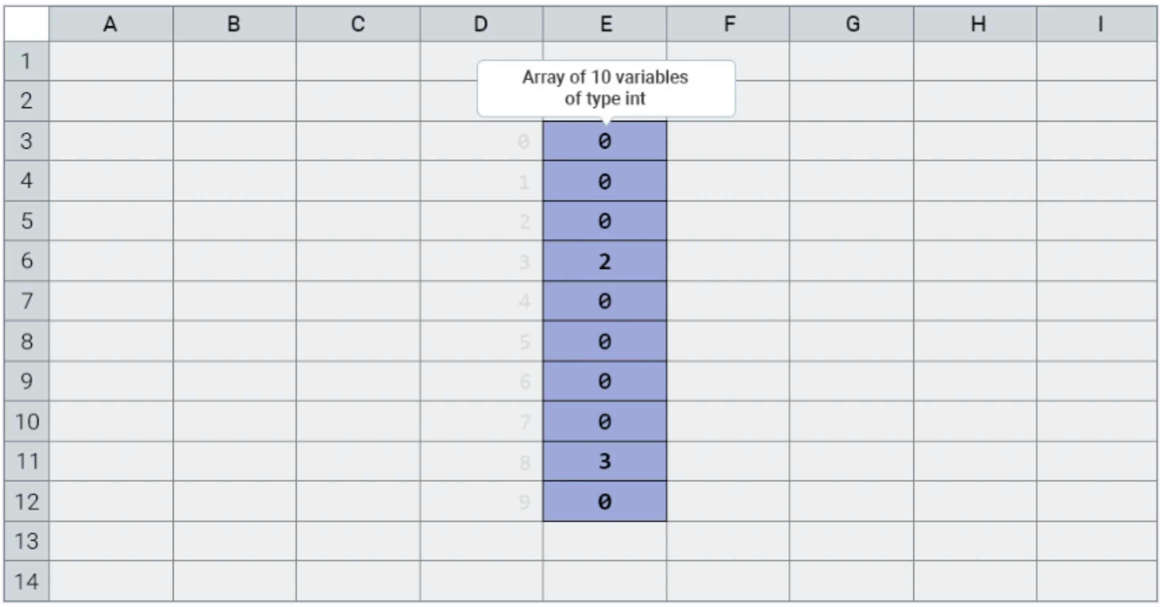
Arrays

1. An array is a container of elements

You've probably heard that computers can process huge amounts of information. Of course, conditional statements (`if-else`) and loops (`for`, `while`) are a big help here. But they can only take you so far. After all, the data you process needs to be stored somehow.

Like almost all programming languages, Java facilitates data processing by providing this great thing called an *array* (`Array` class). They are also sometimes called tables.

An *array* is a special object that lets you store not one value, but several.



Earlier we compared a variable to a box (in which you can store any value). Continuing that analogy, we can think of an array as a box that has internal compartments. Each compartment in the "box" (array) has a number. Of course, the numbering starts from zero...

Or we can make another analogy. Let's compare an ordinary house and a high-rise apartment building. An ordinary house is occupied by a single family, but a high-rise apartment building is divided into apartments. If you want to send a letter to a family that lives in an ordinary house, you indicate the unique address of the house. And to send a letter to a family that lives in an apartment, you indicate the unique address of the building as well as the apartment number.

An array variable is like a high-rise variable. It can store not one but many values. Such a variable has several apartments (cells). Each of them can be addressed by their number (index).

To do this, after the name of the variable, you indicate the index of the cell you want to address, wrapping the index in square brackets. This is pretty simple:

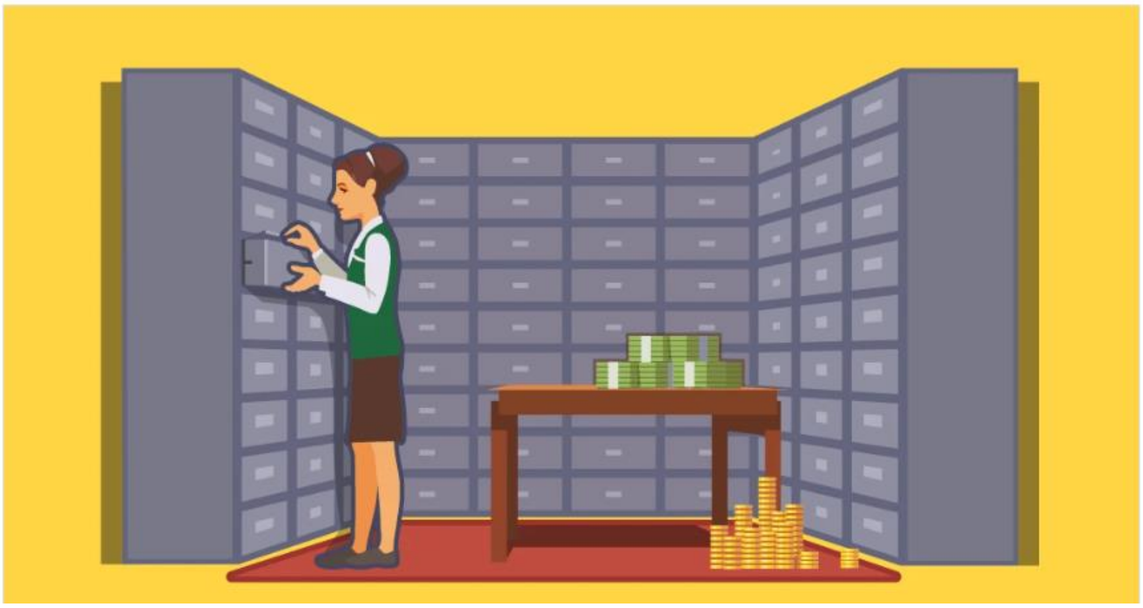
```
array[index] = value;
```

Where `array` is the name of the array variable, `index` is the cell number in the array, and `value` is the value that we want to put into the specified cell.

But to start, let's out how to create arrays.

---

2. Creating an array of elements in Java



Let's say your program needs to store `100` integers somewhere. An array would be a good candidate for this. And how do you create one?

If we wanted to store a single integer, the `int` type would suit us. But if we want to store `100` integers, we probably need an *array* of `int`s. This is how the code to create one would look:

```
int[] array = new int[100];
```

Let's explore this statement.

As you might have guessed, to the left of the equal sign we have the declaration of a variable named `array` whose type is `int[]`. The `int` type is followed by square brackets, which hints that "boxes" of this type can store not one but several values.

To the right of the equal sign, we have an example of "object creation" (the `new` keyword) to get `100` elements (cells) whose type is int. Nothing too difficult here.

Similarly, if we wanted to create an array of 20 cells to store real numbers, then our code would look something like this:

```
double[] vals = new double[20];
```

The number of cells in an array is called the size of the array or the length of the array. And because arrays can store many values, they are also called containers.

Here's an important fact: you cannot change the size of an array after it is created.

You can create a new one, but the length of the existing container cannot be changed.

3. Working with the cells of an array

Okay, we've learned how to create arrays. Now how do we work with them?

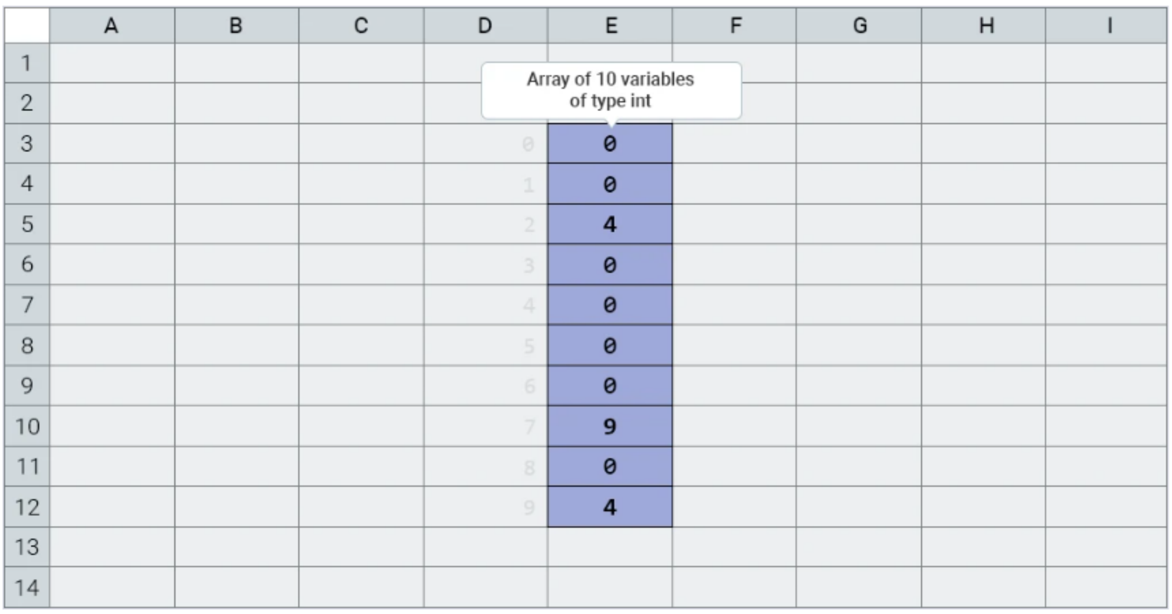
Well, in almost the same way as with ordinary variables. The only difference is that after the name of the array variable, we have to indicate the number of the cell that we are working with.

The numbering of cells in an array always starts from zero. If we have an array of `10` elements, then the numbers (indices) of its cells are `0..9`. If the array contains `200` elements, then the indices are `0..199`. And so on by analogy.

Examples:

Code	Explanation
`int[] a = new int[10];`	Create an array of `10` `int` elements.
`a[2] = 4;`	Assign the value `4` to the cell with index `2`.
`a[7] = 9;`	Assign the value `9` to the cell with index `7`.
`a[9] = a[2] + a[5];`	In the cell with index `9`, write the sum of the values that are stored in cells `2` (which stores the value `4`) and `5` (which stores the value `0`).

This is what will be stored in memory after this code is executed:



The column on the left (in gray) represents the cell numbers (indices). The cells store the assigned values: `4`, `9` and `4`. When the array is created its cells are all filled with zeros.

This is important. All cells in an array have the same data type. If we create an array of `String`s, only strings can be stored in its cells. An array's data type is specified when it is created. Neither the data type nor the length of the array can be changed later.