Day 20 Assignment
By M Mary Margarette
On 18-02-2022

Research and understand scope of variables in C#

- Scope of the variable determines the accessibility of the variable to a particular part of the application.
- Scope of a variable can be determined at compile time.
- C# scope rules of variables can be divided into three categories as follows:
  - Class Level Scope
  - Method Level Scope
  - Block Level Scope

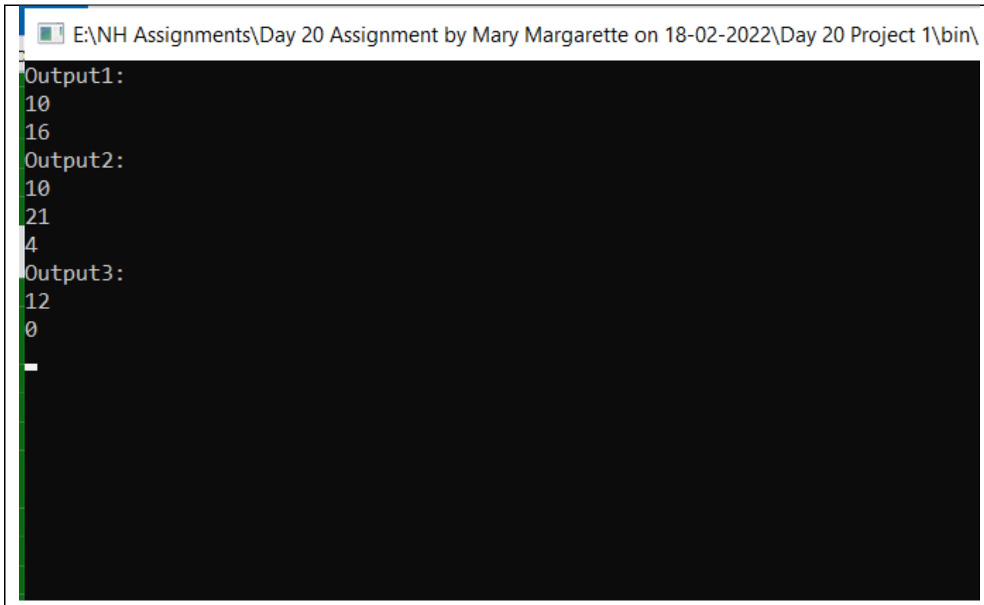
What are delegates in C#   Write the points dicussed about delegates in the class   Write C# code to illustrate the usage of delegates.

- Delegates contain the same return type and same parameters as in methods.
- Ex: public delegate void Mycaller (int a, int b)
  Mycaller me = new Mycaller (add);
  Mc + = Mul;
  Mc + = Div;
- Delegate is a function pointer.
- Using delegates, we can call our point to one or more methods.
- When declaring delegate, return & parameters should be same as delegate.
- Benefit of delegate is that using single call from delegate, all your methods pointing to delegate will be called.
- They are two types of Delegates
  1. Single cast
  2. Multi- cast
- Single cast will point to only one method.
- Multi –cast will point to more than one method.

Code:

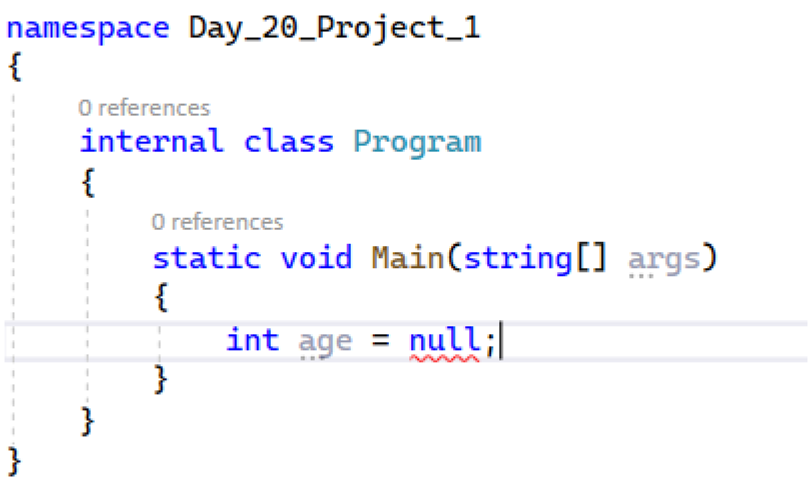
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day_20_Project_1
{
    internal class Program
    {
        public delegate void MyCaller(int a, int b);
        public static void Add(int a, int b)
        {
            Console.WriteLine(a + b);
        }
        public static void Sub(int a, int b)
        {
            Console.WriteLine(a - b);
        }
        public static void Mul(int a, int b)
        {
            Console.WriteLine(a * b);
        }
        static void Main(string[] args)
        {

