For our purposes, a picture is an image stored in a JPEG file. JPEG is an international standard for how to store images with high quality but in little space. JPEG is a *lossy compression* format. That means that it is *compressed*, made smaller, but not with 100% of the quality of the original format. Typically, though, what gets thrown away is stuff that you don’t see or don’t notice anyway. For most purposes, a JPEG image works fine.

If we want to write programs to manipulate JPEG images we need to understand how they are stored and displayed. To do this we need to understand arrays, matrices, pixels, and color.

An array is a sequence of elements, each with an index number associated with it. The first element in an array is at index 0, the second at index 1, the third at index 2, and so on. The last element of the array will always be at the length of the array minus one. An array with 5 elements will have its last element at index 4.

It may sound strange to say that the first element of an array is at index 0 but the index is based on the distance from the beginning of the array to the element. Since the first item of the array is at the beginning of the array the distance is 0. Why is the index based on the distance? Array values are stored one after the other in memory. This makes it easy to find any element of the array by multiplying the size of each element by the index and adding it to the address of the beginning of the array. If you are looking for the element at index 3 in an array and the size of each element is 4 bytes long and the array starts at memory location 26 then the 3rd element is at \((3 \times 4 + 26 = 12 + 26 = 38)\).

![Array Elements](image.png)

**FIGURE 4.1:** A depiction of the first five elements in an array

Every time you join a line (queue) of people you are in something like an array. All you usually care about is how far you are from the front of the line. If you are at the front of the line then that is index 0 (you are next). If you are the second one in line then you are at index 1 (there is one person in front of you). If you are the third person in line then you are at index 2 (there are two people in front of you).

Arrays are a great way to store lots of data of the same type. You wouldn’t want to create a different variable for every pixel in a picture when there are hundreds of thousands of pixels in a picture. Instead you use an array of pixels. You still need a way to refer to a particular pixel, so we use an index for that. You can access elements of an array in Java using `arrayName[index]`. For example, to access the first element in an array variable named pixels use `pixels[0]`. To access the second element use `pixels[1]`. To access the third element use `pixels[2]`. You can get the number of items in an array using `arrayName.length`. So, to
access the last element in the array use `arrayName[arrayName.length - 1]`.

To declare an array in Java you specify the type and then use open and close square brackets followed by a name for the array.

```java
> double[] grades;
> System.out.println(grades);
null
```

or you could have specified the square brackets after the variable name:

```java
> double grades[];
> System.out.println(grades);
null
```

The above code declares an array of doubles with the name `grades`. Notice though that this just declared an object reference and set it to null. It didn’t create the array. In Java you can create an array and specify the values for it at the same time:

```java
> double[] gradeArray = {80, 90.5, 88, 92, 94.5};
> System.out.println(gradeArray.length);
5
> System.out.println(gradeArray[0]);
80.0
> System.out.println(gradeArray[4]);
94.5
```

Making it Work Tip: Using dot notation for public fields
Notice that there are no parentheses following `arrayName.length`. This is because length is not a method but a public field (data). Public fields can be accessed using dot notation `objectName.fieldName`. Methods always have parenthesis after the method name even if there are no input parameters, such as `FileChooser.pickAFile()`.

A two-dimensional array is a matrix. A matrix is a collection of elements arranged in both a horizontal and vertical sequence. For one dimensional arrays you would talk about an element at index `i`, that is `array[i]`. For two-dimensional arrays you can talk about an element at row `r` and column `c`, that is, `matrix[r][c]`. This is called row-major order.

Have you ever played the game Battleship™? If you have then you had to specify both the row and column of your guess (B-3) This means row B and column 3 (Figure 4.2). Have you ever gone to a play? Usually your ticket has a row and seat number. These are both examples of row-major two-dimensional arrays.

Another way to specify a location in a two-dimensional array is column-major order which specifies the column first and then the row: `matrix[c][r]`. This is