Day 20 Assignment

By

Vinay Kudali

18-02-2022

NB Healthcare Technologies

1. Research and Understand the Scope of variables in C#

- In c#, the variables are 3 Types
  a) Class Level Variables
  b) Method level Variables
  c) Block Level Variables

Class Level Variables:

- Declaring the variables in a class but outside any method can be directly accessed anywhere in the class.
- Class variables can also be accessed outside the class
- Class level scoped variable can be accessed by the non-static methods of the class in which it is declared.

Method Level Variables:

- Variables that are declared inside a method have method level scope. These are not accessible outside the method.
- However, these variables can be accessed by the nested code blocks inside a method.
- These variables are termed as the local variables.
- There will be a compile-time error if these variables are declared twice with the same name in the same scope.
- These variables don't exist after method's execution is over.

Block Level Variables:

- These variables are generally declared inside the for, while statement etc.
- These variables are also termed as the loop variables or statements variable as they have limited their scope up to the body of the statement in which it declared.
- Generally, a loop inside a method has three levels of nested code blocks (i.e., class level, method level, loop level).
- A variable which is declared inside a loop body will not be visible to the outside of loop body.

2. What are delegates in C#

- Single – cast delegate
- Multi – cast delegate

Write the points discussed about delegates in the class

- Delegate is a keyword.
- The return type of delegate should be the same as method return type.
- Parameters of delegate should be same as method parameters.
- A delegate is like a function pointer.
- Using delegate, we can call or point to one or more methods.
- Benefit of delegate is that, using a single call from delegate, all your methods pointing to delegate will be called.

Write C# code to illustrate the usage of delegates.

Code:

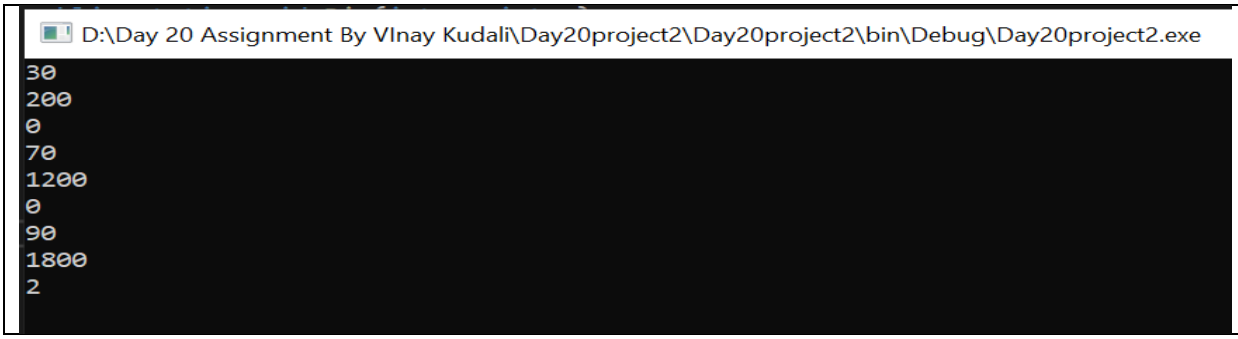
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Day20project2
{
   //Author: Vinay Kudali
   //Purpose: writing C# Program using Delegate
   public class Program
   {
      public delegate void MyCaller(int m, int n);

      public static void Add(int m, int n)
      {
         Console.WriteLine(m+n);
      }
      public static void Mul(int m, int n)
      {
         Console.WriteLine(m*n);
      }
      public static void Div(int m, int n)
      {
         Console.WriteLine(m/n);
      }
      static void Main(string[] args)
      {
         MyCaller mc = new MyCaller(Add);
         mc+=Mul;
         mc+=Div;

         mc(10, 20);
         mc(30, 40);
         mc(60, 30);
         Console.ReadLine();
      }
   }
}
```

Output:



3. What are nullable types in C#

The Nullable type **allows you to assign a null value to a variable**.

WACP to illustrate nullable types

Code:

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

namespace Day20Project3
{
   internal class Program
   {
      //Author : Vinay Kudali
      //Purpose: using Nullable Types
      static void Main(string[] args)
      {
         byte? data = null;
         if(data.HasValue)
            Console.WriteLine(data*data);
         else
            Console.WriteLine("No Data");
            Console.ReadLine();
      }
   }
}
```

Output:

Write Some properties of nullable types

- HasValue
- Value

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

Out Parameter:

The out is a keyword in C# which is used for passing arguments to methods as a reference type.

It is generally used when a method returns multiple values.

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 Day20Project3
{
   //Author: Vinay Kudali
   //Purpose: Out parameter
   class Program

   {
      public static void Add(out int x, out int y)
      {
         x = 79;
         y = 97;
         //x += y;
         //y += y;

      }

      static void Main(string[] args)
      {
         int a, b;
         Add(out a, out b);
         Console.WriteLine(a);
         Console.WriteLine(b);
         Console.ReadLine();
      }
```

```
    }
}
```

Output:

Ref-Parameter:

The ref keyword in C# is used for passing or returning references of values to or from methods.

Basically, it means that any change made to a value that is passed by reference will reflect this change since you are modifying the value at the address and not just the value.

Code:

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

namespace Day20Project4RefParameter
{
    internal class Program
    {
        //Author : Vinay Kudali
        //Purpose : Ref Parameter
        public static void Multi(ref int z)
        {
            z *= z;
            Console.WriteLine("Inside method:"+z);
        }

        static void Main(string[] args)
        {
            int z = 10;
```

```
        Console.WriteLine("Before:"+z);
        Multi(ref z);
        Console.WriteLine("After:"+z);
        Console.ReadLine();
    }
  }
}
```

Output: