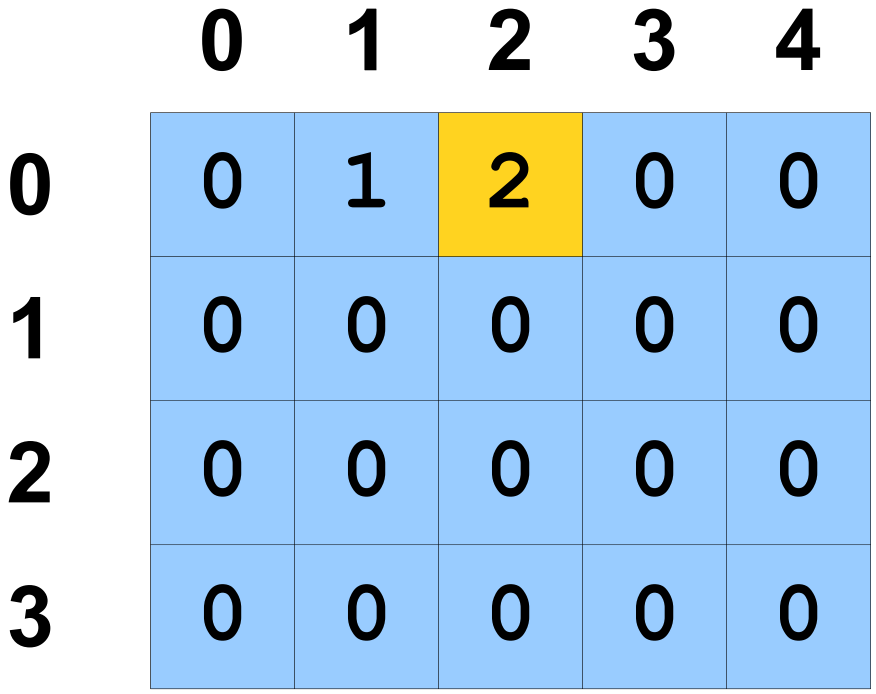
Loops and Multidimensional Arrays

```java
int[][] arr = new int[4][5];
 for (int row = 0; row < arr.length; row++) {
   for (int col = 0; col < arr[row].length; col++) {
      arr[row][col] = row + col;
   }
 }
```

	0	1	2	3	4
0	0	1	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

```java
int[][] arr = new int[4][5];
for (int row = 0; row < arr.length; row++) {
  for (int col = 0; col < arr[row].length; col++) {
     arr[row][col] = row + col;
  }
}
```



	0	1	2	3	4
0	0	1	2	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
 for (int row = 0; row < arr.length; row++) {
   for (int col = 0; col < arr[row].length; col++) {
      arr[row][col] = row + col;
   }
 }
```

	0	1	2	3	4
0	0	1	2	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

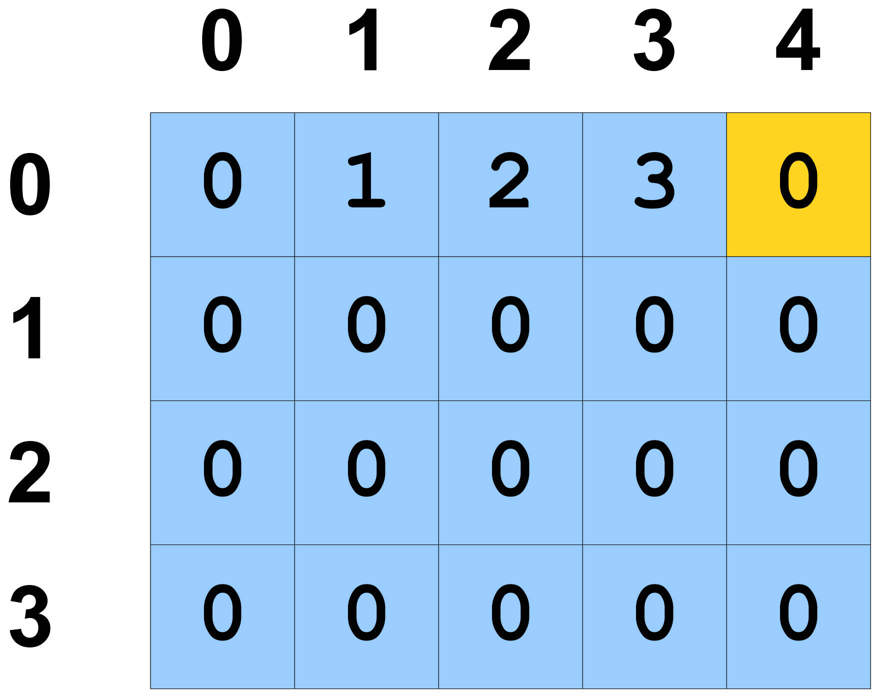
Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
for (int row = 0; row < arr.length; row++) {
  for (int col = 0; col < arr[row].length; col++) {
     arr[row][col] = row + col;
  }
}
```

