|  |
| --- |
| **Day 20 Morning Assignments**  **By**  **Manoj Yekolla**  **18-Feb-2022** |

|  |
| --- |
| **1. Research and understand scope of variables in C#** |
| **Scope of variables :** |
| * A Variable Scope refers to the availability of variables in certain parts of the code * In c# ,A variable has three types of scope : * Class level Scope * Method Level Scope * Block Level Scope |
| **a ) Class Level Scope :** |
| * When we declare a variable inside the class , The variable can access ed with in the class.These is know as class level variable scope. |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day20Project1  {  class ClassScope  {  // class level variable  string str = "Manoj Yekolla-ClassLevel Scope";  public void display()  {  Console.WriteLine(str);  }  }  internal class Program  {  static void Main(string[] args)  {  ClassScope obj=new ClassScope();  obj.display();  Console.ReadLine();  }  }  } |
| Output :  Screenshot (331) |
| **b )Method Level Scope** |
| * When we declare variable inside a method , a variable can not be accessed outside of the method.These is known as Method level variable scope.   using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace MethodLevelScope  {  class MethodLevel  {  public void method1()  {  //display variable inside method  string str = "method level";  }  public void method2()  {  //accessing str from method2() //Error Code  Console.WriteLine(str);  }  }  internal class Program  {  static void Main(string[] args)  {  MethodLevel obj = new MethodLevel();  obj.method1();  Console.ReadLine();    }  }  } |
| Output :  Screenshot (333) |
| **C ) Block Level Scope :** |
| * When We declare a variable inside a block( for loop, while loop ,if else..) the variable can only be accessed with in the block .These is known as block level variable scope. |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace BlockLevelScope  {  class BlockLevel  {  public void display()  {  for (int i = 0; i <= 3; i++)  {  }  Console.WriteLine(i);  }  }  internal class Program  {  static void Main(string[] args)  {  BlockLevel level = new BlockLevel();  level.display();  Console.ReadLine();  }  }  } |
| Output :  Screenshot (335) |

|  |
| --- |
| **2. What are delegates in C#** |
| * Delegates are similar to pointers to function . * A Delegates is a reference type variable that holds the reference to method. * Syntax : Public delegate void MyDelegate(string s); |
| **Write the points dicussed about delegates in the class** |
| * 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. * Benefits of delegate is using single call from delegate ,all your methods pointing to delegate will be called. * They are two types of delegates * .Single cast delegate ,Multi caste Delegate. |
| **Write C# code to illustrate the usage of delegates.** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace DelegatesProgram  {  public delegate void Manoj(int a, int b);  internal class Program  {  public static void Add(int a ,int b)  {  Console.WriteLine(a+b);  }  public static void Mul(int a, int b)  {  Console.WriteLine(a \* b);  }  public static void Div(int a, int b)  {  Console.WriteLine(a / b);  }  static void Main(string[] args)  {  Manoj obj = new Manoj(Add);  obj += Mul;  obj += Div;  //12,13  obj(12, 13);  Console.WriteLine("-----------------------------------------");  //44,6  obj -= Add;  obj(44, 6);  Console.ReadLine();  }  }  } |
| **Output :**  Screenshot (339) |

|  |
| --- |
| **3.What are nullable types in C#** |
| * C# provides a special data types ,the nullable types ,to which you can assign normal range of values as well as null values. * Nullable types allows you to assign a null value to a variable. * Nullable types can only work with value types. |
| **Properties :** |
| * The Hash value Property returns true if the variable contains a value ,Or false it is null. * You can use the “?” operator to shorthand the syntax eg : * Int ? ,long ? instead of using Nullable<T>. |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace NullableTypes\_Pgm  {  internal class Program  {  static void Main(string[] args)  {  int ? input = 4;  if (input.HasValue)    Console.WriteLine(input\*input);    else  Console.WriteLine("Novalues found");  Console.ReadLine();  }  }  } |
| Output :  Screenshot (341) |

|  |
| --- |
| **4. out, ref - parameters**  **please research on these two types of parameters** |
| **Out Parameter :** |
| * The Out is a keyword in c# ,which is used for the passing the arguments to methods as a reference type. * It is generally used when a method returns multiple values . * The out parameter does not pass the parameter. |
| **Ref Parameter :** |
| * The **ref**is a keyword in C# which is used for the passing the arguments by a reference. we can say that if any changes made in this argument in the method will reflect in that variable when the control return to the calling method. * The ref parameter does not pass the property. |
| Code :  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace OutParameter  {    internal class Program  {  public static string Ref(ref int id)  {  string returnText = "Next-" + id.ToString();  id += 5;  return returnText;  }    public static string Out(out int id)  {  id = 5;  string returnText = "Next-" + id.ToString();  return returnText;  }    static void Main(string[] args)  {  Console.WriteLine("\n -----------Ref-----------");  int i = 5;  Console.WriteLine("previous value i is : " + i.ToString());  string testRef = Ref(ref i);  Console.WriteLine("current value i is : " + i.ToString());  Console.WriteLine("\n -----------Out-----------");  int j;  string testOut = Out(out j);  Console.WriteLine("Current value of j:" + j.ToString());  Console.ReadLine();  }  }  } |
| Output :  Screenshot (344) |