



# Dot Net Training

By Besant Technologies

# Data Type

Alias	Type Name	Type	Size(bits)	Range	Default Value
sbyte	System.Sbyte	signed integer	8	-128 to 127	0
short	System.Int16	signed integer	16	-32768 to 32767	0
Int	System.Int32	signed integer	32	-231 to 231-1	0
long	System.Int64	signed integer	64	-263 to 263-1	0L
byte	System.byte	unsigned integer	8	0 to 255	0
ushort	System.UInt16	unsigned integer	16	0 to 65535	0
uint	System.UInt32	unsigned integer	32	0 to 232	0
ulong	System.UInt64	unsigned integer	64	0 to 263	0
float	System.Single		32	$\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^38$	0.0F
double	System.Double		64	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$	0.0D
decimal	System.Decimal		128	$\pm 1.0 \times 10^{-28}$ to $\pm 7.9228 \times 10^{28}$	0.0M
char	System.Char		16	U +0000 to U +ffff	'\0'

# Data Type

Alias	Type Name	Type	Size(bits)	Range	Default Value
bool	System.Boolean			True / False	
string	System.string				string.empty

# Variables



`type variable_name = value;`

or

`type variable_names;`

Example: `int z=10; int x;`

# Operators



Arithmetic Operators : addition , subtraction, Multiplication, Division , Modules

Relational Operators - Increment, Decrement

Logical Operators - <, >, >=, <=, ==, !=

Bitwise Operators - &, !, ^, <<, >>

Assignment Operators - =, +=

Conditional Operator - condition ? first\_expression : second\_expression;

# Conditional & Looping Statement



## Conditional Branching

This statement allows you to branch your code depending on whether or not a certain condition is met.

In C# are the following 2 conditional branching statements:

1. IF statement
2. Switch statement

# Conditional & Looping Statement



## Conditional Loops

C# provides 4 loops that allow you to execute a block of code repeatedly until a certain condition is met; they are:

- For Loop
- While loop
- Do ... While Loop
- Foreach Loop

Each and every loop requires the following 3 things in common.

1. **Initialization:** that sets a starting point of the loop
2. **Condition:** that sets an ending point of the loop
3. **Iteration:** that provides each level, either in the forward or backward direction

# Conditional & Looping Statement



Break and Continue;



# Class and Objects

## Class

A class is like a blueprint of a specific object that has certain attributes and features

example, a car should have some attributes such as four wheels, two or more doors, steering, a windshield, etc. It should also have some functionalities like start, stop, run, move, etc

## Object

An object is an instance of a class. With the help of object we can call the object

# Value Type and Reference Type



## Value type

- Stored in Stack
- Directly contains values
- Ex- int, bool, float

## Reference type

- Heap Memory
- Pointer values
- Ex- string, Class, Array, object

# Boxing and Unboxing



Boxing :

Convert data from Value type to reference type.

Heap Memory

Unboxing:

Convert Reference type to value type

Stack Memory

# Exception Handling



An exception is a problem that arises during the execution of a program.

A **try** block is used by C# programmers to partition code that might be affected by an exception. Associated **catch** blocks are used to handle any resulting exceptions. A **finally** block contains code that is run whether or not an exception is **thrown** in the try block, such as releasing resources that are allocated in the try block. A try block requires one or more associated catch blocks, or a finally block, or both.

# Exception Handling

Sr.No	Exception Class	Description
1	<code>System.IO.IOException</code>	Handles I/O errors.
2	<code>System.IndexOutOfRangeException</code>	Handles errors generated when a method refers to an array index out of range.
3	<code>System.ArrayTypeMismatchException</code>	Handles errors generated when type is mismatched with the array type.
4	<code>System.NullReferenceException</code>	Handles errors generated from referencing a null object.
5	<code>System.DivideByZeroException</code>	Handles errors generated from dividing a dividend with zero.
6	<code>System.InvalidCastException</code>	Handles errors generated during typecasting.
7	<code>System.OutOfMemoryException</code>	Handles errors generated from insufficient free memory.
8	<code>System.StackOverflowException</code>	Handles errors generated from stack overflow.


# Assemblies

It is a collection of code files that are compiled into an executable or a **Dynamic Link Library (DLL)**.

- Assemblies are only loaded into memory if they're required. If they aren't used, they aren't loaded. Therefore, assemblies can be an efficient way to manage resources in larger projects.

# Access Modifiers

**public:** The type or member can be accessed by any other code in the same assembly or another assembly that references it. The accessibility level of public members of a type is controlled by the accessibility level of the type itself.



**private:** The type or member can be accessed only by code in the same class or struct.

**protected:** The type or member can be accessed only by code in the same class, or in a class that is derived from that class.

**internal:** The type or member can be accessed by any code in the same assembly, but not from another assembly. In other words, internal types or members can be accessed from code that is part of the same compilation.

