# **Day-20 Assignment By**

# **M. Pallavi**

1. a.What are delegates in C#.

* Delegates in C# are the Type-Safe Function Pointer. It meansdelegates contains the reference to several methods and call them when needed.
* We can create numbers of methods as you need and attach it to delegates. At runtime, an event gets fired and delegates dynamically call the function and show the result.

b. Write the points discussed about delegates in the class.

1. Delagate is like a function pointer.

2. Using Delegates we can cal or point to one or more methods.

3. When declaring a delegate ,return type and parameters with the method you want to point using the delegate.

Benefits:

Using single call from delegates, all your methods pointing to delegate will be called.

|  |
| --- |
| 1.Write C# code to illustrate the usage of delegates. |
| Code:  using System;  // Author : M.Pallavi  // Purpose : Delegate Example  namespace DelegatesExamp  {  public delegate void rectDelegate(double height, double width);  class Rectangle  {    public void area(double height, double width)  {  Console.WriteLine("Area is: {0}", (width \* height));  }    public void perimeter(double height, double width)  {  Console.WriteLine("Perimeter is: {0} ", 2 \* (width + height));  }  }  internal class Program  {  static void Main(string[] args)  {    Rectangle rect = new Rectangle();    rectDelegate rectdele = new rectDelegate(rect.area);    rectdele = rectdele+rect.perimeter;    rectdele.Invoke(6.3, 4.2);  Console.WriteLine();    rectdele.Invoke(16.3, 10.3);  Console.ReadLine();  }  }  } |
| Output: |

2. What are nullable types in C#.

A value type cannot be assigned a null value. For example, *int i = null* will give you a compile time error.

C# 2.0 introduced nullable types that allow you to assign null to value type variable.

|  |
| --- |
| **3.** WACP to illustrate Nullable types |
| using System;  // Author : M.Pallavi  // Purpose : Nullable Type C# Code Example.  namespace NullableTypes  {  internal class Program  {  static void Main(string[] args)  {  int? firstValue = 20;  int? secondValue = null;  if (firstValue.HasValue)  {  Console.WriteLine(firstValue.Value);  }  if (secondValue.HasValue)  {  Console.WriteLine(secondValue.Value);  }  Console.ReadKey();  }  }  } |
| Output: |

Write some properties of nullable types (like HasValue).

## Shorthand

Example: Shorthand syntax for Nullable types

int? i = null;

double? D = null;

By using the '?' operator to shorthand the syntax

e.g. int?, long?

instead of using Nullable<T>.

### ?? Operator

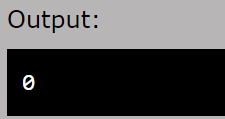
Use the '??' operator to assign a nullable type to a non-nullable type.

Example: ?? operator with Nullable Type

int? i = null;

int j = i ?? 0;

Console.WriteLine(j);



HasValue:

Has Value indicates whether an instance of a nullable value type has a value of its underlying type. ... Value gets the value of an underlying type if HasValue is true .

If HasValue is false , the Value property throws an InvalidOperationException.

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

Ref:

The **ref**is a keyword in C# which is used for the passing the arguments by a reference. Or 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](https://www.geeksforgeeks.org/c-properties/).

Out :

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 property.

.

|  |
| --- |
| 4.write a C# program to illustrate the out, ref – parameters. |
| using System;  // Author : M.Pallavi  // Purpose : C# Code to illustrate, the Ref & Out keywords.  namespace ConsoleApplication1  {  class Refexam  {  public void show(ref int x, out int y)  {  y = x % 2;  x = x / 2;  }  }  class Program  {  static void Main(string[] args)  {  int a;  int reminder;  Refexam obj = new Refexam();    a = 35;  Console.WriteLine("before call a :" + a);// this will display 35  obj.show(ref a, out reminder);  Console.WriteLine("after call a :" + a);// this will display 17  Console.WriteLine("before call a :" + reminder); Console.ReadKey();  }  }  } |
| Output: |