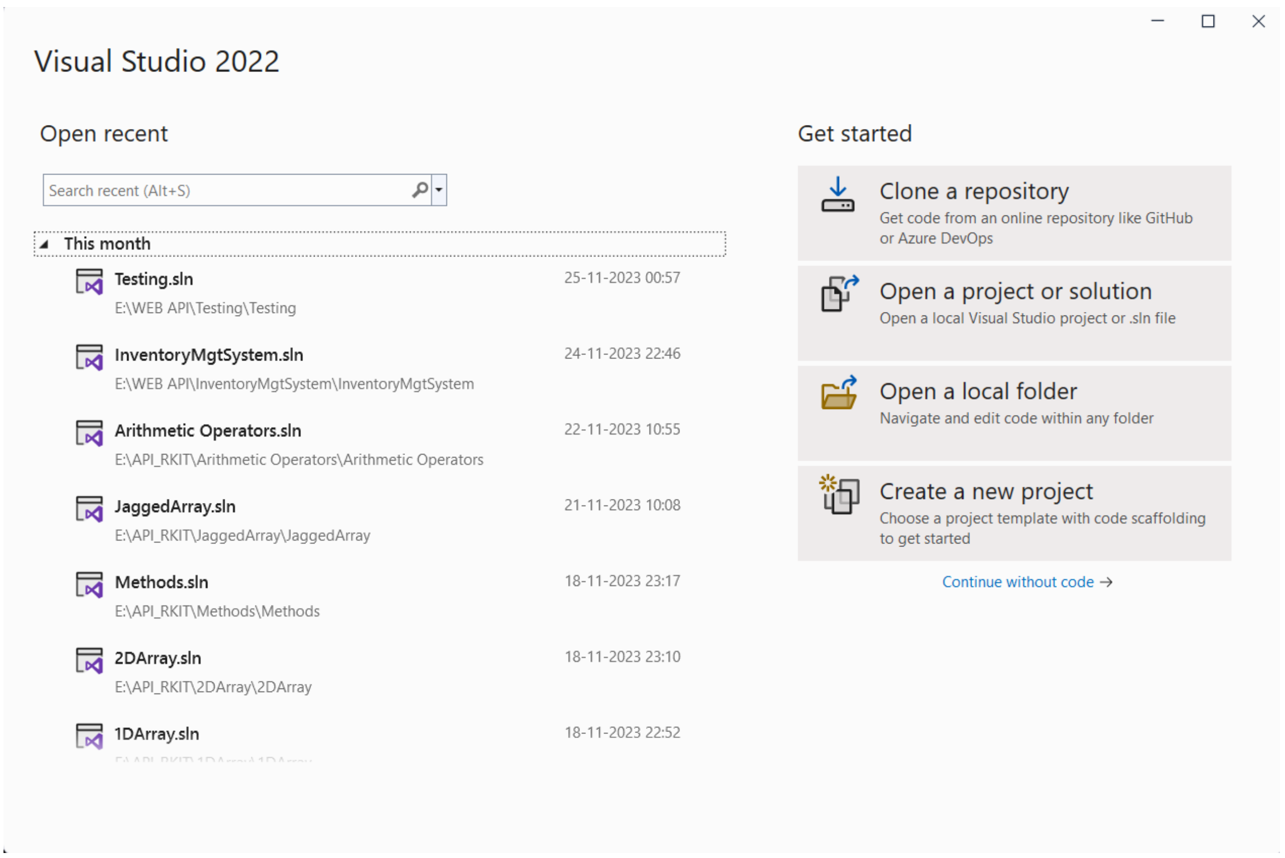
Visual Studio 2022 IDE Overview

Welcome Screen



Visual Studio 2022

Open recent

Search recent (Alt+S)

This month

Testing.sln E:\WEB API\Testing\Testing	25-11-2023 00:57
InventoryMgtSystem.sln E:\WEB API\InventoryMgtSystem\InventoryMgtSystem	24-11-2023 22:46
Arithmetic Operators.sln E:\API_RKIT\Arithmetic Operators\Arithmetic Operators	22-11-2023 10:55
JaggedArray.sln E:\API_RKIT\JaggedArray\JaggedArray	21-11-2023 10:08
Methods.sln E:\API_RKIT\Methods\Methods	18-11-2023 23:17
2DArray.sln E:\API_RKIT\2DArray\2DArray	18-11-2023 23:10
1DArray.sln	18-11-2023 22:52

Get started

Clone a repository
Get code from an online repository like GitHub or Azure DevOps

Open a project or solution
Open a local Visual Studio project or .sln file

Open a local folder
Navigate and edit code within any folder

Create a new project
Choose a project template with code scaffolding to get started

Continue without code →

1. **Clone a Repository**: We can clone the data from version control system.
   - Visual provide 2 types of Version control system.
     1. GitHub: (Microsoft) GitHub is free for public repository with 500 MB Packaging storage.
     2. Azure DevOps: Azure DevOps is free for open source projects and small Projects.

        It's has many advance features to work with.
        - Dashboard Control
        - Improved Source Control
        - Plan and track your work
2. **Open a project or Solution**: By click this we can open the project or solution file of local directories.
3. **Open a local folder**: We can open project folder from this menu.
4. **Create a new Project:** We can create a new project by clicking this menu.

