DAY20 ASSIGNMENT
BY
MANOHAR ANDE
18TH FEB 2022

1.Research and understand scope of variables in C#

1.Class Level Scope
2. Method Level Scope
3. Block Level Scope

1.CLass Level Scope:

- The Variable declared in the class(but out side the method)can be accessedanywhere wihin the class.
- it can accessed by the non-static methods in the class.
- The variable doesn't affect he class level scope variables

  2.Method Level Scope:

  - The variable 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 variable doesn't exist after the methods execution.

    3.Block Level Scope:

    - The variable which are declared inside for, while statement etc are called Block Level Scope
    - These variables are termed as loop variable as they limit their scope up to the body of the statement in which 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.

DELEGATES: Delegates is like a function pointer.

- Using delegates we can call or Point to one or more methods
- When declaring a delegate, return and parameters must be same with the methods you want to point using delegate

  BENEFITS:
- Using single call from delegate all your methods pointing to delegate will be called

  TYPES F OF DELEGATES :

  1. Single cast delegate

  2. Multi cast delegate

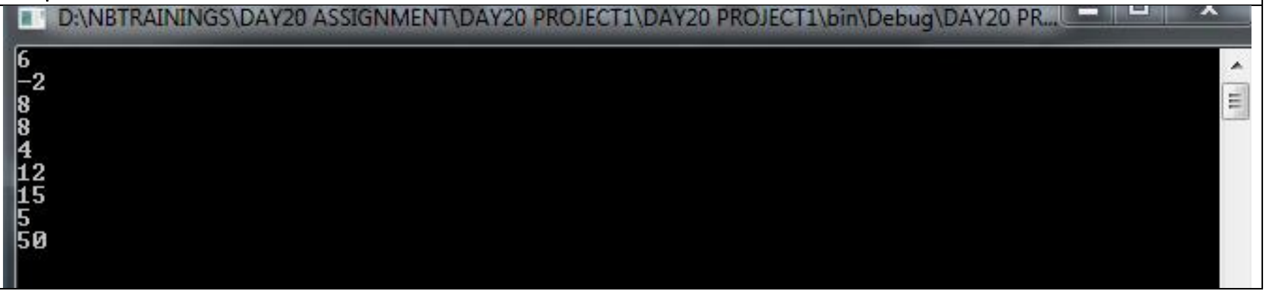
SINGLE CAST DELEGATE: A delegate pointing only one method is called single cast delegate .

MULTI CAST DELEGATE : A delegate pointing multiple methods is called Multi cast Delegate.

Code:

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

namespace DAY20_PROJECT1
{
    public delegate void MyCaller(int a, int b);
    internal class Program
    {
        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)
        {
            MyCaller mc = new MyCaller(Add);
            mc += Sub;
            mc += Mul;

            //2,4

            mc(2, 4);

            //3,2
            mc(6, 2);

            //10,5
            mc(10, 5);



            Console.ReadLine();


        }
    }
}
```

Output:



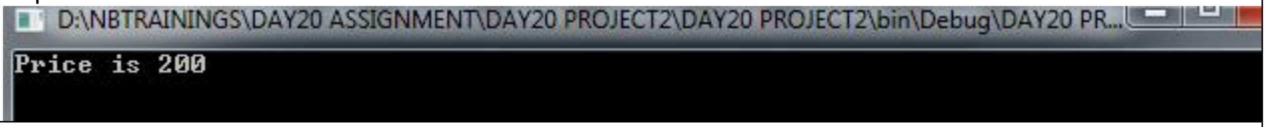
```
D:\NBTRAININGS\DAY20 ASSIGNMENT\DAY20 PROJECT1\DAY20 PROJECT1\bin\Debug\DAY20 PR...
6
-2
8
8
4
12
15
5
50
```

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

Code:

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

namespace DAY20_PROJECT2
{
    internal class Program
    {
        static void Main(string[] args)
        {
            int? Price = 200;

            if(Price.HasValue)
                Console.WriteLine($"Price is {Price}");
            else
                Console.WriteLine("No value");
            Console.ReadLine();
        }
    }
}
```

Ouput:



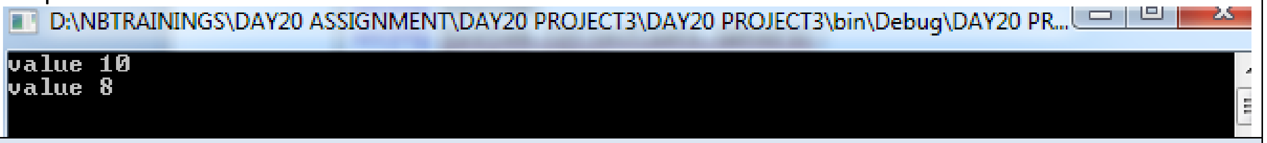
```
D:\NBTRAININGS\DAY20 ASSIGNMENT\DAY20 PROJECT2\DAY20 PROJECT2\bin\Debug\DAY20 PR...
Price is 200
```

Q4. out, ref - parameters
please research on these two types of parameters

write a C# program to illustrate the same.

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

namespace DAY20_PROJECT3
{
    internal class Program
    {
        public static void Out(out int a)
        {
            a = 10;
        }
        public static void Ref(ref int b)
        {
            b = 8;
        }
        static void Main(string[] args)
        {
            int c;
            int d = 4;
            // c value using out parameter
            Out(out c);
            //d value to ref parameter
            Ref(ref d);
            Console.WriteLine($"value {c}");
            Console.WriteLine($"value {d}");

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

Output:



D:\NBTRAININGS\DAY20 ASSIGNMENT\DAY20 PROJECT3\DAY20 PROJECT3\bin\Debug\DAY20 PR...

```
value 10
value 8
```

Out parameters:

- Out variables must be initialized in Methods itself.
- out is used when function return more than one value.

Ref parameters:

- Ref variables must be initialized before passing methods.
- Ref is used to change the value in the call function and return it.