JAVA ARRAYS

An ***array*** is a data abstraction representing a list of items. The basic characteristics of an array are:

- All items in the array have the same data type, called the ***component data type***.
- The items are stored consecutively within the computer's physical memory.
- Each item is numbered sequentially according to its position in the list, starting with 0.

We call each item in the array an ***element*** (or ***entry***) and the number of elements in the array its ***length***. The position of an element's item within the list is often called its ***index***.

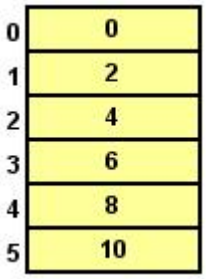
The numbering of items in the list begins with 0 – the first item is numbered **`0`**, the second item **`1`**, the third **`2`**, and so on until the last, which is numbered *array length* – 1.

To declare and initialize an array in one Java statement, you can use the syntax:

component data type **`[]`** *array identifier* **`= {`** *value list* **`};`**

Examples

```
int [] even = {0, 2, 4, 6, 8, 10};
```



`even` is an array of component data type **`int`** and length of 6.
Position 0 of **`even`** contains the value 0.
Position 1 contains 2.
Position 2 contains 4.
etc.

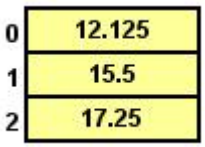
```
char [] vowel = {'a','e','i','o','u'};
```

`vowel` is an array of component data type **`char`** and length of 5.
Position 0 of **`vowel`** contains the value **`'a'`**.
Position 1 contains **`'e'`**.
Position 2 contains **`'i'`**.
etc.

Examples

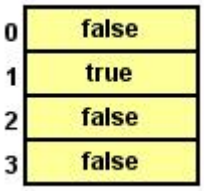
```
double [] v = {12.125, 15.500, 17.250};
```

v is an array of component data type **double** and length of 3.
Position 0 of **v** contains 12.125.
Position 1 contains 15.5.
Position 2 contains 17.25.




```
boolean [] bit = {false, true, false, false};
```

bit is an array of component data type **boolean** and length of 4.
Position **0** of **bit** contains **false**.
Position 1 contains **true**.
Position 2 contains **false**.
etc.



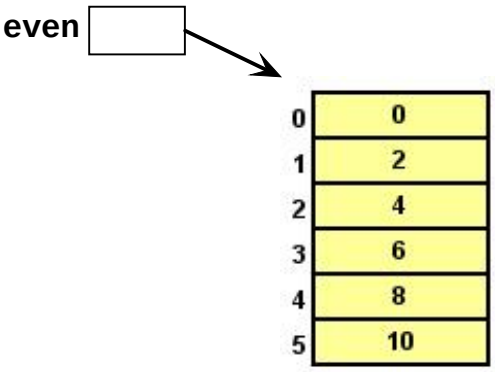
Arrays are Objects

A Java array is an object that must be accessed through a reference variable.

Example

A conceptual picture of memory as allocated by the declaration and initialization of the array **even**.

```
int [] even = {0, 2, 4, 6, 8, 10};
```



Building an Array

To build an array, you must, as with any object, first declare a reference variable through which to access it. The syntax is:

component data type **[]** *array identifier*

Example

Line 1 below declares **ary** to be a reference to an array of **double** values. After this statement is executed, **ary** is allocated but not initialized, so its value is **null**.


```
1  double [] ary;
```

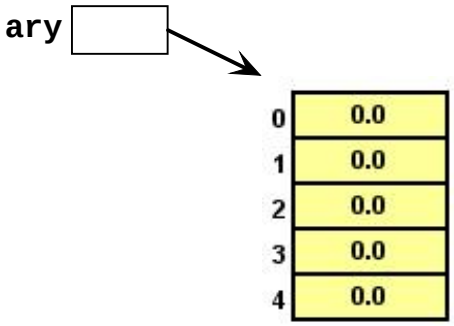
Second, you build the array by using Java's **new** operator. The syntax is:

new *component data type* **[** *length* **]**

length is the number of elements to be allocated for the array; they are numbered 0 to *length*−1.

