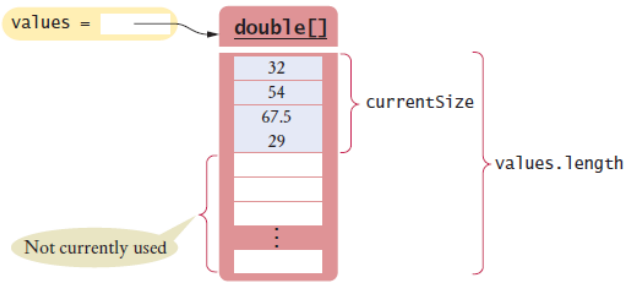
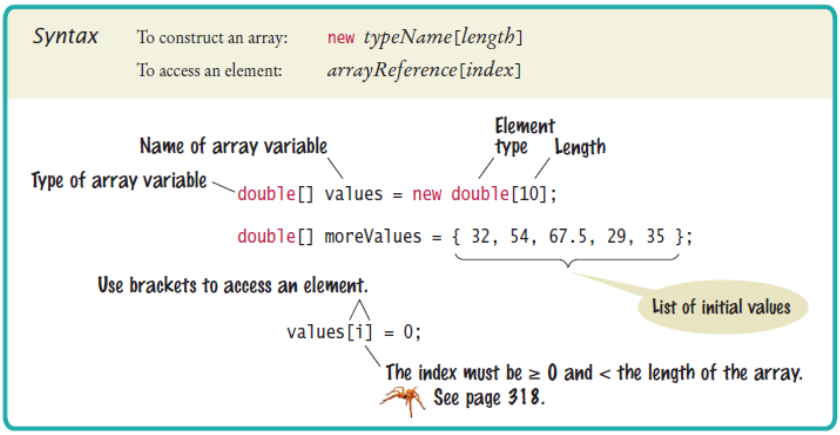
- As with arrays, you need to remember that array list variables hold references. Copying the reference yields two references to the same array list
- When you supply initial values, you don't use the new operator. The compiler determines the length of the array by counting the initial values.
- You have to be careful that the index stays within the valid range. Trying to access an element that does not exist in the array is a serious error. For example, if values has ten elements, you are not allowed to access values[20]



- Arrays suffer from a significant limitation: their length is fixed. If you start out with an array of 10 elements and later decide that you need to add additional elements, then you need to make a new array and copy all elements of the existing array into the new array.
- Arrays can be method arguments and return values, just like any other values.
- When you define a method with an array argument, you provide a parameter variable for the array.
- When you create a method with an array argument, you GIVE a parameter variable for the array. An array's size cannot be changed while it is running. This is a problem when you don't know how many elements you'll need ahead of time. In that case, you must make an educated judgment as to the maximum amount of items that must be stored.



- Array variables function precisely like object variables in that they are merely pointers to the array itself. You must use the new operator to create the real array: new double[10] = double[] values
- These kinds of arrays are referred to as direct arrays. In Java, you may get a two-dimensional array by specifying the rows and columns. For example, new int[7][3] is a seven-row, three-column array. A reference to such an array is stored in an int[][property.