///User Define DataType -> Struct , Class ,Enum, interface , Delegate, Record-> 'Complex DataType'.
/// any User Define DataType ...Define in NameSpace lVl.
////DataType -> Storage, Valdiation, operation
//// Value Type Fast Access Compare Between Reference DataType.

Struct	• Value Type
Class	• Reference Type
interface	• Reference Type
Enum	• Value Type
Delegate	• Reference Type
Record	• Reference Type

Type	Description
Struct	A struct is a lightweight data type that can be used to store data. Structs are similar to classes, but they do not support inheritance or polymorphism.
Class	A class is a data type that can be used to store data and define methods. Classes can be inherited from other classes, and they can be used to create objects.
Enum	An enum is a type that represents a set of named constants. Enums are often used to represent values that can have a limited number of possible values, such as the days of the week or the months of the year.
Interface	An interface is a contract that defines a set of methods that a class must implement. Interfaces are often used to decouple different parts of an application.
Delegate	A delegate is a type that represents a method call. Delegates are often used to implement events or callbacks.
Record	A record is a new type in C# 9 that is similar to a struct, but it supports inheritance and polymorphism. Records are often used to represent data that is related to each other.

- ***Boxing** refers to the process of converting a value type to an object type, and **unboxing** is the reverse process.*

- ***Struct Notes***
- ***Struct is a value type.***
- ***Limited Inheritance: They cannot be derived from other structs or classes, and they cannot be used as a base for other types.***
- ***Default Constructor: By default, structs have an implicit parameterless constructor provided by the compiler.***
- ***(in Case If Create constructor must initialize all the fields of the struct.)***
- ***Structs are commonly used for representing small, simple, and immutable data structures .***
- ***Structs are value types, and they are not subject to boxing and unboxing like reference types.***
- ***Size Limitation: The size of a struct is limited to a maximum of 16 bytes in C#.(else Use Class's)***
- ***Struct can implement interfaces.***
- ***Constructors in Struct***
- ***By default, structs have an implicit parameterless constructor.***
- ***struct constructor, you are responsible for explicitly initializing all the fields of the struct.***
- ***can overload constructors in structs by providing different parameter lists.***

- *Each struct constructor is specific to the struct type and is automatically invoked when a struct instance is created.*
- *Constructor Chaining (Calling Another Constructor)*

```
0 references
public complexNumber()
{
    this.real = 0;
    this.img = 0;
}
1 reference
public complexNumber(int real, int img)
{
    this.real = real;
    this.img = img;
}
0 references
public complexNumber(int real) : this(real, img: 0)
{
}
```

- *Class Notes*
- *Constructors are special methods in a class that are called when an object of that class is created using the new keyword.*
- *Constructors have the same name as the class and do not have a return type.*
- *Constructors are used to initialize the initial state of an object by setting the values of its fields or performing other initialization tasks.*
- *Constructors can be overloaded, allowing for multiple constructors with different parameter lists .*

- ***Class Relationships - "Is-A" and "Has-A"***
- ***"Is-A" Relationship (Inheritance)***
- ***"Has-A" relationship represents a composition or aggregation association between classes, indicating that a class has another class as a part or member.***
- ***Composition:***

->Composition implies that the contained object cannot exist independently of the container object.

- ***Aggregation:***

->Aggregation signifies that the contained object can exist independently of the container object.