Different types of Windows

1. **Solution Explorer:** Using Solution Explorer Window we can manage, create and update projects files and solution.
2. **Properties Window:** Displays all the properties of the Components.
3. **Team Explorer:** We can work with team in Visual Studio. Visual Studio Provides Two Version Control System. GitHub and Azure DevOps.
4. **Bookmark Window:** Displays all bookmarked lines of Code File Location and Line Number.
5. **Error list:** We can View the error and warnings here.
6. **Git Changes:** Here we can view all the Change done in the Source code.
7. **Toolbox:** Here we can find all the complements which is used to create application interface.
8. **Terminal:** we have executed the commend using this terminal window.

Solution and Project

Solution Explorer

- A Project is contained within a solution. Solution Explorer is a container for one or more related projects along with build information, Visual Studio window settings associated with a particular project.

Solution File

- Visual Studio file types
    1. .sln – Visual Studio Solution

- When we Start Building Website or application in visual studio, first we need to create projects.
- Project Contains All files. which is Executable.
- Files includes source code, icons, images, data files etc.
- Visual Studio uses **MS Build** to build each project in a solution.
- The Microsoft Build Engine (MS Build) is a platform for building application.
- It provides XML schema for a project file.
- Types of Projects – C# Project (.csproj), Visual Basic Project (.vbproj), or a database project (.dbproj).

Code editor features

Git (Version Control):

- Using Version Control System, it’s is easy to work with Team Projects.
- Visual Studio Provides Inbuilt Version control system so it’s is easy to Work with. No Need to type commands.

Line Number:

- Now When we have built the large project and one single file has a thousand line of code then it is good to have a line number.
- Visual Studio also provide to move at particular line number.

Format Document:

- It’s is Good Practical’s to have a proper structure code.
- When we have a clean written code than it’s is good to work with.
- Visual provide a feature to format our code in a proper style.

Syntax Colouring:

- Visual Studio colour the code in a proper format. All syntax and keywords are coloured in blur So it is good to identify the code structure.
- Bracket Matching:
- In Visual Studio, All the Brackets are matches from open to close.
- So it’s is Good to identify the Brackets of Code.

Shortcuts

Key	Use
Format Document	Ctrl + K
New Project	Ctrl + Shift + N
Open Project	Ctrl + Shift + O
New File	Ctrl + N
Open File	Ctrl + O
Save As	Ctrl + Shift + S
View in Browser	Ctrl + Shift + W
Complete Word (get Suggestion)	Ctrl + Spacebar
Find word	Ctrl + F
Find and Replace Word	Ctrl + H
Save the File	Ctrl + S
Cut line	Ctrl + X
Copy Line	Ctrl + C
Move Line	Alt + (UP/Down)
Scroll Line	Ctrl + (UP/Down)
Go To Line	Ctrl + G
Rename	F2
Undo	Ctrl + Z
Redo	Ctrl + Y
Duplicate	Ctrl + D
Next tab	Ctrl + Tab
Find in Files	Ctrl + Shift + F
Breakpoint	F9
Debug Start / Continue	F5
Debug Stop	Shift + F5

Understanding C# Program

- Generally, Programs Contains following Parts

Namespace declaration
Class
Class Methods
Class Attributes
Statements and Expressions
Comment

- Now in C# Programs is Executing Task in Sequence manner
- Comments are not considered at the time of application.

Understanding Syntax

- C# follow the standard structure for the code.
- C# is a Cass-Sensitive Language.
- It Follow the Below Structure.

Imports / namespace	Using keyword is use to declare namespace.
Declare namespace	Namespace declare for the class
Class declare	Class declare with proper name
Methods of class	Other methods of class
Main() Method	Main method is the method by which program execute.

Create First C# Program "Hello World"

What is Namespace?

- namespaces are used to logically arrange classes, structs, interfaces, enums and delegates.
- The namespaces can be nested. That means one namespace can contain other namespaces also.
- The .NET framework already contains number of standard namespaces like System, System.Net, System.IO etc.
- In addition to these standard namespaces the user can define their own namespaces.

Declaring a Namespace

Syntax:

```
namespace <namespace_name>
{
        // Classes and/or structs and/or enums etc.
}
```

- **Note: namespace** keyword is used to define namespace
- It is not possible to use any access specifiers like private, public etc with a namespace declaration.
- The namespaces in C# are implicitly have public access and this is not modifiable.
- Default it provide internal access.

Creating Aliases

- Developers can create Aliases of the namespace.

Example:

```
using con = System.Console; // Create an alias
class MyClient
{
        public static void Main()
        {
                con.WriteLine("namespace demo");
        }
}
```

Standard Namespaces in .NET

- System: Contain classes that implement basic functionalities like mathematicaloperations, data conversions etc.
- System.IO: Contains classes used for file I/O operations.
- System.Net: Contains class wrappers around underlying network protocols.
- System.Collections: Contains classes that implement collections of objects such aslists, hashtable etc.
- System.Data: Contains classes that make up ADO.NET data access architecture.System,Drawing: Contains classes that implement GUI functionalities.
- System.Threading: Contains classes that are used for multithreading programming.System.Web: Classes that implement HTTP protocol to access web pages.
- System.Xml: Classes that are used for processing XML data.

We can use namespace using two methods.

1. Dot(.) Operator
2. Using directive

What is class?

- A class is like a **blueprint** of a specific object.
- In real world every object has some colour, shape and some functionalities etc.
- It has also some characteristics like speed, colour etc.
- So Group of some this all things is called class.
- A class enables you to create your custom types by grouping variables of the other types, methods and events.
- Class can be defined in **class** keyword.

Access Modifier

Public: Public modifier allows any part of the program in the same assembly or another assembly to access the type and its members.

Private: Private modifier restricts other parts of the program from accessing the type and its members. Only code in the same class or struct can access it.

Internal: Internal modifier allows other program code in the same assembly to access the type or its members. This is default access modifiers if no modifier is specified.

