ARRAY INITIALIZATION

For an array variable, just as for any variable, you can declare the variable and initialize it in a single step. For example,

```
int[] list = new int[5];
```

If `list` is a local variable in a subroutine, then this is exactly equivalent to the two statements:

```
int[] list;
list = new int[5];
```

(If `list` is an instance variable, then of course you can't simply replace "`int[] list = new int[5];`" with "`int[] list; list = new int[5];`" since the assignment statement "`list = new int[5];`" is only legal inside a subroutine.)

The new array is filled with the default value appropriate for the base type of the array -- zero for int and `null` for class types, for example. However, Java also provides a way to initialize an array variable with a new array filled with a specified list of values. In a declaration statement that creates a new array, this is done with an array initializer. For example,

```
int[] list = { 1, 4, 9, 16, 25, 36, 49 };
```

creates a new array containing the seven values 1, 4, 9, 16, 25, 36, and 49, and sets `list` to refer to that new array. The value of `list[0]` will be 1, the value of `list[1]` will be 4, and so forth. The length of `list` is seven, since seven values are provided in the initializer.

An array initializer takes the form of a list of values, separated by commas and enclosed between braces. The length of the array does not have to be specified, because it is implicit in the list of values. The items in an array initializer don't have to be constants. They can be variables or arbitrary expressions, provided that their values are of the appropriate type. For example, the following declaration creates an array of eight *Colors*. Some of the colors are given by expressions of the form "`new Color(r,g,b)`" instead of by constants:

```
Color[] palette = {
                Color.BLACK,
                Color.RED,
                Color.PINK,
                new Color(0,180,0),  // dark green
                Color.GREEN,
                Color.BLUE,
                new Color(180,180,255),  // light blue
                Color.WHITE
              };
```

A list initializer of this form can be used **only** in a declaration statement, to give an initial value to a newly declared array variable. It cannot be used in an assignment statement to assign a value to a variable that has been previously declared. However, there is another, similar notation for creating a new array that can be used in an assignment statement or passed as a parameter to a subroutine. The notation uses another form of the `new` operator to both create and initialize a new array object at the same time. (The rather odd syntax is similar to the syntax for anonymous classes, which were discussed in Subsection 5.7.3.) For example to assign a new value to an array variable, `list`, that was declared previously, you could use:

```
list = new int[] { 1, 8, 27, 64, 125, 216, 343 };
```

The general syntax for this form of the `new` operator is

```
new base-type [ ] { list-of-values }
```

This is actually an expression whose value is a reference to a newly created array object. This means that it can be used in any context where an object of type **base-type**`[]` is expected. For example, if`makeButtons` is a method that takes an array of *Strings* as a parameter, you could say:

```
makeButtons( new String[] { "Stop", "Go", "Next", "Previous" }
);
```

Being able to create and use an array "in place" in this way can be very convenient, in the same way that anonymous nested classes are convenient.

By the way, it is perfectly legal to use the "`new BaseType[] { ... }`" syntax instead of the array initializer syntax in the declaration of an array variable. For example, instead of saying:

```
int[] primes = { 2, 3, 5, 7, 11, 13, 17, 19 };
```

you can say, equivalently,

```
int[] primes = new int[] { 2, 3, 5, 7, 11, 17, 19 };
```

In fact, rather than use a special notation that works only in the context of declaration statements, I prefer to use the second form.

One final note: For historical reasons, an array declaration such as

```
int[] list;
```

can also be written as

```
int list[];
```

which is a syntax used in the languages C and C++. However, this alternative syntax does not really make much sense in the context of Java, and it is probably best avoided. After all, the intent is to declare a variable of a certain type, and the name of that type is "`int[]`". It makes sense to follow the "**type-name variable-name**;" syntax for such declarations.

Source : http://math.hws.edu/javanotes/c7/s1.html