|  |
| --- |
| **Day-20 Assignment**  **By**  **Bhanu Rama Krishna Prakash Jakkamsetti**  **18/2/2022** |

|  |
| --- |
| 1.Research and understand scope of variables in C# |
| In C# variable can be divided into 3 types  Class Level Scope  Method Level Scope  Block Level Scope |
| Code: Class Level |
| * Access modifier of class level variables doesn’t affect their scope within a class. * Class level scoped variable can be accessed by non-static methods of the class in which it is declared. * Member variables can be accessed outside the class by using the access modifiers. |
| Code: Method Level |
| * Variables that are declared inside a method have method level scope. These are not accessible outside the method. * These variables are termed as 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 doesn’t exist after method’s execution is over. |
| Code: Block Level |
| * These variables are generally declared inside thefor, while statements etc. * 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. |
| 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 * It is like a function pointer. * Using delegate we can point or call multiple methods. * 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   * Single Cast Delegate   Pointing towards only one method.   * Multi Cast Delegate   Pointing towards multiple methods. |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day20\_project1  {  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // Author :bhanu rama krishna prakash jakkamsetti  // Purpose :illustrate the use of delegates.  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  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 lol = new Operations(Add);  lol += Div;  lol += Mul;  //15,16  lol(15, 16);  // 12,13  lol(12, 13);  // 14,18  lol(14, 18);    Console.ReadLine();  }  }  } |
| Output: |
|  |

|  |
| --- |
| 3. What are nullable types in C#  WACP to illustrate nullable types  Write some properties of nullable types (like HasValue) |
| Nullable :   * It allows you to assign a null value to a variables.   Syntax :   * Nullable<data\_type> var\_name = null; * datatype ? variable\_name = null;   Properties :   * 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). * Null-coalescing operator(??) : To assign a value to the nullable type. * Nullable<T> : Allows assignment of null to value types. |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day20\_project2  {  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // Author :bhanu rama krishna prakash jakkamsetti  // Purpose :deal with nullable  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  internal class Program  {  static void Main(string[] args)  {  int? salary = 45000;  Console.WriteLine(salary);  byte? experience = null;  if (experience.HasValue)  Console.WriteLine(experience \* experience);  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. |
| 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. |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day20\_project3  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* author : bhanu rama krishna prakash jakkamsetti  \* purpose : Deal with out keyword  \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  internal class Program  {  public static void Sum(out int i)  {  i = 60;  i += 60;  }  static void Main(string[] args)  {  int i;  Sum(out i);  Console.WriteLine($"Sum of Value is : {i}");  Console.ReadLine();  }  }  } |
|  |
| 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. |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Day20\_project5  {  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // Author : Bhanu rama krishna prakash Jakkamsetti  // Purpose : Deal with ref keyword  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  internal class Program  {  private static void SetValue(ref string s)  {  // Checking parameter value  if (s == "Hello")  {  Console.WriteLine("Welcome to my world");  }  // Assigning new value  s = "Divya ";  }  static void Main(string[] args)  {  string s1 = "Hello";  SetValue(ref s1);  Console.WriteLine(s1);  Console.ReadLine();  }  }  } |
|  |