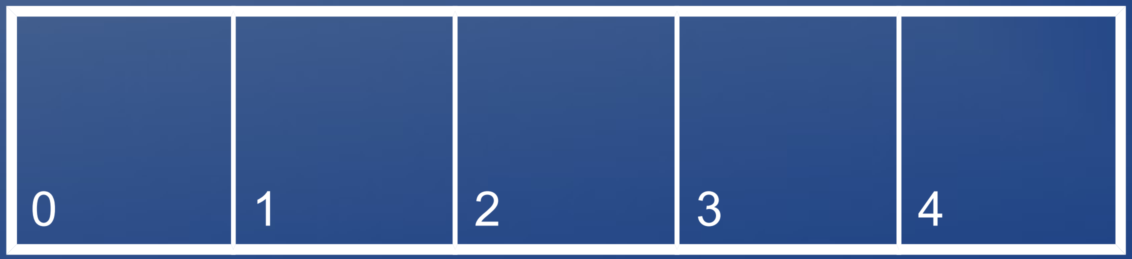
Using Arrays with Methods in Java

Review - What is an Array?

Recall: An array is a collection of one type of data (e.g., integer, string) that is used for a single purpose (e.g., grades, addresses).

Each box is called an element of the array, and the position of each element is the index.



an array with 5 elements

Methods using Arrays

For the purposes of a function or procedure, an array is just like any other variable. You can pass it to the subprogram as a parameter (just like an integer, string, or double)

A function can return an array, and a procedure can modify an array if required.

Since the declaration of an array is more complicated than a variable, we must show the same care when using arrays with functions and procedures.

Arrays in Java

An array in Java is typically declared in one step:

```
dataType name = new dataType[size];
```

name – the name of the array
dataType – **`int`**, **`char`**, **`double`**, **`String`**, etc...

new – tells Java to create space in memory

size – the number of items, or elements, in array

Example – Passing a Specific Array to a Function

```
public static void printArray (int[] array) {
    for int i = 0; i < array.length; i++) {
        System.out.println(array[i]);
    }
}

public static void main(String[] args) {
    int[] numbers = new int[10];

    % initialize the array
    for int i = 0; i < numbers.length; i++) {
        numbers[i] = 2*i – 3;
    }

    printArray(numbers);
}
```

Changing Data using Methods

When a primitive data type (e.g., int or double) is passed to a method, only a copy of the data is provided. The original data cannot be modified.

```
int number = 5;

System.out.println(number); // output is 5
increaseByOne(number); // adds 1 to parameter
System.out.println(number); // output is still 5
```

Changing Data using Methods

When an array is passed to a method, a copy of the address of the array (i.e., the location of the data in memory) is made. The actual data, however, is referenced using this address, so it is not protected and may be changed.

```
int[] numbers = {1, 2, 3, 4, 5};

printArray(numbers); // output is 1, 2, 3, 4, 5
increaseByOne(numbers); // adds 1 to each element
printArray(numbers); // output is 2, 3, 4, 5, 6
```