            Console.WriteLine("Output1:");
            MyCaller mc = new MyCaller(Add);
            mc += Mul;
            mc(8,2);
            Console.WriteLine("Output2:");
            mc += Sub;
            mc(7,3);
            Console.WriteLine("Output3:");
            mc -= Mul;
            mc(6,6);
            Console.ReadLine();

        }
    }
}
```

Output:

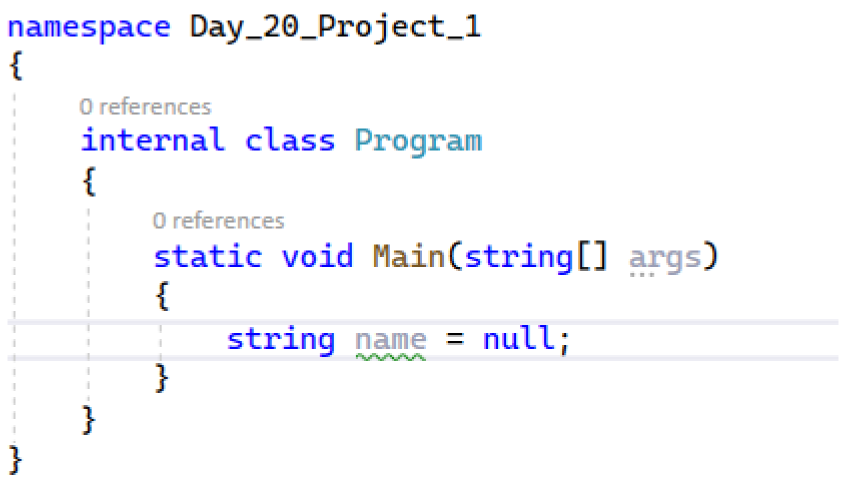


E:\NH Assignments\Day 20 Assignment by Mary Margarette on 18-02-2022\Day 20 Project 1\bin\

