Unsafe Code in .NET

Using unsafe with arrays

When accessing arrays with pointers, there are no bounds check and therefore no `IndexOutOfRangeException` will be thrown. This makes the code faster.

Assigning values to an array with a pointer:

```
class Program
{
    static void Main(string[] args)
    {
        unsafe
        {
            int[] array = new int[1000];
            fixed (int* ptr = array)
            {
                for (int i = 0; i < array.Length; i++)
                {
                    *(ptr+i) = i; //assigning the value with the pointer
                }
            }
        }
    }
}
```

While the safe and normal counterpart would be:

```
class Program
{
    static void Main(string[] args)
    {
        int[] array = new int[1000];
```

```
            {
                array[i] = i;
            }
        }
    }
```

The unsafe part will generally be faster and the difference in performance can vary depending on the complexity of the elements in the array as well as the logic applied to each one. Even though it may be faster, it should be used with care since it is harder to maintain and easier to break.

Using unsafe with strings

```
var s = "Hello";      // The string referenced by variable 's' is normally immutable, bu
                      // since it is memory, we could change it if we can access it in a
                      // unsafe way.

unsafe                // allows writing to memory; methods on System.String don't allow
{
  fixed (char* c = s) // get pointer to string originally stored in read only memory
    for (int i = 0; i < s.Length; i++)
      c[i] = 'a';     // change data in memory allocated for original string "Hello"
}
Console.WriteLine(s); // The variable 's' still refers to the same System.String
                      // value in memory, but the contents at that location were
                      // changed by the unsafe write above.
                      // Displays: "aaaaa"
```

Unsafe Array Index

```
void Main()
{
```

```
            int[] a = {1, 2, 3};
            fixed(int* b = a)
            {
                Console.WriteLine(b[4]);
            }
        }
    }
```

Running this code creates an array of length 3, but then tries to get the 5th item (index 4). On my machine, this printed `1910457872`, but the behavior is not defined.

Without the `unsafe` block, you cannot use pointers, and therefore cannot access values past the end of an array without causing an exception to be thrown.

Remarks

- In order to be able to use the `unsafe` keyword in a .Net project, you must check "Allow unsafe code" in Project Properties => Build
- Using unsafe code can improve performance, however, it is at the expense of code safety (hence the term `unsafe`).

For instance, when you use a for loop an array like so:

```
for (int i = 0; i < array.Length; i++)
{
    array[i] = 0;
}
```

.NET Framework ensures that you do not exceed the bounds of the array, throwing an `IndexOutOfRangeException` if the index exceeds the bounds.

However, if you use unsafe code, you may exceed the array's bounds like so:

```
unsafe
{
        {
            *(ptr+i) = 0;
        }
    }
}
```