# Arrays



* An array is an indexed collection of fixed number of homogenous data elements.
* Elements can be accessed by index.
* Index starts from 0 and ends with size-1
* Advantage:
  + We can store multiple values of same type with single variable by which readability gets improved.
* Disadvantage:
  + It can’t be scaled dynamically as it is fixed in size after creation. This problem can be resolved with the help of collections.

**Types of Arrays:**

* Arrays are of two types:
  + Single dimensional arrays
  + Multi-Dimensional arrays

**Array Declarations:**

* *Single dimensional array declaration:* 
  + **int[] a**
  + **int []a**
  + **int a[]**
* We can’t specify the size otherwise we will get compile time error.
* Int a[5] – invalid
* *Two dimensional array declaration:*
  + **int[][] a**
  + **int a[][]**
  + **int [][]a**
  + **int[] a[]**
  + **int[] []a**
  + **int []a[]**
* If you want to specify dimension just before variable then it’s applied only for first variable.
* Eg :
  + **int[] a1,b1; - valid ( a1 is 1-dimensional, b1 is 1-dimensional)**
  + **int[] a2[],b2; - valid ( a2 is 2-dimensional, b2 is 1-dimensional)**
  + **int[] []a3,b3; - valid ( a3 is 2-dimensional, b3 is 2-dimensional)**
  + **int[] a3,[]b3; - invalid (Compilation Error)**
* Declarations of other types:
  + int[] integerArray;
  + short[] shortArray;
  + byte[] byteArray;
  + long[] longArray;
  + float[] floatArray;
  + double[] doubleArray;
  + char[] charArray;
  + String[] stringArray;

**Array Construction:**

* Every array in java is an object. Needs to be constructed using ***new*** operator.
* Eg: int[] a = new int[10];
* At compile time, size needs to be specified or else compile time error will be thrown stating *array dimension missing.*
* It’s legal to have ‘0’ as array size.
* If array size is taken as negative, then runtime error will be thrown stating *NegativeArraySizeException*.
* Array size should be of byte, short, char, int. Any other type usage will throw error.
* Max array size is 2147483647. If exceeds this then *OutOfMemoryException* will be thrown.
* Main advantage of multidimensional array is memory utilization and readability.
* **int[] a=new int[]//C.E: array dimension missing(invalid)**
* **int[][] a=new int[3][4];(valid)**
* **int[][] a=new int[3][];(valid)**
* **int[][] a=new int[][4];//C.E:']' expected(invalid)**
* **int[][][] a=new int[3][4][5];(valid)**
* **int[][][] a=new int[3][4][];(valid)**
* **int[][][] a=new int[3][][5];//C.E:']' expected(invalid)**

**Array Initialization:**

* When we create an array then it is created by default values.
* When we try to print array by variable name then it prints className@HexaDecimalHashCodeRep
* If you need to reinitialize arrays, then do this as follows *variableName[index] = value*
* **Eg:** int[] a= new int[10]; a[2] = 10;
* If you try to access value at index >= size, then you will get run time exception stating *ArrayIndexOutOfBoundsException*
* Arrays can be initialized in a single line without specifying size as shown below.
* Eg: int[] wholeNumbersUptoTen = {0,1,2,3,4,5,6,7,8,9}

**Array Length:**

* Arrays have a *length* as a final variable which returns length of the array.
* Eg: int[] a = new int[10]; a.length;// returns 10
* In multidimensional arrays, it returns base size instead of total size.
* Eg: int[][] a = new int[10][20]; a.length; //returns 10
* Eg: int[][] a = new int[10][20]; a[0].length; //returns 20
* *length* variable is applicable only for arrays whereas *length()* is applicable for Strings.

**Anonymous Arrays:**

* Sometimes we create arrays without name. Such arrays are called as Anonymous arrays.
* Main objective is for instant use.
* Eg: new int[]{1,2,3,4};
* If we specify size, then we get compile time error stating ‘;’ expected.