```
Output1:
10
16
Output2:
10
21
4
Output3:
12
0
```

What are nullable types in C# WACP to illustrate nullable typesWrite some properties of nullable types (like HasValue)
• A value type cannot be assigned a null value. It will give you a compile time error.
• It can be assigned only to string type.


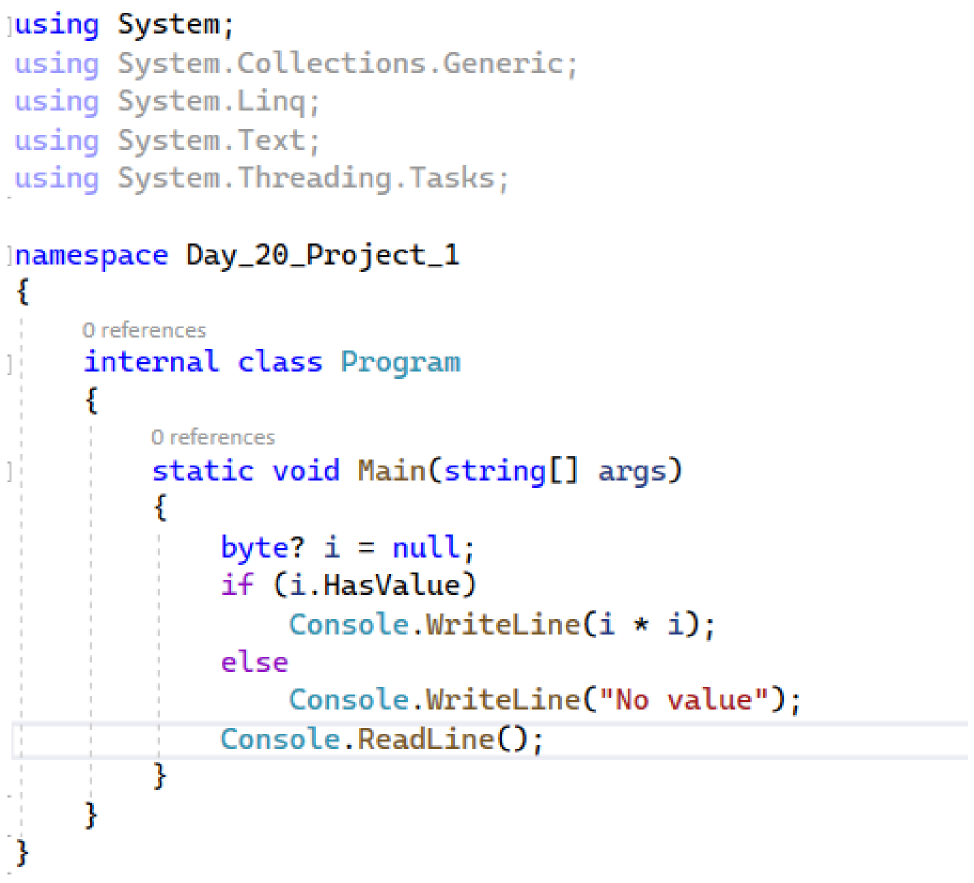
```
namespace Day_20_Project_1
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            int age = null;
        }
    }
}
```



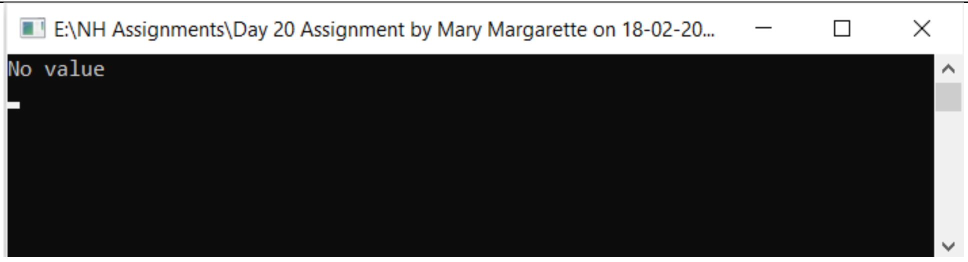
```
namespace Day_20_Project_1
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            string name = null;
        }
    }
}
```

Properties of Nullable types in C#:

- ❖ HasValue is one of the properties of Nullable types.
- ❖ Value is another property of Nullable types.

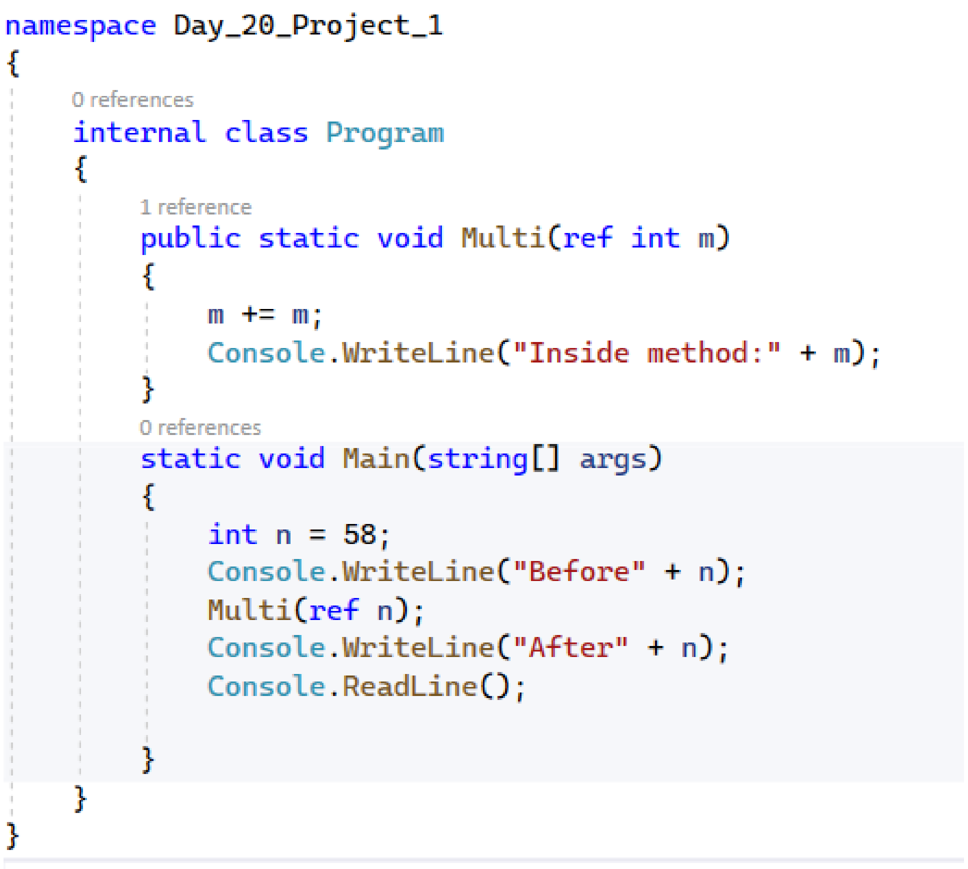


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day_20_Project_1
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            byte? i = null;
            if (i.HasValue)
                Console.WriteLine(i * i);
            else
                Console.WriteLine("No value");
            Console.ReadLine();
        }
    }
}
```

