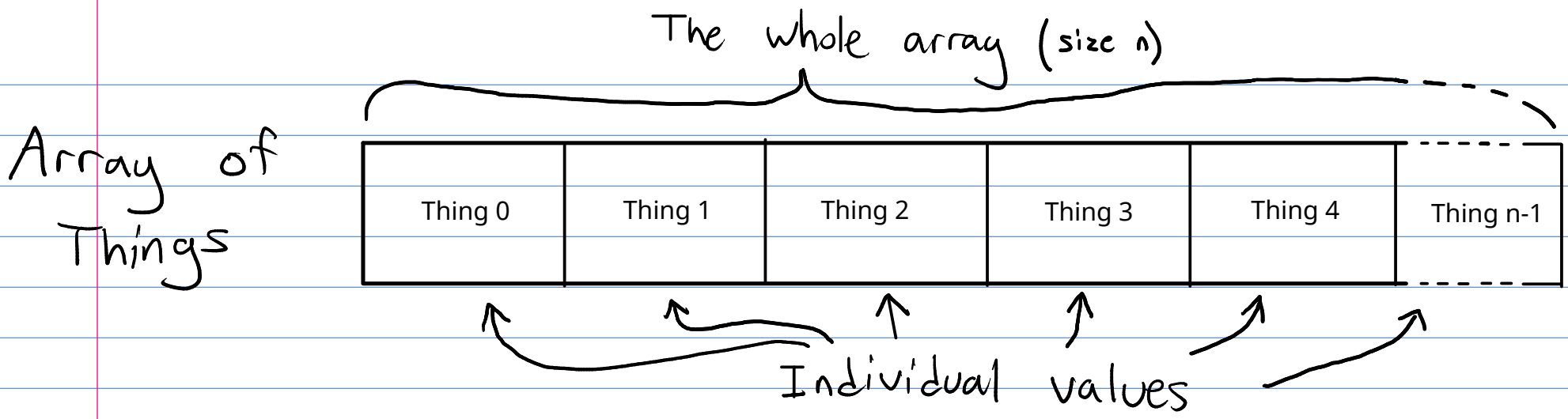


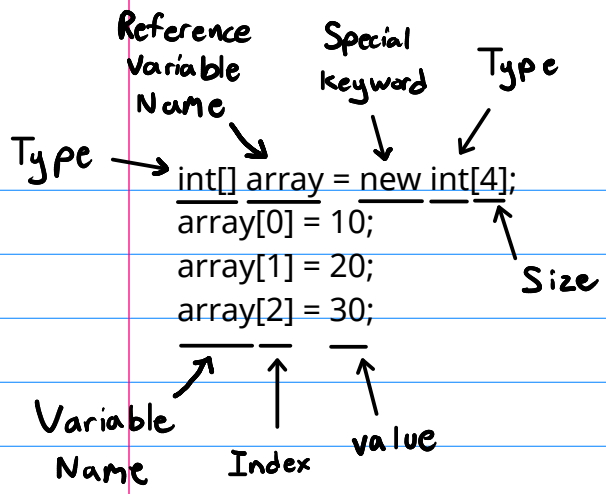

And no, I don't have any connections to Dr. Seuss!

An array can be thought of as a collection of "things". The type of things you can store in the array depend on the type of the array, which is ALWAYS declared when the variable is created in code. For example, the following statement would declare a reference variable to refer to an array of integers (although the object itself would not exist yet):
int[] myIntegerArray;

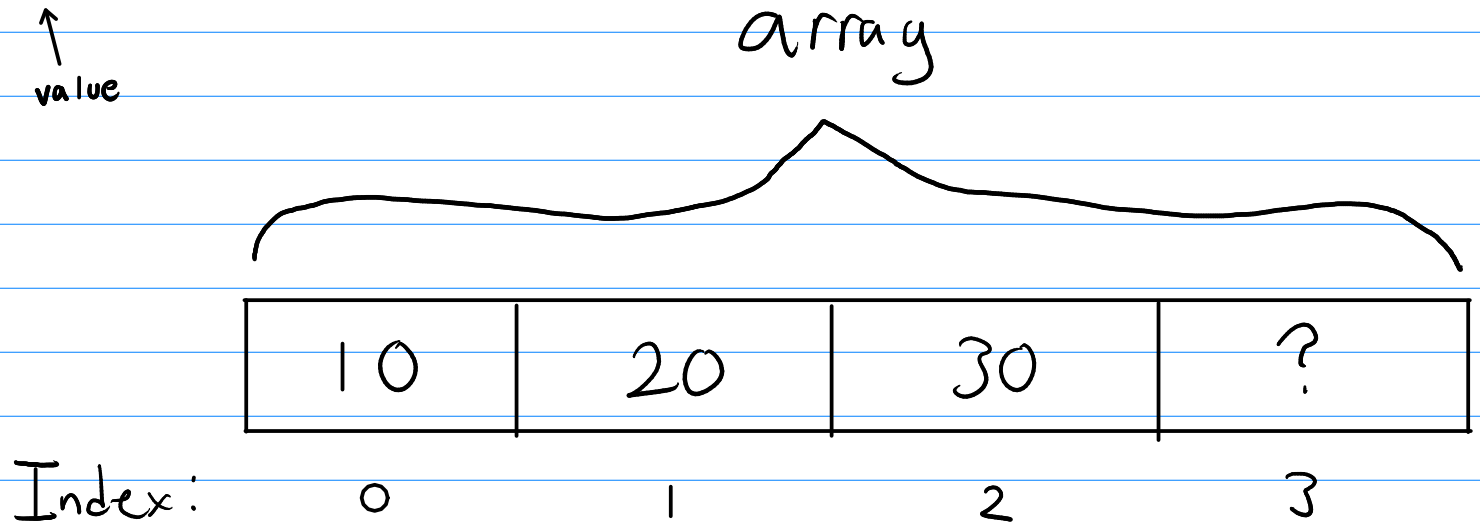
The above statement would be very similar to this statement:
int myInteger;

Like most of everything in Java, an array is an object! An array is a special type of object in Java, and what makes it special is that it is "built-in" to the Java language itself. It is possible to create an array of basically any type in Java, including primitive data types, as well as objects. It's also possible to create arrays OF arrays, which in this case would be a 2-dimensional array, but let's just stick with the basics for now.

Let's make some arrays with some very basic code!




Note: The special keyword "new" is a keyword in Java that tells the compiler to create a "new" instance of an object. For more information about objects, check out my ClassObjectInstanceDemo.




So with the basic code above, we created an array of size 4, and then assigned values to indexes 0, 1, and 2, but what about index 3?

The cool thing about creating arrays of PRIMITIVE data types (int, double, byte, char, long, short, etc.) is that Java will automatically initialize every value at every index to 0. So in our example code above, the value at index 3 is just a plain 0! Note: this doesn't work with objects. If you create an array of objects, Java will initialize all of them to null pointers, and we'll talk about what "null" and "pointers" mean later.