1. One and Two Dimensional Arrays :: Declaring Arrays

The syntax for an **array declaration** is:

```
T[] name = new T[length]
```

where T is the data type of each element of the array, *name* is the name of the array variable being declared, and *length* is the number of elements in the array. *length* may be an integer literal, an integer constant, an integer variable, or an expression that evaluates to an integer. Examples,

```
static final int L = 100;
int x = 10;

// a is an array of 5 ints (5 is an integer literal).
int[] a = new int[5];

// b is an array of 10 doubles (x is an integer variable).
double[] b = new double[x];

// c is an array of 100 String objects (L is an integer constant).
String[] c = new String[L];

// d is an array of 19 booleans (2 * x - 1 is an arithmetic expression that
// evaluates to an integer.
boolean[] d = new boolean[2 * x - 1];
```

1. One and Two Dimensional Arrays :: Initializing the Elements of an Array

The elements of the array may be **initialized** at the time the array is declared:

```
int[] a = { 10, 20, 30, 40, 50 };
```

Note that we do not use the **new** operator when declaring and initializing an array at the same time.

1. One and Two Dimensional Arrays :: Accessing Array Elements

Remember that the elements of an array are accessed using the **array subscript operator [ ]** with an integer **subscript** or **index** inside the brackets. The indices (or subscripts) are numbered starting at 0 and *arrayname*.length evaluates to the number of elements in *arrayname*.

```
static final int Z = 4;
int x = 2;
int a[] = { 10, 20, 30, 40, 50 };

// Change a[0] to 30; a is now { 30, 20, 30, 40, 50 }.
a[0] = (a[0] + a.length - 1) / 2;

// Change a[2] to 29; a is now { 30, 20, 29, 40, 50 }.
--a[x];

// Change a[3] to a[2]; a is now { 30, 20, 29, 29, 50 }.
a[Z-1] = a[Z - x];
```

1. One and Two Dimensional Arrays :: The Enhanced For Loop

To iterate through the elements of an array we can write a **for loop**:

```
for (int i = 0; i < a.length; ++i) {
  System.out.println(a[i]);
}
```

Because this is such a common operation we can also do this using the **enhanced for loop**. The syntax is:

```
for (T value : arrayname) {
  statements
}
```

where ***T*** is the data type of the elements of the array *arrayname*. During the first pass of the loop *value* will be *arrayname*[0], on the second pass it will be *arrayname*[1], on the third pass it will be *arrayname*[2], and so on until the final pass when *value* will be *arrayname*[*arrayname*.length].