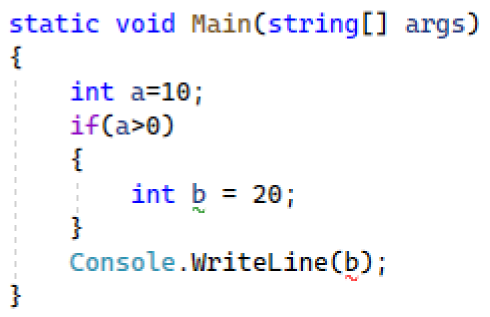
Day 20(18-02-2022) Assignment
By
Sudha Kumari Sugasani

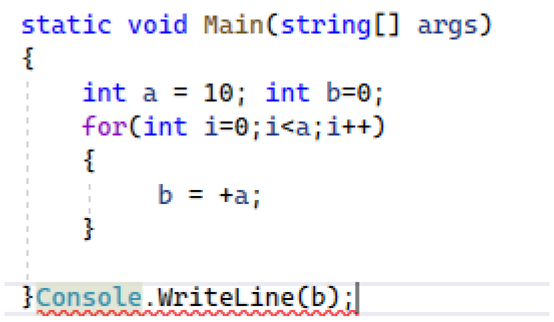
Q1. Research and understand scope of variables in C#

Here the scope of variable(b) is only within the if condition



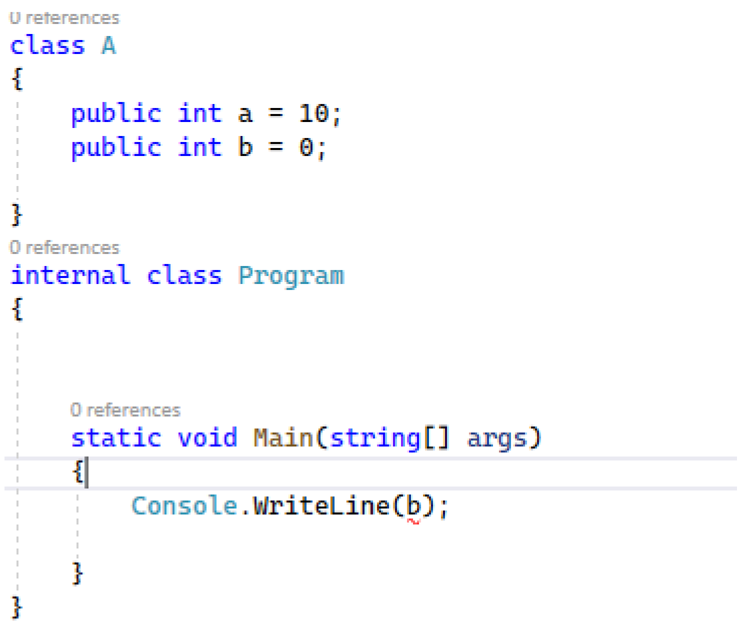
```
static void Main(string[] args)
{
    int a=10;
    if(a>0)
    {
        int b = 20;
    }
    Console.WriteLine(b);
}
```

Here the scope of variable is within the method only



```
 static void Main(string[] args)
 {
     int a = 10; int b=0;
     for(int i=0;i<a;i++)
     {
          b = +a;
     }

 }Console.WriteLine(b);
```

Here the scope of variable is within the class only



```
0 references
class A
{
    public int a = 10;
    public int b = 0;

}
0 references
internal class Program
{


    0 references
    static void Main(string[] args)
    {
        Console.WriteLine(b);

    }
}
```

