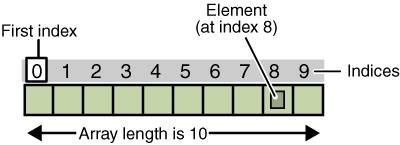
**2.9 Arrays in Java :-**

An *array* is a container object that holds a fixed number of values of a single type. The length of an array is established when the array is created. After creation, its length is fixed. You've seen an example of arrays already, in the main method of the "Hello World!" application. This section discusses arrays in greater detail.



An array of ten elements

Each item in an array is called an *element*, and each element is accessed by its numerical *index*. As shown in the above illustration, numbering begins with 0. The 3rd element, for example, would therefore be accessed at index 2.

The following program, [ArrayDemo](http://download.oracle.com/javase/tutorial/java/nutsandbolts/examples/ArrayDemo.java), creates an array of integers, puts some values in it, and prints each value to standard output.

class ArrayDemo {

public static void main(String[] args) {

int[] anArray; // declares an array of integers

anArray = new int[4]; // allocates memory for 4 integers

anArray[0] = 100; // initialize first element

anArray[1] = 200; // initialize second element

anArray[2] = 300; // etc.

anArray[3] = 400;

System.out.println("Element at index 0: " + anArray[0]);

System.out.println("Element at index 1: " + anArray[1]);

System.out.println("Element at index 2: " + anArray[2]);

System.out.println("Element at index 3: " + anArray[3]);

Element at index 0: 100

Element at index 1: 200

Element at index 2: 300

Element at index 3: 400

.

**Declaring a Variable to Refer to an Array**

The above program declares anArray with the following line of code:

int[] anArray; // declares an array of integers

Like declarations for variables of other types, an array declaration has two components: the array's type and the array's name. An array's type is written as *type*[], where *type* is the data type of the contained elements; the square brackets are special symbols indicating that this variable holds an array. The size of the array is not part of its type (which is why the brackets are empty). An array's name can be anything you want, provided that it follows the rules and conventions as previously discussed in the [naming](http://download.oracle.com/javase/tutorial/java/nutsandbolts/variables.html#naming) section. As with variables of other types, the declaration does not actually create an array — it simply tells the compiler that this variable will hold an array of the specified type.

Similarly, you can declare arrays of other types:

byte[] anArrayOfBytes;

short[] anArrayOfShorts;

long[] anArrayOfLongs;

float[] anArrayOfFloats;

double[] anArrayOfDoubles;

boolean[] anArrayOfBooleans;

char[] anArrayOfChars;

String[] anArrayOfStrings;

You can also place the square brackets after the array's name:

float anArrayOfFloats[]; // this form is discouraged

However, convention discourages this form; the brackets identify the array type and should appear with the type designation.

**Creating, Initializing, and Accessing an Array**

One way to create an array is with the new operator. The next statement in the ArrayDemo program allocates an array with enough memory for ten integer elements and assigns the array to the anArray variable.

anArray = new int[10]; // create an array of integers

If this statement were missing, the compiler would print an error like the following, and compilation would fail:

ArrayDemo.java:4: Variable anArray may not have been initialized.

The next few lines assign values to each element of the array:

anArray[0] = 100; // initialize first element

anArray[1] = 200; // initialize second element

anArray[2] = 300; // etc.

Each array element is accessed by its numerical index:

System.out.println("Element 1 at index 0: " + anArray[0]);

System.out.println("Element 2 at index 1: " + anArray[1]);

System.out.println("Element 3 at index 2: " + anArray[2]);

Alternatively, you can use the shortcut syntax to create and initialize an array:

int[] anArray = {100, 200, 300, 400, 500, 600, 700, 800, 900, 1000};

Here the length of the array is determined by the number of values provided between *{* and *}*.

You can also declare an array of arrays (also known as a *multidimensional* array) by using two or more sets of square brackets, such as String[][] names. Each element, therefore, must be accessed by a corresponding number of index values.