Protected: Protected modifier allows codes in the same class or a class that derives from that class to access the type or its members.

Constructor

- Class have parameterized or parameter less constructors.
- Constructor will be called when you create an instance of a class.
- Constructor will be defined by using as access modifier.

Syntax:

```
class MyClass
{
   public MyClass()
   {

   }
}
```

Variable & Method Declaration

Variables:

- variable contains a data value of the specific data type

Syntax:

```
<data type> <variable name> = <value>;
```

Example:

```
int no = 10;
```

- There are some rules to declare variables name.
- Variable names must be unique.
- Variable names can contain letters, digits, and the underscore _ only.
- Variable names must start with a letter.
- Variable names are case-sensitive, num and Num are considered different names.
- Variable names cannot contain reserved keywords. Must prefix @ before keyword if want reserve keywords as identifiers.

Method

- A program causes the statements to be executed by calling the method.
- Method is use redenied code.
- A method can also perform some specific task without returning anything.

Syntax:

{access modifier} {return type} MethodName({parameterType parameterName})

- Method have some elements explain below
- Return Type: We have to specify the return type of the method.
- Method Body: This is the actual part of code which is execute when method called.
- Parameters Body: We have to specify the method parameters.
- Access Specifier: set the visibility of a variable or a method from another class.

Program Flow

Working with code files, projects & solutions

Understanding structure of solution

Solutions:

- A Solution contains a collection of projects, along with information on dependencies between those projects.
- The project themselves contain files.
- We can Create many projects in solution
- We can one open one solution at a time in a particular instance of visual studio.

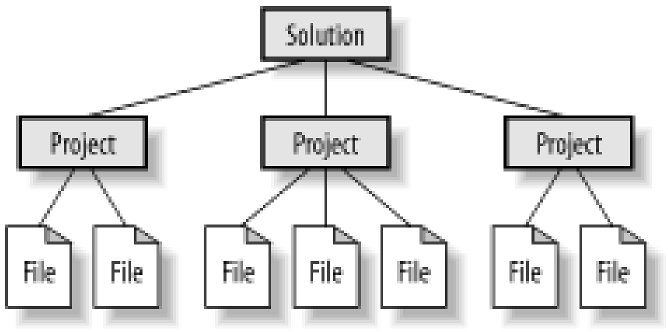


Figure show the Solution structure

- One project can have multiple solutions.

Understanding structure of Project

Windows App:

- Windows application is used widely.
- There are many architectural for windows application.

Controls:

- In the Windows application there are many controls like textbox, edit text menu etc.
- This are used to complete the functionally of users.

Forms: This is the actually pages of Windows Application

Dependencies:

- Here we can find the projects dependencies.

Program.cs:

- This is the Main Entry Point of the application.

Web Applications:

- Using Web application, we can build a website.
- In the Website Project there is many folders.

Appsettings.json:

- Appsettings json file is an application configuration file used to store configuration settings such as database connections strings.
- Using this file, we can declare the value globally.

Dependencies:

- Here we can find the projects dependencies.

Wwwroot folder:

- CSS: this folder has the css file of the website.
- JS: This folder has the JavaScript files of the website.

Project.json file:

- It contains the information about the project.

Familiar with different type of file extensions

Name	Use
.html,.htm	Html web file
.asp	Active server page file
.css	Cascading Style Sheets
.txt	A text file
.sln	Solution file
.csproj	C# Project
.png	A image file

Understanding datatypes & variables with conversion

Base Datatypes

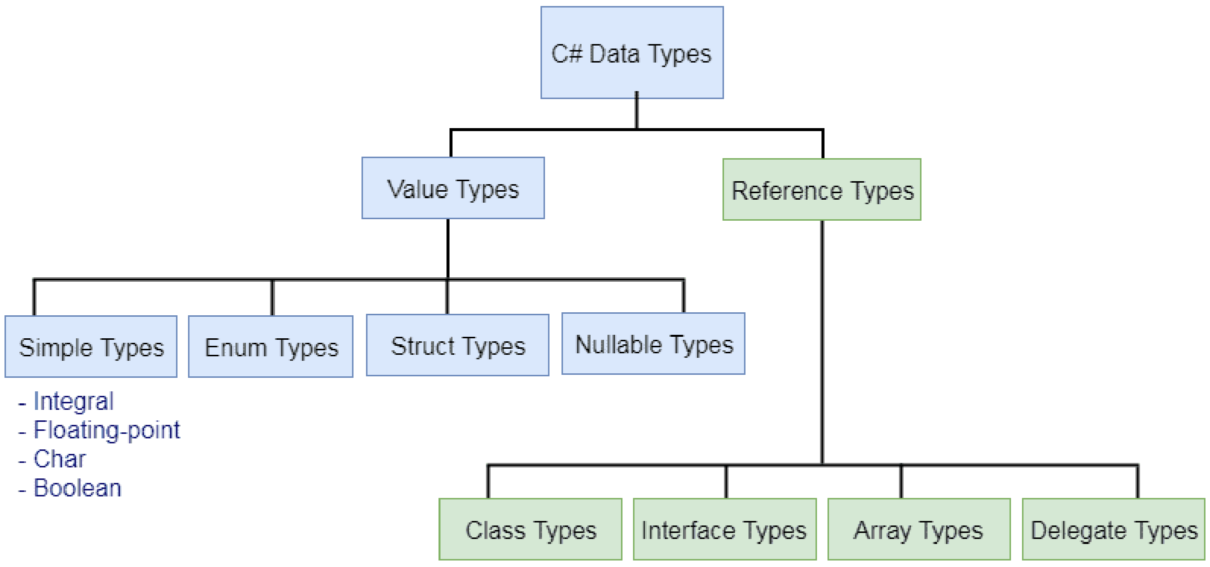
- C# is a strongly-typed language.
- It means we must declare the type of a variable that indicates that kind of values.
- Like int, float, text, etc.
- C# has two type of data types (value and reference type)

Value Datatype:

- Value datatype store the data itself.
- Value types include simple types such as int, float, bool.