Example

Line 2 below allocates the 5-element **double** array and places its memory address into the reference variable **ary**. After this statement is executed, **ary** has been initialized but the 5 **double** elements have not.




```
1  double [] ary;
2  ary = new double[5];
```

Subscripting an Array

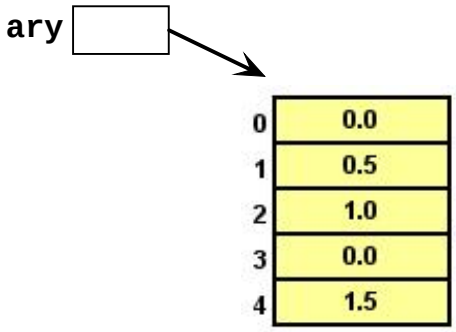
Once an array is built and its reference established, you can individually refer to each element by ***subscripting*** the array. The syntax is:

array identifier **[** *index value* **]**

A subscripted array is treated as a variable with all the properties of the array's component data type.

Example

Lines 3-6 initialize the array elements as shown in the picture at right.




```
1  double [] ary;
2  ary = new double[5];
3  ary[1] = ary[0] + 0.5;
4  ary[2] = ary[1] * 2;
5  ary[3] = ary[0];
6  ary[4] = ary[1] + ary[2];
```

Shorthand Array Declaration and Initialization

As with any reference variable, the declaration and initialization of an array reference can be combined into one statement.

Example

```
double [] ary = new double[5];
```

And, as we've already seen at the beginning of this topic, the initialization of the array can be combined into it as well.

Example

```
double [] ary = { 0, 0.5, 1, 0, 1.5 };
```

Array Length

Every array has a public field named **`length`** that gives you the number of elements that are allocated to the array.

Example

The array's length, used to calculate and print the average of the values in the array, is referenced in lines 3 and 4 of this code fragment. The output is:

```
45.0/3 = 15.0
```

```
1  double [] v = {10.0, 20.0, 15.0};
2  double sum, avg;
3  sum = v[0] + v[1] + v[2];
4  avg = sum / v.length;
5  System.out.println( sum + "/" + v.length + " = " + avg );
```

You can build an array with a *length* value that is calculated at run-time.

Example

The number of elements in array **`v`** is determined by the value input into variable **`count`**.

```
1  Scanner scanner = new Scanner( System.in );
2  System.out.print( "How many values? " );
3  int count = scanner.nextInt( );
4  double [] v = new double[count];
```

Exercises

For each, draw a picture of the allocated memory.

1.
```
int [] a = { 5, 4, 3, 2, 1, 0 };
```

2.
```
char [] b = { '%', '$', '#', '@', '&' };
```

3.
```
double [] c = { 0.5, 0.1, 0.15, 0.2, 0.25, 0.3 };
```

4.
```
int [] v;
```

5.
```
int [] v = new int[5];
```

6.
```
int [] v = new int[5];
v[0] = 1;
v[2] = v[0]*3;
v[v[2]] = 4;
v[1] = v[v[0]+1];
v[4] = v[v[0]]+1;
```

7.
```
char [] punctuation = new char[5];
punctuation[0] = '.';
punctuation[1] = ';';
punctuation[2] = ',';
punctuation[3] = '!';
punctuation[4] = '?';
```

For each, draw a picture of the allocated memory assuming that user enters 5.

8.
```
Scanner scanner = new Scanner( System.in );
int len = scanner.nextInt( );
double [] myArray = new double[len];
```

9.
```
Scanner scanner = new Scanner( System.in );
int a = scanner.nextInt( );
int [] vector = { a*1, a*2, a*3, a*4, a*5 };
```

For each, assume this array and give the output.

```
int [] c = { 5, 4, 3, 2, 1, 0 };
```

10.	`System.out.println( c[0] );`
11.	`System.out.println( c[1] );`
12.	`System.out.println( c[1]*3 );`
13.	`System.out.println( c[1*3] );`
14.	`System.out.println( c[c[1]] );`
15.	`System.out.println( c[c[1]]-2 );`
16.	`System.out.println( c[c[1]-2] );`
17.	`System.out.println( c.length );`

18.	Write the Java statement to declare and initialize an array containing 5 zeros. Make it a **`double`** array.
19.	Write the Java statement to declare and initialize an array of single character values containing 's', 'p', 'e', 'l' and 'l'.
20.	Write the Java statement to declare and initialize an array of containing 5 **`int`** values, each initialized to -1.
21.	Given: **`int [] v = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };`** Write the Java statements to swap the values of **`v[1]`** and **`v[8]`**.