	0	1	2	3	4
0	0	1	2	3	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
for (int row = 0; row < arr.length; row++) {
 for (int col = 0; col < arr[row].length; col++) {
    arr[row][col] = row + col;
 }
}
```



	0	1	2	3	4
0	0	1	2	3	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
for (int row = 0; row < arr.length; row++) {
  for (int col = 0; col < arr[row].length; col++) {
    arr[row][col] = row + col;
  }
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
for (int row = 0; row < arr.length; row++) {
  for (int col = 0; col < arr[row].length; col++) {
     arr[row][col] = row + col;
  }
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
for (int row = 0; row < arr.length; row++) {
 for (int col = 0; col < arr[row].length; col++) {
   arr[row][col] = row + col;
 }
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
for (int row = 0; row < arr.length; row++) {
   for (int col = 0; col < arr[row].length; col++) {
      arr[row][col] = row + col;
   }
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
for (int row = 0; row < arr.length; row++) {
  for (int col = 0; col < arr[row].length; col++) {
     arr[row][col] = row + col;
  }
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
for (int row = 0; row < arr.length; row++) {
  for (int col = 0; col < arr[row].length; col++) {
     arr[row][col] = row + col;
  }
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
for (int row = 0; row < arr.length; row++) {
  for (int col = 0; col < arr[row].length; col++) {
    arr[row][col] = row + col;
  }
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
 for (int row = 0; row < arr.length; row++) {
   for (int col = 0; col < arr[row].length; col++) {
      arr[row][col] = row + col;
   }
 }
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
 for (int row = 0; row < arr.length; row++) {
   for (int col = 0; col < arr[row].length; col++) {
      arr[row][col] = row + col;
   }
 }
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	0	0	0	0	0
3	0	0	0	0	0

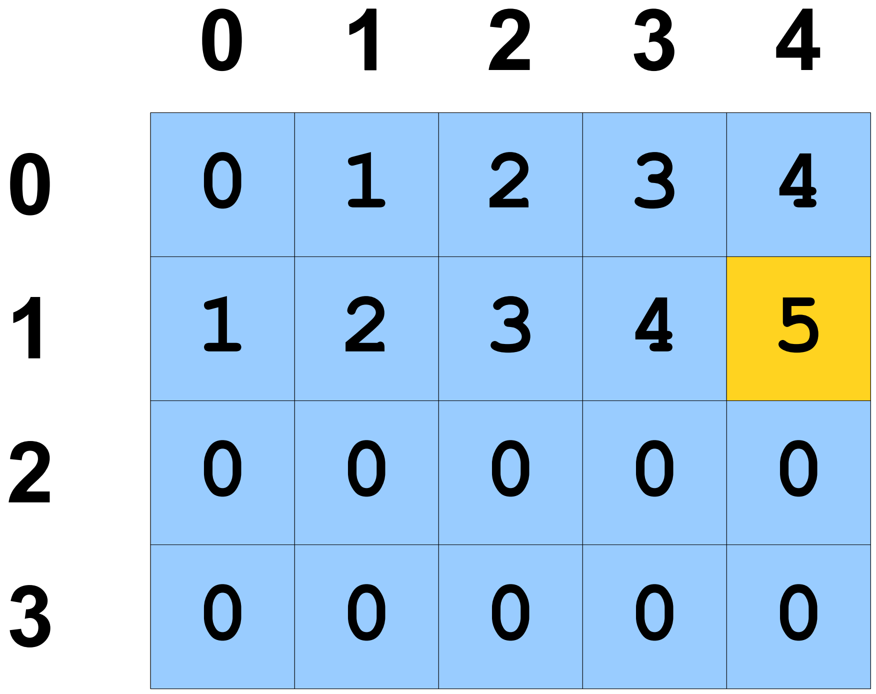
Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
 for (int row = 0; row < arr.length; row++) {
   for (int col = 0; col < arr[row].length; col++) {
      arr[row][col] = row + col;
   }
 }
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
for (int row = 0; row < arr.length; row++) {
  for (int col = 0; col < arr[row].length; col++) {
     arr[row][col] = row + col;
  }
}
```