Reference Type:

- Reference type doesn't store its vale directly.
- It stores the address where the value is being stored.
- Reference type contains a pointer to another memory location that holds the data.



Datatype Conversion

- C# has a two type of Conversion of datatype.
- It is also known as type Casting.
- Two types are Implicit Type Conversion and Explicit Type Conversion.

1. Implicit Type Conversion:

- These conversions are performed by C# in a type-safe manner.
- From the smaller to larger integral types and conversions from derived classes to base classes.

2. Explicit Type Conversion:

- These conversions are done explicitly by users using the predefined functions.
- Explicit Conversion require a cast operator.

Methods	Description
ToBoolean	Converts a type to a Boolean value, where possible.
ToByte	Converts a type to a byte
ToChar	Converts a type to a single Unicode character, where possible
ToDateTime	Converts a type (integer or string type) to date-time structures.
ToDecimal	Converts a floating point or integer type to a decimal type.
ToDouble	Converts a type to a double type
ToInt16	Converts a type to a 16-bit integer.
ToInt32	Converts a type to a 32-bit integer.
ToInt64	Converts a type to a 64-bit integer.
ToString	Converts a type to a string.

Boxing and Unboxing

- In C#, there are two Kinds of datatype.
  Value type (int, double, decimal).
  Reference type (object, string, array).
- **Value Type:** value type stores the value itself.
- **Reference Type:** reference type stores the address of the value where it is stored.

What is Boxing?

- Process of converting a value type to the object type.
- Boxing is implicit.

Example:

```
int i = 10;

object o = i ;
```

- In the Above Example we have did Boxing. Now Object O store the address of the variable i.
- So This is called as Boxing.
- Now its Name is Boxing because it's store the address in heap and actual value is store in the stack So heap wrap the value and box on this value and address is store in the heap so It's is Called Boxing.

What is Unboxing?

- Unboxing is the reverse of boxing.
- It is the process of converting a reference type to value type.
- Unboxing is Explicit. It's Means we need to cast explicitly.

Example:

```
Int I = 10

Object o = i;

Console.WriteLine((int)0);
```

- **Note: Boxing and Unboxing degrade the performance.**

If-else & Switch Statement

If-Else:

- We can get Many decision-making statements in C#
- like if, else if and else
- below is the syntax.

If Statement

```
if(condition)
{
    // code block to be executed when if condition evaluates to true
}
```

If Else Statement

```
if(condition1)
{
    // code block to be executed when if condition1 evaluates to true
}
else if(condition2)
{
    // code block to be executed when
    //      condition1 evaluates to flase
    //      condition2 evaluates to true
}
```

Example:

```
int i = 10;


if (i > 3)
{
   Console.WriteLine("i is greater than 3");
}

if (i >= 10)
{
   Console.WriteLine("i is greater than or equal to 10");
}
```

OPERATORS AND EXPRESSIONS

TERNARY OPERATOR

- DECISION MAKING OPERATOR?**:** WHICH IS CALLED THE CONDITIONAL OPERATOR OR TERNARY OPERATOR
- **SYNTAX:** CONDITION **?** STATEMENT 1(TRUE) : STATEMENT 2(FALSE)
- IF CONDITION BECOME TRUE, THEN STATEMENT 1 EXECUTE ELSE STATEMENT 2 EXECUTE.

NESTED TERNARY OPERATOR

- IN THE NESTED TERNARY OPERATOR, WE MUST PASS SECOND CONDITIONAL EXPRESSION.
- **SYNTAX:** CONDITION **? CONDITION ? STATEMENT 1(TRUE) : STATEMENT 2(FALSE)** : STATEMENT 2(FALSE)

INCREMENT AND DECREMENT OPERATORS

- INCREMENT OPERATOR IS USED TO INCREMENT THE VALUE BY 1.
- DECREMENT OPERATOR IS USED TO DECREMENT THE VALUE BY 1.

LOOP ITERATION

FOR LOOP

- FOR LOOP IS USED TO EXECUTE BLOCK OF CODE MULTIPLE TIMES.
- NOW TO USE FOR LOOP WE HAVE ONE

  KEYWORD:

  FOR SYNTAX:

```
FOR (INITIALIZER; CONDITION; ITERATOR)
{
//CODE
}
```

INITIALIZER:

- USED TO INITIALIZE A VARIABLE THAT WILL BE LOCAL TO A FOR LOOP.
- IT CAN ALSO BE ZERO OR MORE ASSIGNMENT STATEMENTS

CONDITION:

- THE CONDITION IS A BOOLEAN EXPRESSION THAT WILL RETURN EITHER TRUE OR FALSE.
- IF AN EXPRESSION EVALUATES TO TRUE, THEN IT WILL EXECUTE THE LOOP AGAIN; OTHERWISE, THE LOOP ISEXITED.

ITERATOR:

- THE ITERATOR DEFINES THE INCREMENTAL OR DECREMENTAL OF THE LOOP VARIABLE.

FOREACH LOOP

- THE FOREACH LOOP IS USED TO ITERATE OVER THE ELEMENTS OF THE COLLECTION.
- THE COLLECTION MAY BE AN ARRAY OR A LIST.
- IT EXECUTES FOR EACH ELEMENT PRESENT IN THE ARRAY.