Q2. What are delegates in C#

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

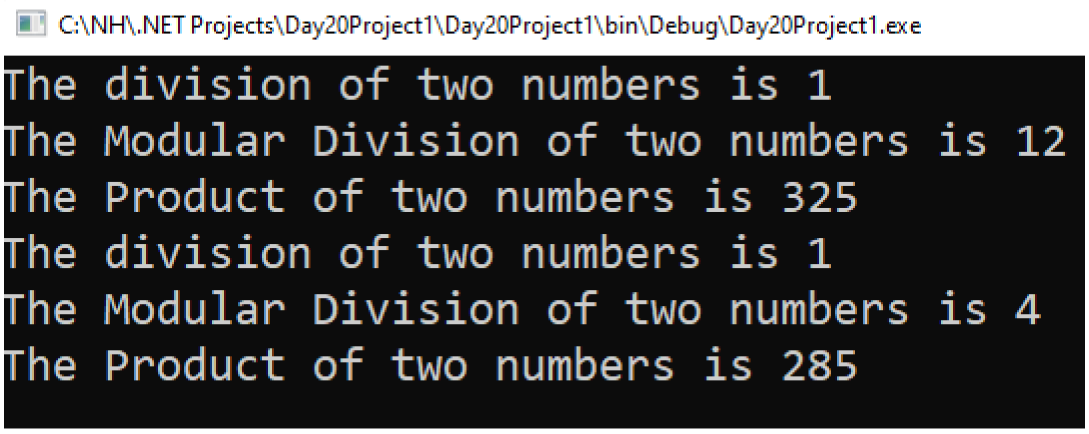
- ➢ Delegate is a function pointer.
- ➢ Using delegates we can call or point to one or more methods.
- ➢ When declaring a delegate,return type and parameters must watch with the methods you want to point using the delegates.
- ➢ Benefit of delegate is that using single call from delegate all your methods pointing to delegate will be called.
- ➢ Delegates are two types:

a) Single-Cast Delegate-If a delegate is pointing to single method it is single-cast delegate.
b) Multi-Cast Delegate-If a delegate is pointing to multiple methods then it is mullti-cast delegate.

Code:

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

namespace Day20Project1
{
    /*******************************************
     * Author:Sudha Sugasani
     * Purpose:Example program for delegates
     * *****************************************/


    /// <summary>
    /// By creating this method we will use delegates
    /// </summary>
    /// <param name="a">int</param>
    /// <param name="b">int</param>
    public delegate void Mycaller(int a, int b);


    class Delegates1
    {
        /// <summary>
        /// This method will caluculate the Division of two numbers
        /// </summary>
        /// <param name="a">int</param>
        /// <param name="b">int</param>
       public static void Div(int a,int b)
        {
            Console.WriteLine($"The division of two numbers is {a/b}");
        }

        /// <summary>
        /// This method will caluculate the ModularDivision of two numbers
        /// </summary>
        /// <param name="a">int</param>
        /// <param name="b">int</param>
        public static void ModuloDiv(int a,int b)
        {
            Console.WriteLine($"The Modular Division of two numbers is {a%b}");
        }

        /// <summary>
        /// This method will caluculate the Product of two numbers
        /// </summary>
        /// <param name="a">int</param>
        /// <param name="b">int</param>
        public static void Mul(int a,int b)
        {
            Console.WriteLine($"The Product of two numbers is {a*b}");
        }

    }
    internal class Program
    {

        static void Main(string[] args)
        {
            Mycaller mc = new Mycaller(Delegates1.Div);
            mc += Delegates1.ModuloDiv;
            mc += Delegates1.Mul;

            //25,13
            mc(25, 13);

            //19,15
            mc(19, 15);
            Console.ReadLine();
        }
    }
}
```

Output:



```
C:\NH\.NET Projects\Day20Project1\Day20Project1\bin\Debug\Day20Project1.exe
The division of two numbers is 1
The Modular Division of two numbers is 12
The Product of two numbers is 325
The division of two numbers is 1
The Modular Division of two numbers is 4
The Product of two numbers is 285
```

Q3. What are nullable types in C#
WACP to illustrate nullable types
Write some properties of nullable types (like HasValue)

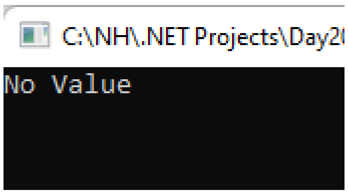
For value type we cannot assign null value,to assign null values for value type we will use nullable type.

- **Nullable.HasValue and Nullable.Value**:It is used to check the value. If the object assigned with a value, then it will return "True" and if the object is assigned to null, then it will return "False". If the object is not assigned with any value then it will give compile-time error.
- **GetValueOrDefault(T):**It is used to get the assigned value or the provided default value, if the value of nullable type is null.
- **Null-coalescing operator(??)** -It is used to assign a value to the underlying type originate from the value of the nullable type.

Code:

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

namespace Day20Project2
{
        /************************************************
         * Author:Sudha Sugasani
         * Purpose:Example program for Nullable types
         * ********************************************/
    class NullableTypes1
    {
         byte?  input = null;
        /// <summary>
        /// The method will check if the input has value or not
        /// </summary>
        public void Check()
        {
            if(input.HasValue)
            {
                Console.WriteLine(input*input);
            }
            else
            {
                Console.WriteLine("No Value ");
            }
        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            NullableTypes1 obj = new NullableTypes1();
            obj.Check();
            Console.ReadLine();
        }
    }
}
```