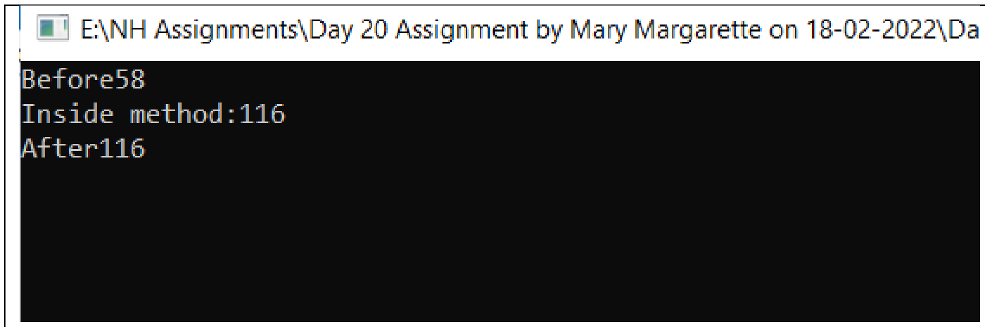


E:\NH Assignments\Day 20 Assignment by Mary Margarette on 18-02-20...

```
No value
```

out, ref - parametersplease research these two types of parameterswrite a C# program to illustrate the same.
Ref Parameter Code:

Output:

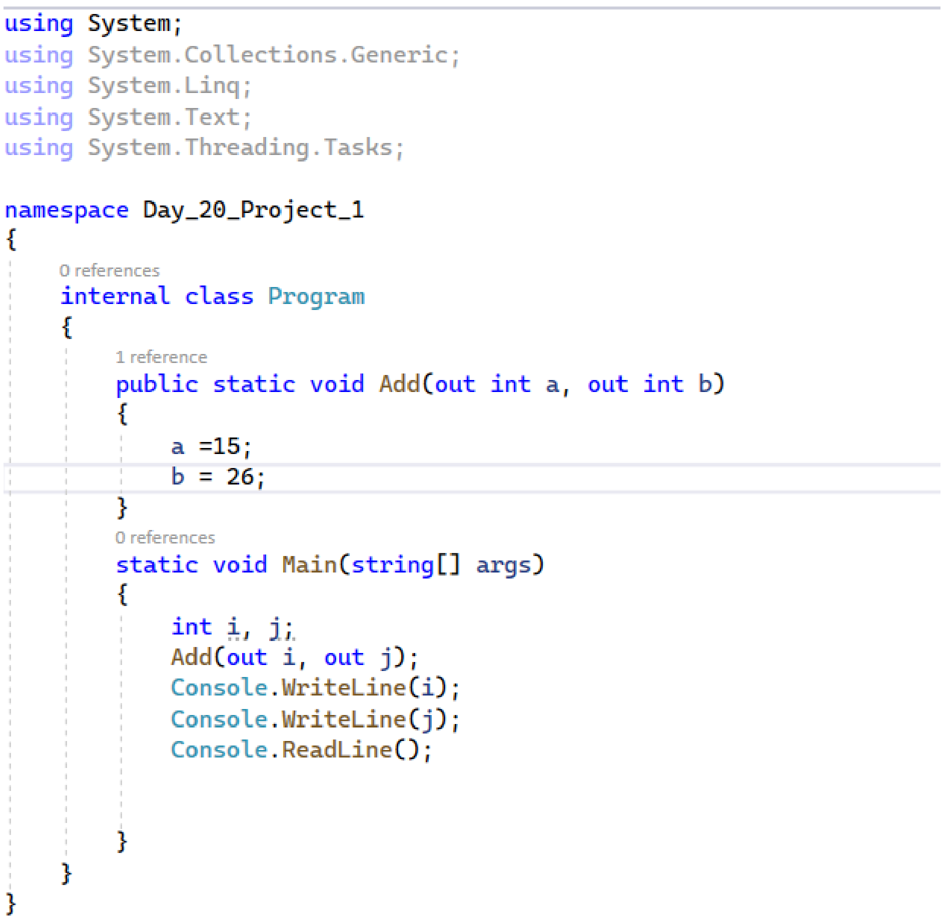
```
namespace Day_20_Project_1
{
    0 references
    internal class Program
    {
        1 reference
        public static void Multi(ref int m)
        {
            m += m;
            Console.WriteLine("Inside method:" + m);
        }
        0 references
        static void Main(string[] args)
        {
            int n = 58;
            Console.WriteLine("Before" + n);
            Multi(ref n);
            Console.WriteLine("After" + n);
            Console.ReadLine();

        }
    }
}
```



E:\NH Assignments\Day 20 Assignment by Mary Margarette on 18-02-2022\Da

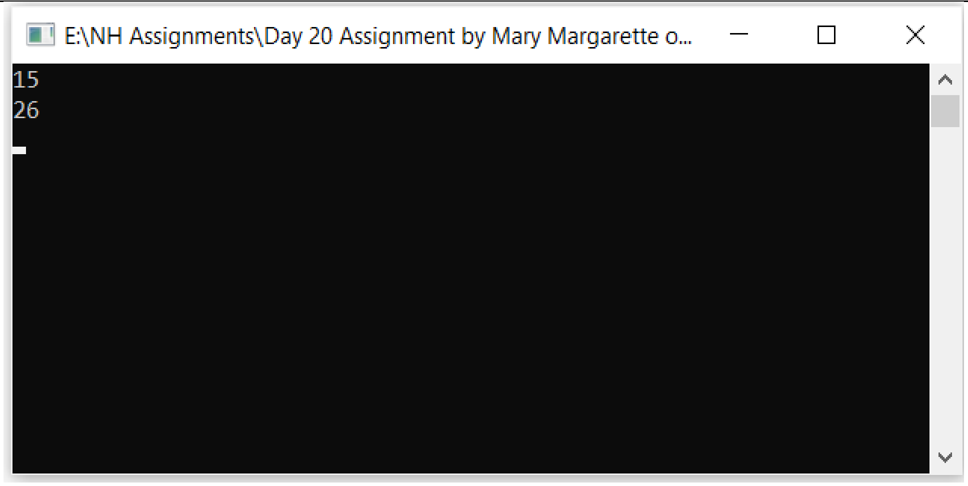
```
Before58
Inside method:116
After116
```

Out Parameter Code:



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

namespace Day_20_Project_1
{
    0 references
    internal class Program
    {
        1 reference
        public static void Add(out int a, out int b)
        {
            a =15;
            b = 26;
        }
        0 references
        static void Main(string[] args)
        {
            int i, j;
            Add(out i, out j);
            Console.WriteLine(i);
            Console.WriteLine(j);
            Console.ReadLine();


        }
    }
}
```

Output:



E:\NH Assignments\Day 20 Assignment by Mary Margarette o...

```
15
26
```