SYNTAX:

```
FOREACH(DATA_TYPE VAR_NAME IN COLLECTION_VARIABLE)

{

   // CODE

}
```

WHILE LOOP

- EXECUTE THE BLOCK OF CODE UNTIL THE SPECIFY CONDITION NOT BECOME FALSE.

SYNTAX:

```
WHILE(CONDITION)

{

//CODE

}
```

- WHILE LOOP START WITH **WHILE** KEYWORD.
- FOR LOOP CONTAINS THE INITIALIZATION PART BUT, IN THE WHILE, LOOP WE HAVE TO INITIALIZATION BEFOREUSE.
- TO BREAK THE LOOP, WE CAN USE THE BREAK KEYWORD.

DO-WHILE LOOP

- DO WHILE LOOP EXECUTE THE CODE BEFORE THE CONDITION CHECK.
- IT MEANS FIRST CODE WILL BE EXECUTING AND AT THE END IT WILL CHECK THE CONDITION.
- IT WILL EXECUTE THE BLOCK UNTIL CONDITION BECOME FALSE.

SYNTAX:

```
DO {

//CODE

} WHILE(CONDITION);
```

- DO-WHILE LOOP START WITH **DO** KEYWORD.

NOTE: ABOVE MENTION LOOPS CAN BE NESTED.

BREAK STATEMENT

- THE BREAK STATEMENT IS USED TO TERMINATE THE LOOP OR STATEMENT.
- AFTER THAT, THE CONTROL WILL PASS TO THE STATEMENTS THAT PRESENT AFTER THE BREAK STATEMENT.
- IF THE BREAK STATEMENT PRESENTS IN THE NESTED LOOP, THEN IT TERMINATES ONLY THOSE LOOPSWHICH CONTAINS BREAK STATEMENT.

SYNTAX: BREAK;

- USING **BREAK** KEYWORD, WE CAN BREAK THE LOOP OR STATEMENTS.

CONTINUE STATEMENT

- THIS STATEMENT IS USED TO SKIP OVER THE EXECUTION PART OF THE LOOP ON A CERTAIN CONDITION.
- AFTER THAT, IT TRANSFERS THE CONTROL TO THE BEGINNING OF THE LOOP.
- BASICALLY, IT SKIPS ITS FOLLOWING STATEMENTS AND CONTINUES WITH THE NEXT ITERATION OF THE LOOP.

SYNTAX: CONTINUE;

- USING **CONTINUE** KEYWORD, WE CAN CONTINUE THE NEXT ITERATION OF LOOP.

GOTO STATEMENT

- THIS STATEMENT IS USED TO TRANSFER CONTROL TO THE LABELLED STATEMENT IN THE PROGRAM.
- THE LABEL IS THE VALID IDENTIFIER AND PLACED JUST BEFORE THE STATEMENT FROM WHERE THE CONTROLIS TRANSFERRED.

UNDERSTANDING ARRAYS

- AN ARRAY IS A GROUP OF LIKE-TYPED VARIABLES THAT ARE REFERRED TO BY A COMMON NAME. AND EACHDATA ITEM IS CALLED AN ELEMENT OF THE ARRAY.
- THE DATA TYPES OF THE ELEMENTS MAY BE ANY VALID DATA TYPE LIKE CHAR, INT, FLOAT, ETC.
- THE ELEMENTS ARE STORED IN A SEQUENCE.
- LENGTH OF THE ARRAY SPECIFIES THE NUMBER OF ELEMENTS PRESENT IN THE ARRAY.
- IN C# THE ALLOCATION OF MEMORY FOR THE ARRAYS IS DONE DYNAMICALLY.
- AN ARRAYS ARE KIND OF OBJECTS; THEREFORE, IT IS EASY TO FIND THEIR SIZE USING THE PREDEFINEDFUNCTIONS.
- THE VARIABLES IN THE ARRAY ARE ORDERED AND EACH HAS AN INDEX BEGINNING FROM 0.
- A C# ARRAY VARIABLE CAN ALSO BE DECLARED LIKE OTHER VARIABLES WITH [] AFTER THE DATA TYPE.
- C# ARRAY IS AN OBJECT OF BASE TYPE SYSTEM.ARRAY.
- A JAGGED ARRAY ELEMENTS ARE REFERENCE TYPES AND ARE INITIALIZED TO NULL.
- ARRAY ELEMENTS CAN BE OF ANY TYPE, INCLUDING AN ARRAY TYPE.

SYNTAX :

< DATA TYPE > [ ] < NAMEARRAY >

- HERE FIRST WE MUST SPECIFY THE DATA TYPE ALONG WITH THE BRACKETS [].
- AFTER THAT WE MUST SPECIFY THE NAME OF THE ARRAY.

IN C# THERE ARE THREE TYPES OF ARRAY.