	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	5
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

```java
int[][] arr = new int[4][5];
for (int row = 0; row < arr.length; row++) {
   for (int col = 0; col < arr[row].length; col++) {
       arr[row][col] = row + col;
   }
}
```

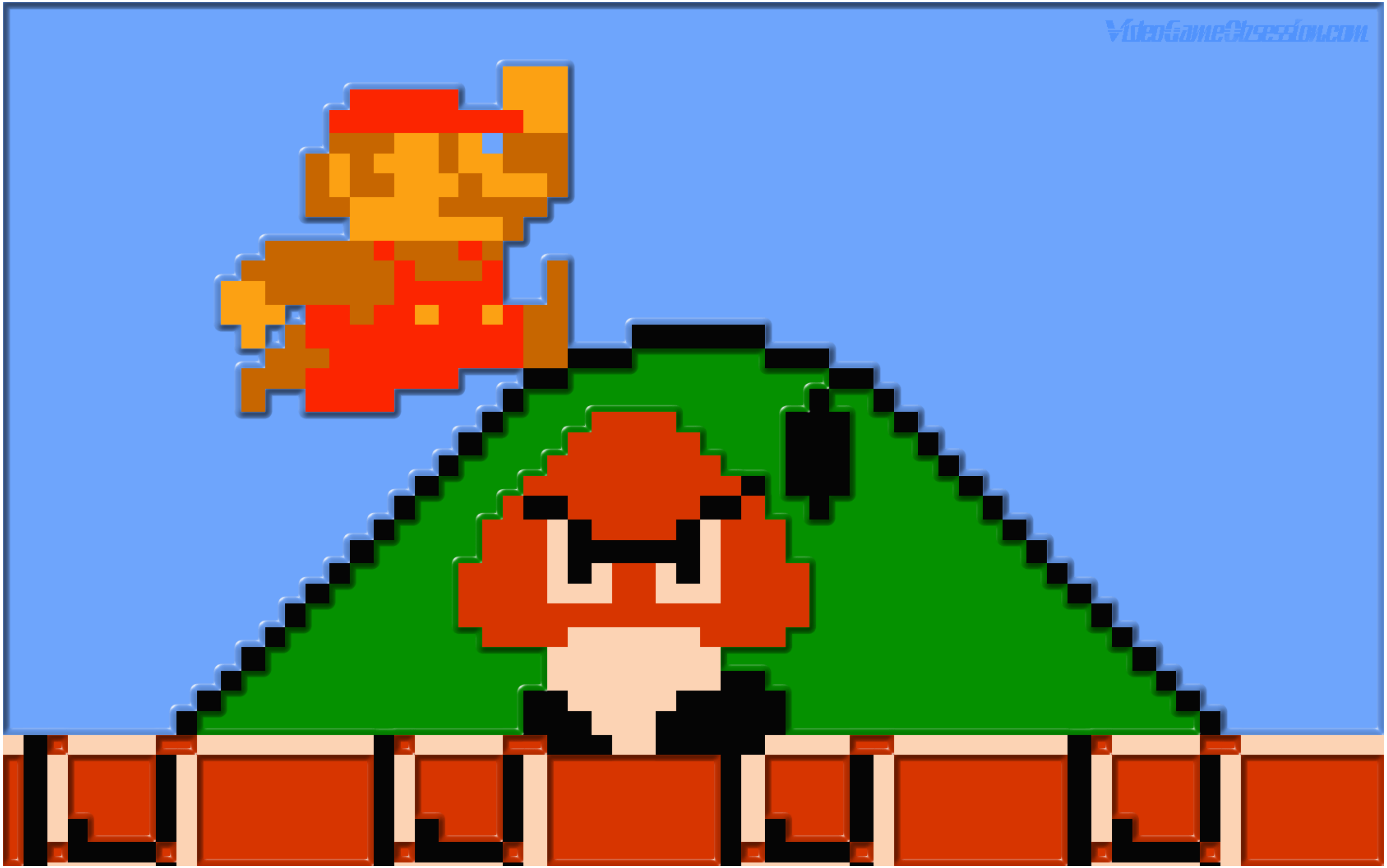
	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	5
2	0	0	0	0	0
3	0	0	0	0	0