**protected internal:** The type or member can be accessed by any code in the assembly in which it's declared, or from within a derived class in another assembly.

**private protected:** The type or member can be accessed by types derived from the class that are declared within its containing assembly

# Constructors

- Constructor name is same as class name.
- By default C# will create default constructor internally.
- Constructor with no arguments and no body is called default constructor.
- Constructor with arguments is called parameterized constructor.
- Constructor by default public.
- We can create private constructors.
- A method with same name as class name is called constructor there is no separate keyword.
- Constructor will not return Anything

## Types

1. Default Constructor
2. Parameterized Constructor
3. Static Constructor
4. Private Constructor
5. Copy Constructor

## Destructors :

.Net will clean up the un-used objects by using garbage collection process. It internally uses the destruction method to clean up the un-used objects. Some times the programmer needs to do manual cleanup.



# Structures

A Structure is a collection various variables with different variables like Int ,Float Char etc .

```
struct stu
{
    char name[10],lastname[10];
    int age;
    float height;
    funt()
{
}
}
```

Structure is also user defined data type but the Main difference between the Arrays and Structure is that All the Elements of An Array are of Same type but in Structure all the Elements of a Structure are of different data type.

A Structure May be Nested Means we can also define a Structure inside another Structure and Structure is a container of various Elements which have a different name and different data type.

# Difference Between the Structure and Class



- All the Variables in the Structure are value type so that they are stored on the Stack in Memory and in Class Variables are of reference type so that they are stored on the heap of the Memory.
- Classes Support Inheritance but a Structure can't be inherited.
- Classes have both default and Parameters Constructor but a Structure have not a default Constructor.
- When we Assign the Class Variable then the Original Values are Copied but in the Structure only the Values are Copied it doesn't Passes the Original Reference for that Variables.

# Enumerations

Enumerated data type is user defined data type and This is a Special data type which provides a user to opportunity to invent your own data type

Syntax : 


```
enum enum-Name{ valusr  
}
```

Sample:

```
enum days {  
  
    // enum data members  
    monday,  
    tuesday,  
    wednesday,  
    thursday,  
    friday,  
    saturday,  
    sunday  
}
```

# Enumerations

Enumerated data type is user defined data type and This is a Special data type which provides a user to opportunity to invent your own data type

Syntax :   
Enum Name{ valusr  
}

Sample:  
enum days {  
  
    // enum data members  
    monday,  
    tuesday,  
    wednesday,  
    thursday,  
    friday,  
    saturday,  
    sunday  
}

# Types Of Classes



1. Abstract Class
2. Partial Class
3. Sealed Class
4. Static class

# Abstract Class



An abstract class is a class that provides a common definition to the subclasses, and this is the type of class whose object is not created.

## Some key points of Abstract classes are:

- Abstract classes are declared using the abstract keyword.
- We cannot create an object of an abstract class.
- It must be inherited in a subclass if you want to use it.
- An Abstract class contains both abstract and non-abstract methods.
- The methods inside the abstract class can either have an or no implementation.
- We can inherit two abstract classes; in this case, implementation of the base class method is optional.
- An Abstract class has only one subclass.
- Methods inside the abstract class cannot be private.
- If there is at least one method abstract in a class, then the class must be abstract.

# Partial Class



It is a type of class that allows dividing their properties, methods, and events into multiple source files, and at compile time, these files are combined into a single class.

## Some key points of Partial classes are:

- All the parts of the partial class must be prefixed with the partial keyword.
- If you seal a specific part of a partial class, the entire class is sealed, the same as for an abstract class.
- Inheritance cannot be applied to partial classes.
- The classes written in two class files are combined at run time.

# Sealed Class



A Sealed class is a class that cannot be inherited and used to restrict the properties.

**Some key points of Sealed classes are:**

- A Sealed class is created using the sealed keyword.
- Access modifiers are not applied to a sealed class.
- To access the sealed members, we must create an object of the class.



# Static Class



It is the type of class that cannot be instantiated. In other words, we cannot create an object of that class using the new keyword, such that class members can be called directly using their name.

## Some key points of Abstract classes are:

- It was created using the static keyword.
- Only static members are allowed; in other words, everything inside the class must be static.
- We cannot create an object of the static class.
- A Static class cannot be inherited.
- It allows only a static constructor to be declared.
- The static class methods can be called using the class name without creating the instance.

# Abstract class Vs Interface



- Abstract classes can contain implemented methods, while interfaces only contain method signatures.
- Classes can implement multiple interfaces, but they can inherit from only one abstract class.
- Abstract classes can have constructors, while interfaces cannot.
- Abstract classes can have fields and properties, while interfaces can only have properties.
- Abstract classes are typically used for creating a base class for other classes to inherit from, while interfaces are used for defining a contract that classes must implement.

# Sealed Class vs static Class

