Dot Net Basics

Tuesday, February 2, 2021 12:04 PM

Built In Data Type	Byte - Sbyte	Short-ushort	Int-uint	Long-ulong	Float	Double	Decimal	Boolean	char

Other Data Type	String	DateTime

User Define Types	Class	Structure	Enumerated Data Type

Data Structures	Array	Collections	Generics	Tuple	ValueTuple

Generics	List	Stack	Queue	HashSet	Sorted Set	Sorted List	Dictionary	Sorted Dictionary	Link List

Collections	Array-List	Stack	Queue	Sorted List	Hash Table

Namespaces: Contains All the Type like Classes and All
- Contains All the Types.
- Default Access Specifier is internal
- Possible Access Specifiers are Internal and Public Only.

Access Specifiers:
1. **Private :** With in Class
2. **Protected :** Within Derived Class Only
3. **Private Protected:** In Derived Class Only (But Only With in Assembly)
4. **Internal:** Within Same Assembly
5. **Protected Internal:** In Derived Class Only (Across Assembly)
6. **Public :** Every Where

Methods:
1. Can be Static or instantaneous
2. Using abstract can create abstract (not private method) and no body but must in abstract class only
3. Using Partial and No body can make partial but in partial class only, void type only and must define once only
4. Using **params** can take any number of arguments but it must on last position.
5. Using Default value for example function name(int a,int b=5) can create optional filed here also optional or default field must on last position
6. Using [Optional] Attribute Can Make Field optional Same as 4,5 filed must be last.
7. Using of ref and output
   a. **Ref** : To Pass the reference of variable but before pass must initialize filed, and use this object when want to change ref of object other then not require
   b. **Out**: Can Say when want output with particular variable then passing out para and in this we must have to re initialize data
   c. Can Say Both ref and out use to return multiple value.
8. Can Overload Methods or function based on (Can Do Same With Indexers and Constructor)
   a. Number of Arguments
   b. Types of Argument
   c. Sequence of Argument
   d. But Not Using Static or Non Static , Access Specifiers, Return Type
9. Can Allow Override a method using virtual and override.
10. Can Sealed Only Overridden Method
11. To Hide Overriding of method using new keyword called method hiding, but it create issue because it will invoke method of reference variable not memory allocator.
12. While Passing arguments can use name if want to break order like (para3:value,para2:value,para1:value)

Class:

1. Variables
2. Properties
3. Constructors
4. Methods
5. Static Variables **or** Methods **or** Properties **or** Constructor
6. Indexers
7. May Other Types Interfaces, Classes, Data Structures, Structures, Static Classes, Enums, Delegates, abstract class
8. More:
   a. Class Can Be Partial With partial keyword means divided one class in same physical file, but must follow rule of class like
      i. Must have Same access Specifier,
      ii. Not inherit more then one class at the end
      iii. If any of the mart made sealed or abstract then whole become abstract
      iv. Not Have Same Methods or members as all accessible in every part of class
   b. Static Constructor not have Access Specifiers
   c. Have Default empty Constructor but if declare any constructor manually then that will removed.
   d. Sealed to stop inheritance, base to access Members of Base Class(Not Interrace and abstract method) and this to access of this particular class only, abstract to create abstract method
   e. Default access specifier is private and can set out of 6.
   f. Can Inherit multiple interface but not more than one class
   g. To Implement methods of abstract class have to write override keyword, while not in case of interface

h. If Multiple interface have same method then can implement method like interfcename.Method name but then have to keep in mind that have to follo rules of interface method and to access either reference variable is type of that interface or have to type cast in that particular interface type.
i. If Inheriting Any generic Class Then Either Have to Make Derived Class as generic and Have to Proceed or Have to Define Type in Base.

```
Ex:
class Hello : Hello2<string>
      {
        public override Method(string name){}
        }
 or

class Hello<T> : Hello2<T>
      {
        public override Method(T name){}
        }
```

j. If Class is static then contains only static members and static function and Cant Not Derived or inherit also not have indexer, Static Class is Sealed By Default

Interface:
1. Only Abstract methods By default and public by default not permitted to allocate Access Specifier.
2. Can Inherit Interface only, and no needed to implement the inherited methods.
3. Methods are Non Static Only.
4. Can Not Sealed
5. Can Be Partial

Abstract Class:
1. Unlike Class It Contains all the things
2. With class name have keyword abstract and may have methods which not have implementation.
3. Can Inherit interfaces or class or both but not more then one class.
4. Can Not Sealed.
5. Other Same As Class as it is class.

Structure:
1. Value Type Not Reference type unlike class
2. Have Default Constructor can not write manually.
3. Can Not Assign Default value to instantaneous fields unlike class but can to static.
4. Have Properties, Methods, indexers, data members unlike class.
5. If Have Parameterized Constructor then allocate data to every instantaneous filed.
6. Can Be Partial.
7. Can Inherit only Interface/ es, but structures can not inherit anywhere.

Generics:
1. Basically Generics is a concept used when have to allow code for every data type but must define and pass data accordingly
   Ex: Calculator Code for Float, int, double
2. Can Make Generic Class or Function or Delegates or structure.

Exception:
1. Try, Catch, Finally, throw Keywords
2. If Return Statement is Inside the try, catch and finally block is there then first finally block get execute and then only value will return.
3. Can Not Write Return in Finally.
4. To Create Custom Exception Have to Inherit Exception class and just have to deal with constructor accordingly.
5. If Multiple Catch Blocks are there then appropriate block handle that so blocks must be in sequence for batter output.

Attribute:
1. Just to Add Meta Data Which is accessible using Compiler and it is helpful when using reflection API to get information About Assembly or Type or Member.
2. Can Create Custom Attribute By Just Inherit System.Attribute Class and Using Members and Properties.
3. AttributeUsage Attribute And Enum AttributeTargets Can Set for Which Type of this attribute is applicable.

Reflection API:
1. Using System. Reflection Namespace's classes we can get different information about various type and their member.

Delegate:
1. It is Function Pointer, which store function.
2. We Can Pass as argument in function or return, at time of calling have to pass name of function or lambda function.
3. It also Can Be Generic.
4. Can Subscribe Multi function and if Returning Value Then only get last value not all.

Lambda Expression:
1. It is anonymous function and useful when we want to give definition for delegate.
2. As it is anonymous function so can not call every time it is just code for one time or if give to delegate then useful when use that delegate.

Conditional Statements:
1. Switch Case
2. Go to Statement
3. If else if else , nested and ladder
4. Loops
   a. While
   b. Do While
   c. For
   d. Foreach

Operators:

1. Arithmetic
2. Relational
3. Conditional
4. Bitwise
5. Ternary
6. Special Like sizeof and typeof
7. Logical
8. Assignemnt

Data Structure:

1. List Contains Duplicate Elements, Hash Set Not Contain duplicate elements and sorted set Contain Unique and Ascending Elements
2. Dictionary Contain Key Value Pair and Sorted Dictionary and Sorted List Both Store Key Value Pair in Ascending Order but Sorted List is Faster.
3. Array List Que To List & Hash Set, Sorted List is equivalent to Sorted set.
4. Stack and Queue is present in both Generic and Collections.
5. May Possible in complex Type like structure and Class Have To Inherit Icomparer or IComparable for Sort Method of Some of Clauses like List and Same