Arrays: It is a data structure that allows you to store sequence of values all of same time.

We can instantiate a new array using new Keyword. We have array declaration on the left side of = array creation expression on the right side.

Array Creation

```
int[] integerArray = new int[10];
```

The array initializer

```
int[] firstFivePositives = {1, 2, 3, 4, 5};
String[] names = {"Andy", "Bob", "Charlie", "David", "Eve"};
```

- You cannot change the size of an array once the array is instantiated.
- We can't add or delete elements. We can only assign values to one of the elements assigned in the array.
- In the example we use new keyword to create an array. And specified the length of the array as 10.

Array initializer as an anonymous array:

- Above is an example of anonymous array initializer which can be only used in declaration statement. Here the size of array is determined by the number of elements specified in {}
- We cannot declare an array in one line And initialize the array elements in another line without specifying the size of an array. However, we can Initialize an array as an anonymous array in a single line.

Array is a special class in Java which inherits from java.lang.Object. Java array type is very basic and it comes with limited built in functionality. However, Java provides a helper class named Java.Util.Arrays providing all the common functionalities and some static methods on arrays which can be used.

Array initialization to default values –

- For primitive types, this is zero for any kind of numeric primitive, like int, double or short.
- For booleans, the default value will be false.
- And for any class type, the elements will be initialized to null.

Reference types versus value types –

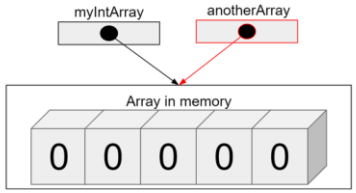
When you assign an object to a variable, the variable becomes reference to that object.

```
int[] myIntArray = new int[5];
int[] anotherArray = myIntArray;

System.out.println("myIntArray= " + Arrays.toString(myIntArray));
System.out.println("anotherArray= " + Arrays.toString(anotherArray));

anotherArray[0] = 1;

System.out.println("after change myIntArray= " + Arrays.toString(myIntArray));
System.out.println("after change anotherArray= " + Arrays.toString(anotherArray));
```



- Here we have instantiated an array called myIntArray using new keyword which is of length 5 elements.
- Here the variables myIntArray and anotherArray are referencing to the same array in memory.

Variable arguments. - We can replace the brackets after the String type in the main function with three periods. This is a special designation in Java that means Java will either take 0 or 1 or many strings as arguments to this method.

However, we need to remember 2 important concepts while using variable arguments.

- There can be only one variable argument in a method.
- The variable argument must be the last argument.

```arduino
Example.java:4: error: varargs parameter must be the last parameter
    public void printInfo(String message, int... numbers, String... names) {
                                                          ^
1 error
```

The above code throws an error because there is more than one variable argument as a parameter in the method.

```java
public void printInfo(String message, String[] names, int... numbers) {
    // Method body
}
```

We can resolve the error by having only one variable argument in the method as a parameter. And that parameter will be the last argument.

Two-dimensional and multi-dimensional arrays : Java supports 2 dimensional and three-dimensional arrays of varying dimensions using a concept called nested array.

We can also initialize the two-dimensional array without specifying the size of nested arrays.

```
int[][] myDoubleArray;
int[] myDoubleArray[];
```

- These are the two different ways of initializing a 2-dimensional array. However, first method is preferred over the second to avoid confusion.

In this example, we have an outer array with three elements.

```
Object[] multiArray = new Object[3];
multiArray[0] = new Dog[3];
multiArray[1] = new Dog[3][];
multiArray[2] = new Dog[3][][];
```

The first element is itself a single-dimension array.

The second element is a two-dimensional array.

And lastly, the third element is a three-dimensional array.