Arrays – Quick Notes

Concept

An array holds a fixed number of things, every one of the same type. Each of the things can be accessed using an index number starting at zero and running up to the number of things in the array minus 1. The index number provides an ordering to the things in the array. Each of the things can be accessed randomly, anytime, using its index. The things in the array are often called the elements of the array. The entire array, including all the things in it, can be somethings be used as a whole (e.g., as a parameter to a function).

Declaration of an Array

The declaration tells the compiler the name of the array and what type things will be stored in it. It is one form of variable declaring. The declaration is information to define the name to the compiler. It does *not* set the number of things that can be in the array.

```
int[]      myNumbers;
Person[]   myFriends;
```

Instantiation of an Array

Instantiation or creation of the actual array fixes the number of things in it and assigns actual memory to the array. Instantiation takes place at run time and the size of the array can be any expression. Good style is to separate the declaration and the instantiation on different lines of code.

```
myNumbers1 = new int[10];
myNumbers2 = new int[(x * 2) % 5];  //x defined and has a value
myFriends = new Person[21];
```

Initialization of an Array

There are many ways to give an existing array (declared and instantiated) values. The compiler even does some initialization for you if you don't (but bad style to depend on that). Initializations are done at run time and are assignments to one or all of the array elements. Good style always uses all three parts of an instantiation combined with an initialization (even though some are optional – it prevents problems with changes).

```
myNumbers1 = new int[10] {0,1,2,3,4,5,6,7,8,9};

for(int i = 0, i<myNumbers.Length; i++) {
     myNumbers[i] = i*10 + i%2;
}
```

Accessing Elements of an Array

To access an element of the array use the name of the array variable and square brackets around an expression that will generate a valid index number. Elements accessed in this way may appear on the left side of assignments and anywhere in expressions. The foreach statement is useful to access all the elements of an array.

```
myNumbers1[3] = 42;
theAnswer = myNumbers[0] / myNumbers[x-2]; //assumes x exists 2<=x<12

foreach(Person p in myFriends) {
      Process ( p );  //Process a function doing something with people
}
```

Rules and Recommendations

At least for freshers or newbies, *please* (it will help you learn and prevent troublesome problems):

1. Array variables are plural nouns in camelCase.

2. Always separate declaration and instantiation of arrays onto separate lines. When initializing arrays with the {…..} form, still include new and the size of the array.

3. Don't use anonymous arrays, you will only confuse yourself and possibly waste memory.

   ```
   EatArray( new int[] {1,2,3});  //DON'T YOU DARE
   ```

4. Don't alias array variables. You are welcome to understand aliasing (where two different names reference the same array object), but don't ever do it on purpose. Such awful bugs can result!

   ```
   int[] x;
   int[] y;
   y = new int[3] {1,2,3};
   x = y;   //EGAD, now no one knows when one or the other may change
   ```

5. Never, ever, not even once, change a value of an array inside a function which received it as a parameter. When you really need to change two or more arrays inside one function, learn how to return a struct or create a class for the return type and return an object of that type.

Want More? (and follow the links from the second below)

Single-Dimensional Arrays (C# Programming Guide)
https://msdn.microsoft.com/en-us/library/0a7fscd0.aspx

Arrays (C# Programming Guide)
https://msdn.microsoft.com/en-us/library/9b9dty7d.aspx