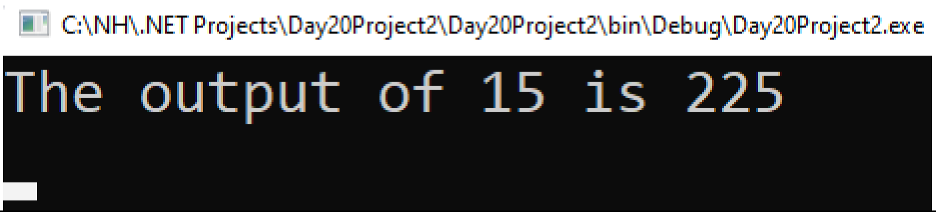
Output if we give input as null:



C:\NH\.NET Projects\Day2

```
No Value
```

Output if we give some value(15):



C:\NH\.NET Projects\Day20Project2\Day20Project2\bin\Debug\Day20Project2.exe

```
The output of 15 is 225
```

Q4. Out, ref - parameters
please research on these two types of parameters
write a C# program to illustrate the same.

Ref:

- It is necessary the parameters should initialize before it pass to ref.
- It is not necessary to initialize the value of a parameter before returning to the calling method.

Out:

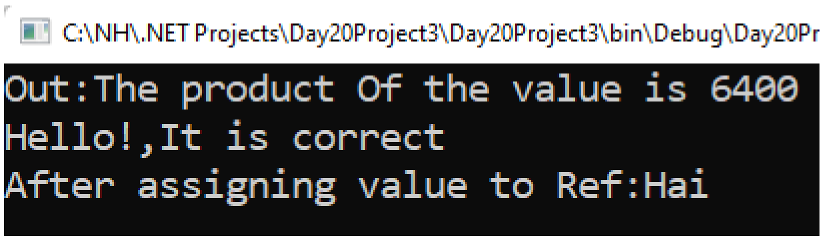
- It is not necessary to initialize parameters before it pass to out.
- It is necessary to initialize the value of a parameter before returning to the calling method.

Code:

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

namespace Day20Project3
{
    /*******************************************************
     * Author:Sudha Sugasani
     * Purpose:Example program for ref,out type parameters
     * ****************************************************/
    class TypesOfParameters
    {
        /// <summary>
        /// This method will tell give the Example for Out type parameter
        /// </summary>
        /// <param name="a">int</param>
        public static void ExampleforOut(out int a)
        {
            a= 80;
            a*= a;
        }
        /// <summary>
        /// This method will tell give the Example for Ref type parameter
        /// </summary>
        /// <param name="s1">String</param>
        public static void ExampleforRef(ref string s1)
        {
            if (s1 == "Hello")
            {
                Console.WriteLine("Hello!,It is correct");
            }
            s1 = "Hai";

        }
    }
    internal class Program
    {
        static void Main(string[] args)
        {
            int a;
            TypesOfParameters.ExampleforOut(out a);
            Console.WriteLine($"Out:The product Of the value is {a}" );

            string s2 = "Hello";
```

```
                TypesOfParameters.ExampleforRef(ref s2);


                Console.WriteLine($"After assigning value to Ref:{s2}");

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

Output:



C:\NH\.NET Projects\Day20Project3\Day20Project3\bin\Debug\Day20Pr

```
Out:The product Of the value is 6400
Hello!,It is correct
After assigning value to Ref:Hai
```