Loops and Multidimensional Arrays

```
int[][] arr = new int[4][5];
for (int row = 0; row < arr.length; row++) {
  for (int col = 0; col < arr[row].length; col++) {
     arr[row][col] = row + col;
  }
}
```

	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	5
2	2	3	4	5	6
3	3	4	5	6	7

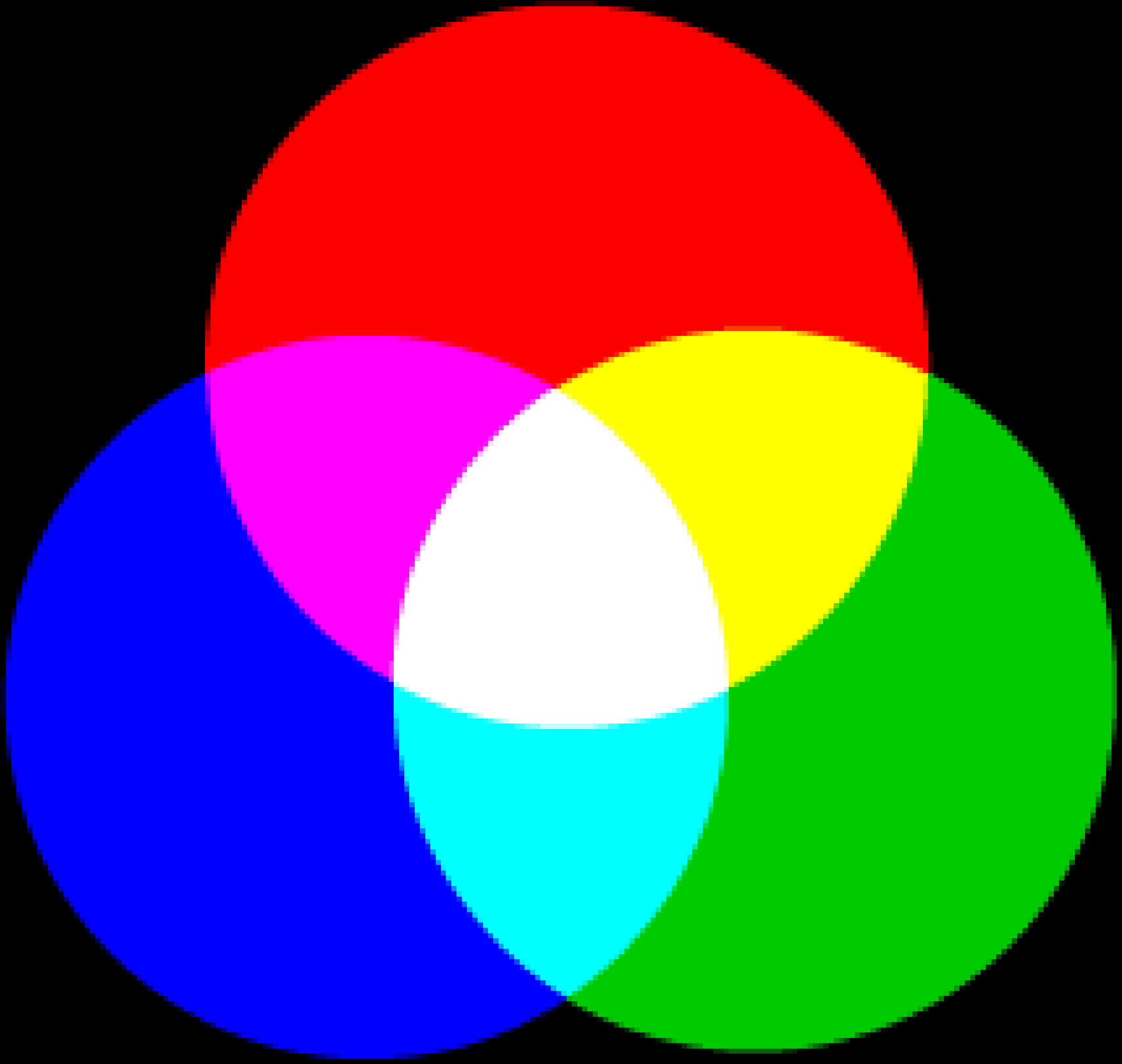
Working with Images



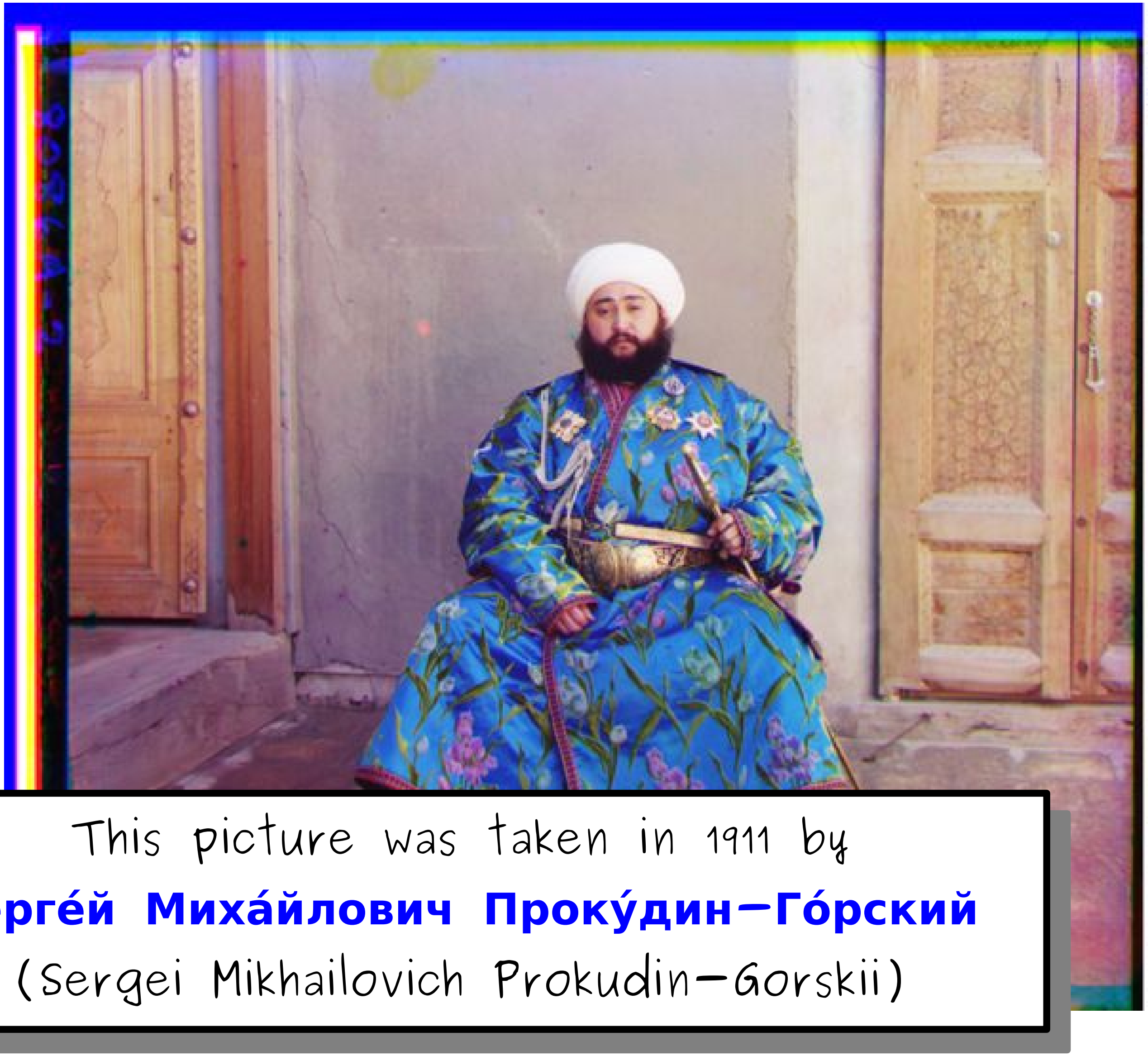
http://www.videogameobsession.com/videogame/wallpapers/VGO-NES-SuperMario_03-vgo.png

