Day 20 Assignment

By

Praveen Chakravarthi

19-02-2022

NB Health Care

1. Research and understand scope of variables in C#

The part of a Program where a particular variable can be Accessed is known as the **“Scope of that Variable”.**

In C#, The Scope of Variables is divided into 3 Categories:

- **Class Level Scope**
- **Method Level Scope**
- **Block Level Scope**

Class Level Scope:

- The Variables declared in the class(but outside the method) can be accessed anywhere within the class.
- It can be accessed by the non-static methods in the class
- Access Modifiers doesn’t affect the Class Level Scope Variables

Method level Scope :

- The variables that are declared inside a method is called Method level scoping and cannot be accessed outside the Method
- These methods can be accessed by the nested code blocks inside a method
- The variables doesn’t exisit after the method’s execution

Block Level Scope :

- The variables which are declared inside for, while statements etc are called Block Level Scope
- These variables are termed as Loop Variables as they limit their scope up to the body of the statement in which it is declared.
- A Variable declared inside a Loop will not be visible outside of loop body

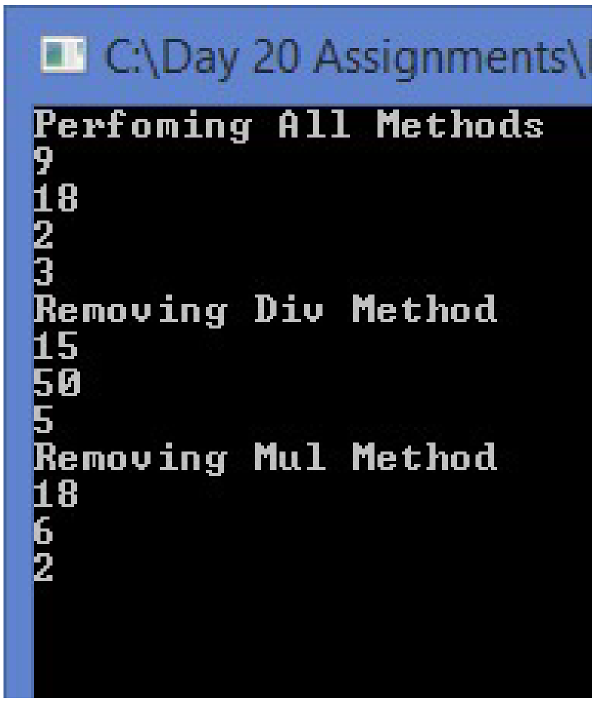
2. What are delegates in C#
Write the points discussed about delegates in the class
Write C# code to illustrate the usage of delegates.

- A delegate is like a function pointer.
- using delegates we can call or point to one or more methods
- when declaring a delegate, return type and parameters must match with the methods you want to point using the delegate.
- Benefit of Delegates :
- using single call from delegate, all your methods pointing to delegate can be called.

Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day_20_Project_1
{
    // Author : Praveen Chakravarthi
    // Purpose : Using Delegates in C#


    internal class Program
    {
        public delegate void Math(int a, int b);

        /// <summary>
        /// This Method Adds the given Numbers
        /// </summary>
        public static void Add(int a, int b)
            {
                Console.WriteLine(a + b);
            }
        /// <summary>
        /// This Method Multiplies the given Numbers
        /// </summary>
            public static void Mul(int a, int b)
            {
                Console.WriteLine(a * b);
            }
        /// <summary>
        /// This Method Divides the given Numbers
        /// </summary>
            public static void Div(int a, int b)
            {
                Console.WriteLine(a/b);
            }
        /// <summary>
        /// This Method Subtracts the given Numbers
        /// </summary>
            public static void Sub(int a, int b)
            {
                Console.WriteLine(a-b);
            }
        static void Main(string[] args)
        {
            // Creating Delegate Object and Initialising Add Method
            Math m = new Math(Add);
            // Adding Methods to the Delegate
            m += Mul;
            m += Div;
            m += Sub;

            // Perfoming all the Methods
            Console.WriteLine("Perfoming All Methods");
            m(6, 3);

            // Removing Div Method from the Delegate
            Console.WriteLine("Removing Div Method");
            m -= Div;
            m(10, 5);