1. ONE DIMENSIONAL ARRAY
2. MULTIDIMENSIONAL ARRAYS
3. JAGGED ARRAYS

ONE DIMENSIONAL ARRAY

- IN THIS ARRAY CONTAINS ONLY ONE ROW FOR STORING THE VALUES.
- ALL VALUES OF THIS ARRAY ARE STORED CONTIGUOUSLY STARTING FROM 0 TO THE ARRAY SIZE.
- **EXAMPLE:** INT[] ARRAYINT = NEW INT[2];

MULTIDIMENSIONAL ARRAYS

- THE MULTI-DIMENSIONAL ARRAY CONTAINS MORE THAN ONE ROW TO STORE THE VALUES.
- IT IS ALSO KNOWN AS A RECTANGULAR ARRAY IN C# BECAUSE IT'S EACH ROW LENGTH IS SAME.
- IT CAN BE A 2D-ARRAY OR 3D-ARRAY OR MORE.
- TO STORING AND ACCESSING THE VALUES OF THE ARRAY, ONE REQUIRED THE NESTED LOOP.
- **EXAMPLE:** INT[,] INTARRAY = NEW INT[4, 2];

<u>JAGGED ARRAYS</u>

- AN ARRAY WHOSE ELEMENTS ARE ARRAYS IS KNOWN AS JAGGED ARRAYS IT MEANS "ARRAY OF ARRAYS ".
- THE JAGGED ARRAY ELEMENTS MAY BE OF DIFFERENT DIMENSIONS AND SIZES.
- IT'S POSSIBLE TO MIX JAGGED AND MULTIDIMENSIONAL ARRAYS.
- **EXAMPLE:** INT[][] ARR1 = { NEW INT[] { 1, 3, 5, 7, 9 }, NEW INT[] { 2, 4, 6, 8 } };

DEFINING AND CALLING METHODS

- METHODS ARE GENERALLY THE BLOCK OF CODES OR STATEMENTS IN A PROGRAM THAT GIVES THE USER THEABILITY TO REUSE THE SAME CODE WHICH ULTIMATELY SAVES THE EXCESSIVE USE OF MEMORY, ACTS AS A TIME SAVE.
- MORE IMPORTANTLY, IT PROVIDES A BETTER READABILITY OF CODE.
- SO BASICALLY, A METHOD IS A COLLECTION OF STATEMENTS THAT PERFORM SOME SPECIFIC TASK ANDRETURN THE RESULT TO THE CALLER. A METHOD CAN ALSO PERFORM SOME SPECIFIC TASK WITHOUT RETURNING ANYTHING.

METHOD DECLARATION

METHOD DECLARATION MEANS THE WAY TO CONSTRUCT METHOD INCLUDING ITS NAMING.

SYNTAX:

- <ACCESS_MODIFIER> <RETURN_TYPE> <METHOD_NAME>([<PARAM_LIST>])

METHOD CALLING

- METHOD INVOCATION OR METHOD CALLING IS DONE WHEN THE USER WANTS TO EXECUTE THE METHOD.
- THE METHOD NEEDS TO BE CALLED FOR USING ITS FUNCTIONALITY.
    - A METHOD RETURNS TO THE CODE THAT INVOKED IT WHEN:
    - IT COMPLETES ALL THE STATEMENTS IN THE METHOD
    - IT REACHES A RETURN STATEMENT
    - THROWS AN EXCEPTION

METHOD PARAMETERS

- THERE MIGHT BE CERTAIN SITUATIONS THE USER WANTS TO EXECUTE A METHOD BUT SOMETIMES THATMETHOD REQUIRES SOME VALUE INPUTS IN ORDER TO EXECUTE AND COMPLETE ITS TASKS.
- THESE INPUT VALUES ARE KNOWN AS PARAMETERS IN A COMPUTER LANGUAGE TERMS.
- NOW, THESE PARAMETERS CAN BE EITHER INT, LONG OR FLOAT OR DOUBLE OR CHAR. HOWEVER, ITDEPENDS UPON THE USER REQUIREMENTS.
- THE METHODS IN C# CAN BE CLASSIFIED INTO DIFFERENT CATEGORIES BASED ON RETURN TYPE AS WELL ASINPUT PARAMETERS.

ADVANTAGES OF USING THE METHODS:

THERE ARE MANY ADVANTAGES OF USING METHODS. SOME OF THEM ARE LISTED BELOW:

- IT MAKES THE PROGRAM WELL STRUCTURED.
- METHODS ENHANCE THE READABILITY OF THE CODE.
- IT PROVIDES AN EFFECTIVE WAY FOR THE USER TO REUSE THE EXISTING CODE.
- IT OPTIMIZES THE EXECUTION TIME AND MEMORY SPACE.

Understanding Classes

- A class is like a blueprint of a specific object.
- Class and Object are the basic concepts of Object-Oriented Programming which revolvearound the real-life entities.
- Basically, a class combines the fields and methods (member function which defines actions)into a single unit.
- In C#, classes support polymorphism, inheritance and also provide the concept of derivedclasses and base classes.

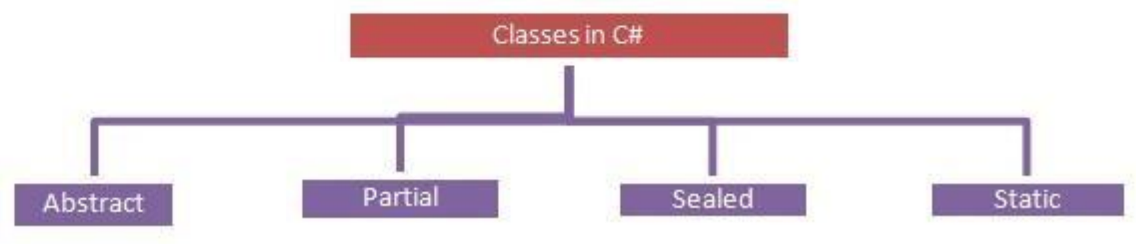
Declaration of class

- Generally, a class declaration contains only keyword class, followed by an identifier(name) ofthe class.
- class declarations can include these components – Modifiers, Keyword class, Base Class,Interfaces and Body.