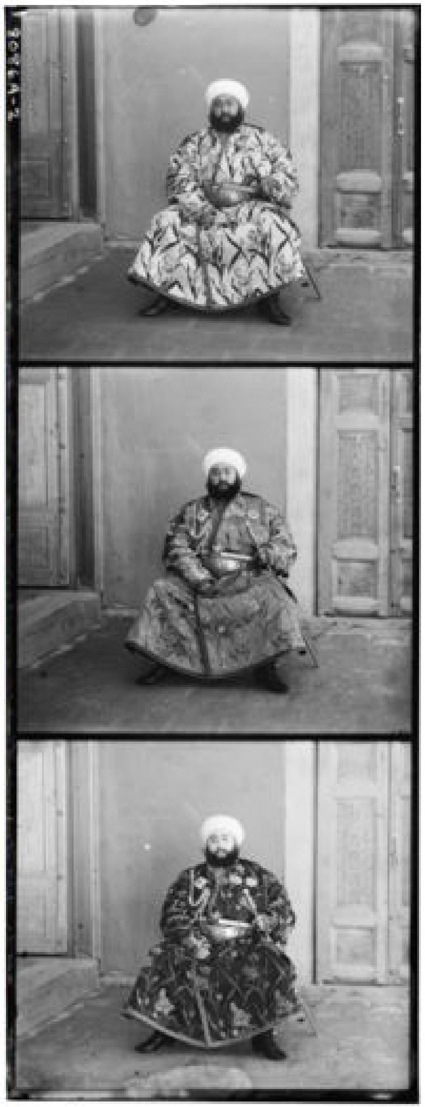
Representations of Color

- The human eye has three different types of color receptors that pick up colors (close to) red, green, and blue.
- Computers usually represent color as ***RGB triplets***:
  - Describe the intensity of the red, green, and blue components of the color.
  - Values range from 0 (min) to 255 (max), inclusive.



Early Color Photographs





This picture was taken in 1911 by
Серге́й Миха́йлович Проку́дин-Го́рский
(Sergei Mikhailovich Prokudin-Gorskii)

http://en.m.wikipedia.org/wiki/File:Rgb-compose-Alim_Khan.jpg

Early Color US Photographs:

http://www.collectorsweekly.com/articles/the-forgotten-photo-technology-that-romanticized-america/

RGB Triples and `GImages`

- Up to this point, we've been using the `Color` type to represent colors.
- When working with individual pixels in a `GImage`, we represent colors as **`int`**s.
- We use cute tricks with the internal representations of **`int`**s to store four values in one **`int`**.
    - If you're curious, check the book for details. We're not going to expect you to know this.
- Takeaway point: with `GImages`, **`int`** can mean “color.” Don't try using the numeric value itself; you'll go blind.