In the Java programming language, a multidimensional array is simply an array whose components are themselves arrays. This is unlike arrays in C or Fortran. A consequence of this is that the rows are allowed to vary in length, as shown in the following [MultiDimArrayDemo](http://download.oracle.com/javase/tutorial/java/nutsandbolts/examples/MultiDimArrayDemo.java) program:

class MultiDimArrayDemo {

public static void main(String[] args) {

String[][] names = {{"Mr. ", "Mrs. ", "Ms. "},

{"Smith", "Jones"}};

System.out.println(names[0][0] + names[1][0]); //Mr. Smith

System.out.println(names[0][2] + names[1][1]); //Ms. Jones

}

}

The output from this program is:

Mr. Smith

Ms. Jones

Finally, you can use the built-in length property to determine the size of any array. The code

System.out.println(anArray.length);

will print the array's size to standard output.

**Copying Arrays**

The System class has an arraycopy method that you can use to efficiently copy data from one array into another:

public static void arraycopy(Object src,

int srcPos,

Object dest,

int destPos,

int length)

The two Object arguments specify the array to copy *from* and the array to copy *to*. The three int arguments specify the starting position in the source array, the starting position in the destination array, and the number of array elements to copy.

The following program, [ArrayCopyDemo](http://download.oracle.com/javase/tutorial/java/nutsandbolts/examples/ArrayCopyDemo.java), declares an array of char elements, spelling the word "decaffeinated". It uses arraycopy to copy a subsequence of array components into a second array: class ArrayCopyDemo {

public static void main(String[] args) {

char[] copyFrom = { 'd', 'e', 'c', 'a', 'f', 'f', 'e',

'i', 'n', 'a', 't', 'e', 'd' };

char[] copyTo = new char[7];

System.arraycopy(copyFrom, 2, copyTo, 0, 7);

System.out.println(new String(copyTo)); }}

**Definition of One Dimensional Array**

**One dimensional array is a list of variables of same type that are accessed by a common name.** An individual variable in the array is called  an array element. Arrays forms a way to handle groups of related data.

**Syntax of Multi dimensional Arrays**

**To create an array, you need to perform two steps :**

**1) Declare the array :**

To declare an array below is a general form or syntax.

*type varName[];*

Where **type** is valid data type in java and **varName** is a name of an array.

**Example :      int iarr[];**

**2) Allocate space for its elements :**

To allocate space for an array element use below general form or syntax.

*varName = new type[size];*

Where **varName** is the name of the array and **type** is a valid java type and **size** specifies the number of elements in the array.

**Example :      iarr = new int[10];**

Above statement will create an integer of an array with ten elements that can by accessed by iarr.

**Structure of one dimensional array**

|  |
| --- |
| iarr[0] |
| iarr[1] |
| iarr[2] |
| iarr[3] |
| iarr[4] |
| iarr[5] |
| iarr[6] |
| iarr[7] |
| iarr[8] |
| iarr[9] |

Note that array indexes begins with zero. That means if you want to access 1st element of an array use zero as an index. To access 3rd element refer below example.

iarr[2] = 10; // it assigns value 10 to the third element of an array.

java also allows an **abbreviated syntax** to declare an array. General form or syntax of is is as shown below.

type varName[] = {v0, v1, v2 ........ v10};

Where varName is the name of the array and type is a valid data type and v0, v1 to v10 are the elements of an array. In this case new is not used and memory for the array is automatically provided.

**Example :**    **int iarr = {1,2,3,4,5,6,7,8,9,10};**

**Example of One dimensional array**

**Example : This example shows how to declare initialize  and display an array.**

// **Declaration of allocating memory to an array**

int iarr[]  = new int[3];

// **Initializing elements**

iarr[0] = 1;

iarr[1] = 2;

iarr[2] = 3;

//**Display array elements**

System.out.println(iarr[0]);

System.out.println(iarr[1]);

System.out.println(iarr[2]);

// **Or Use for loop to display elements**

 for (int i = 0; i < iarr.length; i = i + 1)

{

       System.out.print(iarr[i]);

       System.out.print(" ");

 }