Some Key points about classes

- Classes are reference types that hold the object created dynamically in a heap.
- All classes have a base type of System.Object.
- The default access modifier of a class is Internal.
- The default access modifier of methods and variables is Private.
- Directly inside the namespaces declarations of private classes are not allowed.

Types of 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.

- Abstract classes are declared using the abstract keyword.
- We cannot create an object of an abstract class.
- If you want to use it then it must be inherited in a subclass.
- An Abstract class contains both abstract and non-abstract methods.
- The methods inside the abstract class can either have an implementation or no implementation.

- We can inherit two abstract classes; in this case the base class method implementation 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.

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

sealed Class:

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

- properties.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 class name.

- Created using the static keyword.
- Inside a static class only static members are allowed, in other words everything inside the static 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 methods of the static class can be called using the class name without creating the instance.

Constructor

- Constructor of a class must have the same name as the class name in which it resides.
- A constructor cannot be abstract, final, and Synchronized.
- Within a class, you can create only one static constructor.
- A constructor doesn’t have any return type, not even void.
- A static constructor cannot be a parameterized constructor.
- A class can have any number of constructors.

Types of Constructor

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

Default Constructor

- A constructor with no parameters is called a default constructor.
- The default constructor initializes all numeric fields to zero and all string and object fields to null inside a class.

Parameterized Constructor

- A constructor having at least one parameter is called as parameterized constructor.
- It can initialize each instance of the class to different values.

Copy Constructor

- This constructor creates an object by copying variables from another object.
- Its main use is to initialize a new instance to the values of an existing instance.

Private Constructor

- If a constructor is created with private specifier is known as Private Constructor.
- It is not possible for other classes to derive from this class and also it’s not possible to create an instance of this class.
- It is the implementation of a singleton class pattern.
- use private constructor when we have only static members.
- Using private constructor, prevents the creation of the instances of that class.

Static Constructor

- Static Constructor has to be invoked only once in the class and it has been invoked during the creation of the first reference to a static member in the class.
- A static constructor is initialized static fields or data of the class and to be executed only once.
- It can’t be called directly.
- When it is executing then the user has no control.
- It does not take access modifiers or any parameters.
- It is called automatically to initialize the class before the first instance created.

<u>Object</u>:

- Object is a real world entity, for example, chair, car, pen, mobile, laptop etc.
- In other words, object is an entity that has state and behaviour. Here, state means data and behaviour means functionality.
- Object is a runtime entity; it is created at runtime.
- Object is an instance of a class. All the members of the class can be accessed through object.

<u>Properties:</u>

- Property in C# is a member of a class that provides a flexible mechanism for classes to expose private fields.
- Properties are an extension of fields and are accessed using the same syntax.
- They use accesses through which the values of the private fields can be read, written or manipulated.
- Properties do not name the storage locations. Instead, they have accesses that read, write, or compute their values.

<u>Events:</u>

- Events are user actions such as key press, clicks, mouse movements, etc., or some occurrence such as system generated notifications. Applications need to respond to events when they occur. For example, interrupts.
- Events are used for inter-process communication.

<u>Using Delegates with Events</u>

- The events are declared and raised in a class and associated with the event handlers using delegates within the same class or some other class. The class containing the event is used to publish the event. This is called the publisher class. Some other class that accepts this event is called the subscriber class. Events use the publisher-subscriber model.
- A publisher is an object that contains the definition of the event and the delegate. The event-delegate association is also defined in this object. A publisher class object invokes the event and it is notified to other objects.
- A subscriber is an object that accepts the event and provides an event handler. The delegate in the publisher class invokes the method (event handler) of the subscriber class.

<u>Points to Publishing an Event</u>

1. First need to declare a delegate, that is the contract between the publisher and the subscriber.
2. Define the event based on the delegate.
3. Publish the event.

Interface

- In the human world, a contract between the two or more humans binds them to act as per the contract.
- In the same way, an interface includes the declarations of related functionalities.
- The entities that implement the interface must provide the implementation of declared functionalities.
- In C#, an interface can be defined using the interface keyword.
- An interface can contain declarations of methods, properties, indexers, and events. However, it cannot contain fields, auto-implemented properties.

Example

```
interface IFile
{
    void ReadFile();
    void WriteFile(string text);
}
```

- You cannot apply access modifiers to interface members.
- All the members are public by default.
- If you use an access modifier in an interface, then the C# compiler will give a compile-timeerror "The modifier 'public/private/protected' is not valid for this item.".

Example

```
interface IFile
{
    protected void ReadFile(); //compile-time error
    private void WriteFile(string text);//compile-time error
}
```

Note: An interface can only contain declarations but not implementations.

Implementing an Interface

- A class or a Struct can implement one or more interfaces using colon (:).
- **Syntax**: <Class or Struct Name> : <Interface Name>

Note: Interface members must be implemented with the public modifier; otherwise, the compiler will give compile-time errors.

Explicit Implementation

- An interface can be implemented explicitly using <InterfaceName>.<MemberName>.
- Explicit implementation is useful when class is implementing multiple interfaces; thereby, it is more readable and eliminates the confusion.
- It is also useful if interfaces have the same method name coincidently.
- When you implement an interface explicitly, you can access interface members only through the instance of an interface type.

Note: Do not use public modifier with an explicit implementation. It will give a compile-time error.

Implementing Multiple Interfaces

- A class or struct can implement multiple interfaces. It must provide the implementation of all the members of all interfaces.
- It is recommended to implement interfaces explicitly when implementing multiple interfaces to avoid confusion and more readability.