Creating `GImages`

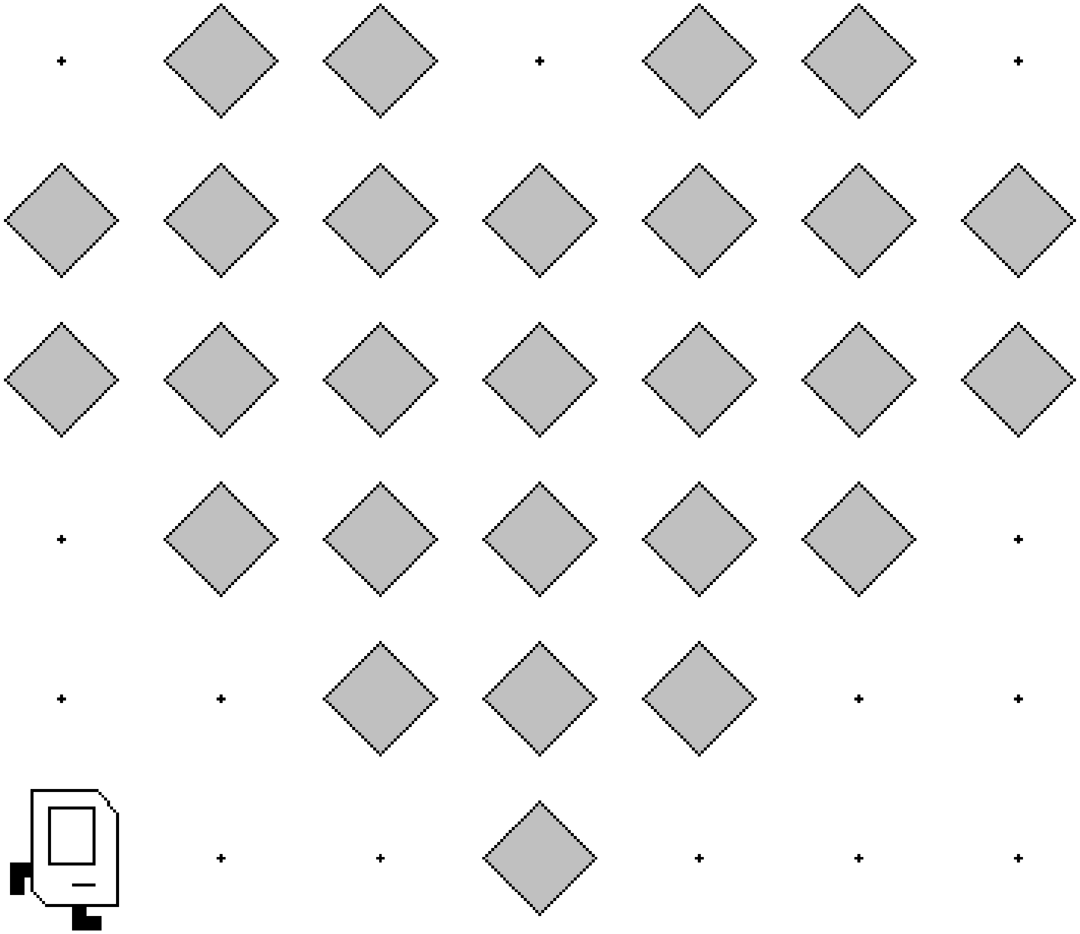
- It is possible to directly create a `GImage` by specifying the RGB values of each pixel in the image.
- To do so:
  - Create an **`int`**`[][]` two-dimensional array to hold the pixel values.
  - Use `GImage.createRGBPixel` to convert the RGB triplets to **`int`**.
  - Construct a **`new`** `GImage` from the array.

Time-Out for Announcements!

Announcements

- Assignment 5 is due Wednesday at 3:15PM.
    - ***Recommendation:*** Have drafts of all four parts of the assignment completed by Monday. Do extensive testing on the first two parts.

Happy Valentine's Day!



Back to CS106A!

Manipulating Images

- You can extract an array of pixels from a `GImage` by calling

```
image.getPixelArray()
```

- You can then create a new image by changing the pixel values.
  - Changing these pixel values doesn't change the underlying image; the `getPixelArray` method returns a copy of the pixels.
- Can read color components with `GImage.getRed`, `GImage.getGreen`, and `GImage.getBlue`.