|  |
| --- |
| **Day 20 assignment**  **By**  **Nanam vaishnavi**  **18-feb-2022** |

|  |
| --- |
| 1. **Research and understand scope of variables in C#**   **Ans:** |
| So C# scope rules of variables can be divided into 3 categories. They are   * Class Level Scope * Method Level Scope * Block Level Scope   **Class Level Scope**  1) Access modifier of class level variables doesn’t affect their scope within a class.  2) Class level scoped variable can be accessed by non-static methods of the class in which it is declared.  3) Member variables can be accessed outside the class by using the access modifiers. |
| **Method Level Scope**  1) Variables that are declared inside a method have method level scope. These are not accessible outside the method.  2) These variables are termed as Local variables.  3) There will be a compile – time error if these variables are declared twice with the same name in the same scope.  4) These variables doesn’t exist after method’s execution is over. |
| **Block Level Scope**  1) These variables are generally declared inside the for, while statements etc.  2) A variable which is declared inside a loop body will not be visible to the outside of loop body.  3) Generally, a loop inside a method has three level of nested code blocks (I.e. class level, method level, loop level). |

|  |
| --- |
| **2) What are delegates in C#**  **Write the points discussed about delegates in the class**  **Write C# code to illustrate the usage of delegates.**  **Ans :** |
| **Delegate**  It is a type that represents reference to methods with a particular parameter list and return type.  **Points discussed about delegates in the class**  1) It is like a function pointer.  2) Using delegate we can point or call multiple methods.  3) When declaring a delegate, return type and parameter must match to methods.  **Benefits:**   * Using single call from delegate, all your method pointing to delegate will be called.   **Types of Delegate**  **1) Single Cast Delegate**   * Pointing towards only one method**.**   **2) Multi Cast Delegate**   * Pointing towards multiple methods. |
| **CODE** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // Author : Nanam Vaishnavi  // Purpose : C# code to illustrate the usage of delegates.  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  namespace Day20Project1  {  public delegate void Operations(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)  {  Operations op = new Operations(Add);  op += Div;  op += Mul;  //15,16  op(15, 16);  Console.WriteLine("==========================");  // 12,13  op(12, 13);  Console.WriteLine("===========================");  // 14,18  op(14, 18);  Console.WriteLine("===========================");  Console.ReadLine();  }  }  } |
| **OUTPUT** |
|  |

|  |
| --- |
| **3) What are nullable types in C#**  **WACP to illustrate nullable types**  **Write some properties of nullable types (like HasValue)**  **Ans :** |
| **Nullable :**  It allows you to assign a null value to a variables.  **Syntax :**  Nullable<data\_type> var\_name = null;  OR  datatype ? variable\_name = null;  **Properties :**  1) **Nullable.HasValue & Nullable.Value :** Used to check the value.(if value assigned then it will return TRUE; if object assigned to null it will return FALSE).  2) **null-coalescing operator(??) :** To assign a value to the nullable type.  3) **Nullable<T>** : Allows assignment of null to value types. |
| **CODE** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // Author : Nanam Vaishnavi  // Purpose :To illustrate nullable types  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  namespace Day20Project2  {  internal class Nullable  {  static void Main(string[] args)  {  int ? salary = 60000;  Console.WriteLine(salary);  byte ? age = null;  if (age.HasValue)  Console.WriteLine(age \* age);  else  Console.WriteLine("No Value");  Console.ReadLine();  }  }  } |
| **OUTPUT** |
|  |

|  |
| --- |
| **4) out, ref – parameters**  **please research on these two types of parameters**  **write a C# program to illustrate the same.** |
| **Ref :**   * It is used to pass an argument as a reference. When value of that parameter is changed in the method, it gets reflected in the calling method. * An argument that is passed using a ref keyword must be initialized in the calling method before it is passed to the called method. |
| **CODE** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // Author : Nanam Vaishnavi  // Purpose : C# code using REF parameter  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  namespace Day20Project4  {  internal class Program  {  private static void SetValue(ref string str1)  {  // Checking parameter value  if (str1 == "NBH")  {  Console.WriteLine("Welcome to NBH");  }    // Assigning new value  str1 = "Hello NBH!!!";  }  static void Main(string[] args)  {  string str = "NBH";  SetValue(ref str);  Console.WriteLine(str);  Console.ReadLine();  }  }  } |
| **OUTPUT** |
|  |
| **Out :**   * Out is a keyword in C# which is used for passing the arguments to methods as reference type. * It is generally used when a method returns multiple values. * Out Parameter doesn’t pass the property. |
| **CODE** |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // Author : Nanam Vaishnavi  // Purpose : Program using OUT parameter  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  namespace Day20Project3  {  internal class Program  {  public static void Sum (out int V)  {  V = 60;  V = V + 60;  }  static void Main(string[] args)  {  int V;  Sum (out V);  Console.WriteLine ("Sum of Value is : {0}",V);  Console.ReadLine();  }  }  } |
| **OUTPUT** |
|  |