Day 3

Function parameter - the out Modifier

- Is used to declare that a function parameter is used for output purpose only
- Such parameters must be declared with **out** keyword in function declaration
- When function is called a variable to receive the output should be specified in position of output parameters prefixed with **out** keyword.

Function parameter - the ref Modifier

- Is used to declare that a function parameter is used to for passing value to the function as well as for returning the results back to caller
- Such parameters must be declared with **ref** keyword in function declaration
- When function is called a variable to receive the output should be specified in position of ref parameters prefixed with **ref** keyword.

Abstract Classes

- Is a class that is an incomplete class
- It has a missing or incomplete implementation.
- Abstract classes can be used only as a base class from which other classes can inherit

Abstract Methods

- An abstract method is a member in a base class that does not have an implementation/function body.
- Abstract methods have only a signature.
- When a class derives from a abstract class , derived class must give a definition for the defining an abstract method by overriding

sealed class

- A class if declared as sealed then no further inheritance will be possible from that class.

Interfaces

- Prior to C# v 8.0 an interface was a named set of abstract members.
- From C# 8.0 onwards an interface may contain methods with implementation that may or may not be overridden by the implementing class.
- Ie., interfaces can contain member definitions (like abstract members), members with default implementations (like virtual methods), and static members and properties.
- a class or structure can implement as many interfaces as necessary,

- By convention, .NET interface names are prefixed with a capital letter *I*.

- an interface is defined in C# using the interface keyword.
- An interface can inherit another interface
- From C# v 8.0 on wardsAn interface can have members with private, public internal, protected, access specifiers.
- Interface can contain static elements

Interface Types vs. Abstract Base Classes

- 2 differences are there between interface and abstract class
  - interfaces cannot have nonstatic constructors
  - a class can implement multiple interfaces.

Interface

- We cannot create instance of interface
- But interface need to be implemented by a class
- A class can impement inerface by using : (colon) followd by interface name in class definition
- If a class is inheriting another class and implementing an interface the base class name should come first after colon

Structure

- Structures are types that can contain any number of data fields and methods that operate on these fields.
- Structures can't inherit from other class or structure
- Also structures can't be the base of a class.
- Structure can implement interface
- Structure may contain data members, member functions and properties
- Constructor may be there in a structure
- Structure and enum are value type where as class is reference type

enum

- An enum is a custom data type of name-value pairs.
- By default, the first element of enum is set to the value zero (0)
- Next elements will get value with the pattern n+1
- But you can change the initial value/value of any element as per requirement.
- Successive elements will get value incremented by 1
- You can declare enum type variables in your classes/structures/functions

Value type & Reference Type

- Value type data elements allocated memory on the stack
- Reference type data elements allocated memory on the heap.
- Data allocated on the stack can be created and destroyed quickly, as its lifetime is determined by the defining scope.
- Heap-allocated data, on the other hand, is monitored by the .NET Core Garbage Collector

- C# reference type by default inherit from System.Object
- C# value types derive from System.ValueType

C# Nullable Types

- a nullable type can represent all the values of its underlying type, plus the value null.
- This will be helpful when working with relational databases using ORM.
- To define a nullable variable type, the question mark symbol (?) is suffixed to the underlying data type.
- Eg int?, float?

?? operator

- C# ?? Operator referred null-coalescing operator. allows you to assign a value to a nullable type if the retrieved value is null.
- int? i = null
- int j= i?? 100;

The Null-Coalescing Assignment Operator

- (??=). This operator assigns the left-hand side to the right-hand side only if the left-hand side is null.
- .eg

```
int? nullableInt = null;
nullableInt ??= 12;
 nullableInt ??= 14;
```

.

Null Conditional Operator

- To check if a variable is null

  if (args != null) { int i= args.Length }

  Can be replaced by

  {args?.Length}
- Is used to avoid exceptions if a variable is null
- We can change the command with a null-coalescing operator to assign a default value also
- Int I =args?.Length ?? 0}