            // Adding the Div Method, Removing Mul Method from the Delegate
            Console.WriteLine("Removing Mul Method");
            m += Div; m -= Mul;
            m(12, 6);

            Console.ReadLine();
        }
    }
}
```

Output :



```
Perfoming All Methods
9
18
2
3
Removing Div Method
15
50
5
Removing Mul Method
18
6
2
```

3. What are nullable types in C#
WACP to illustrate nullable types
Write some properties of nullable types (like HasValue)

As we know we cannot assign null value to Value Types

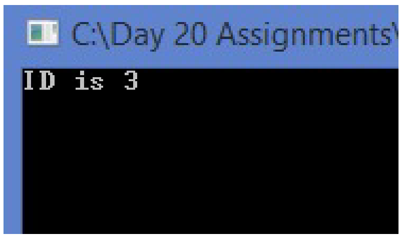
- C# introduced that allows us to assign null to Value Types using ‘?’ after the Value Type
- The usage of ‘?’ after the Value Type doesn’t affect it’s Range
- By using the property, such as HasValue, it returns true if the Value type is Assigned a value or returns false saying the value is ”null”

Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day_20_Project_2
{
    // Author : Praveen Chakravarthi
    // Purpose : Nullable Types in C#
    internal class Program
    {
        static void Main(string[] args)
        {
            int? ID = 3; // Declaring Nullable to Value Type using '?'

            if (ID.HasValue) // Checks if ID has a Value
                Console.WriteLine($"ID is {ID}");
            else
                Console.WriteLine("No Value");
            Console.ReadLine();
        }
    }
}
```

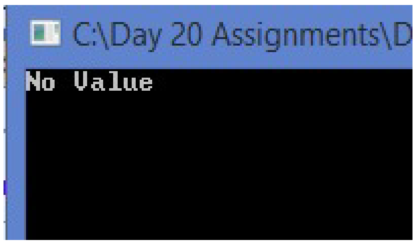
Output :

When ID has a Value



```
ID is 3
```

When ID is Null



```
No Value
```

4. out, ref- parameters
please research on these two types of parameters
write a C# program to illustrate the same

Out Parameter :

- Out variable must be initialised in the method itself
- Out is used when function return more than one value

Ref Parameter :

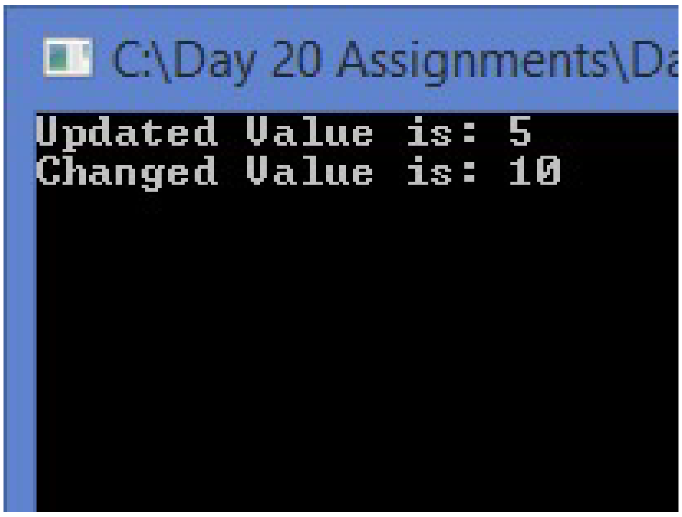
- Ref Variables must be initialised before passing method
- Ref is used to change the value in the called function and return it

We cannot use ‘ref’ and ‘out’ keyword with the same method name as both may be considered same at compile time.

Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day_20_Project_3
{
    // Author : Praveen Chakravarthi
    // Purpose : Usage of Out, Ref Parameters in C#
    public class Program
    {
        public static void Out(out int a)
        {
            a = 5;
        }
        public static void Ref(ref int b)
        {
            b = 10;
        }
        static void Main(string[] args)
        {
            int c;
            int d = 3;

            Out(out c); // Updating the 'c' value using out parameter
            Ref(ref d); // Changing the 'd' value using ref parameter

            Console.WriteLine($"Updated Value is: {c}");
            Console.WriteLine($"Changed Value is: {d}");
            Console.ReadLine();
        }
    }
}
```

Output :



```
Updated Value is: 5
Changed Value is: 10
```