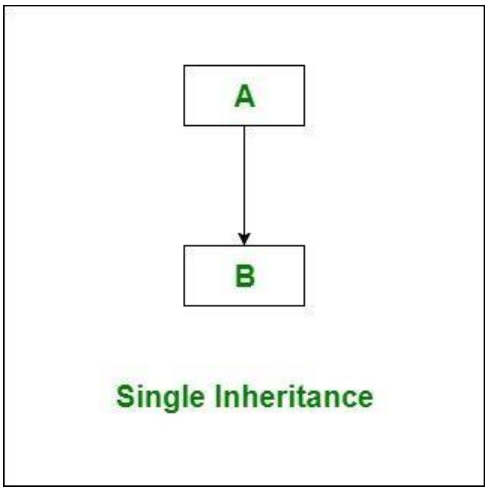
Points to Remember:

1. Interface can contain declarations of method, properties, indexers, and events.
2. Interface cannot include private, protected, or internal members. All the members are public by default.
3. Interface cannot contain fields, and auto-implemented properties.
4. A class or a struct can implement one or more interfaces implicitly or explicitly. Use public modifier when implementing interface implicitly, whereas don't use it in case of explicit implementation.
5. Implement interface explicitly using InterfaceName.MemberName.
6. An interface can inherit one or more interfaces.

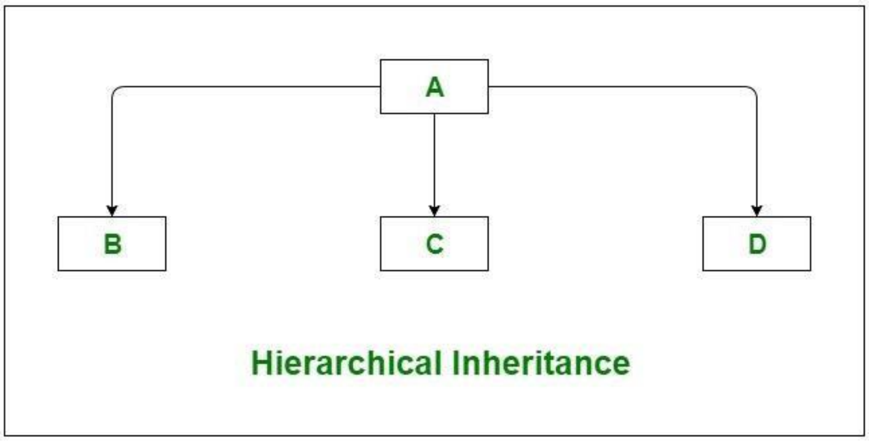
Inheritance

- Acquiring (taking) the properties of one class into another class is called inheritance.Inheritance provides reusability by allowing us to extend an existing class.
- The reason behind OOP programming is to promote the reusability of code and to reducecomplexity in code and it is possible by using inheritance.
- The following are the types of inheritance in C#.

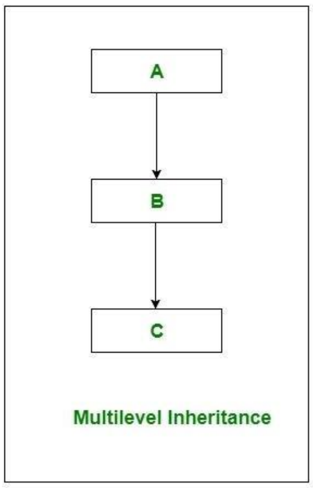
1. single



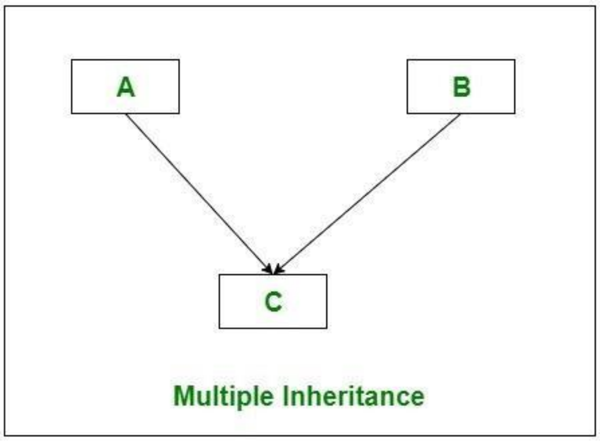
2. hierarchical



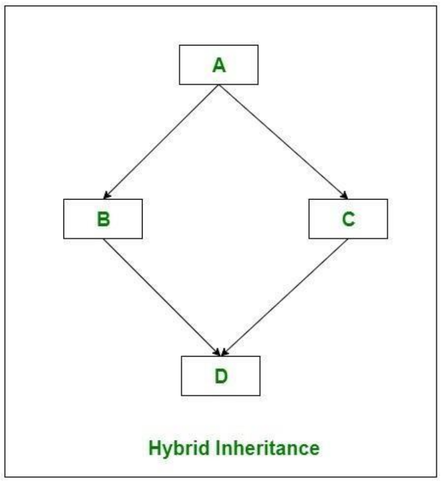
3. Multi level



4. Multiple (using interface)



5. Hybird (using interface)



- The inheritance concept is based on a base class and derived class. Let us see the definitionof a base and derived class.

Base class - is the class from which features are to be inherited into another class.
Derived class - it is the class in which the base class features are inherited.

Scope & Accessibility Modifiers

- Access Modifiers are keywords that define the accessibility of a member, class or datatype ina program.
- These are mainly used to restrict unwanted data manipulation by external programs orclasses.
- There are 4 access modifiers (public, protected, internal, private) which defines the 6accessibility levels as follows:

Access Modifiers	Public	Protected	Internal	Protected Internal	Private	Private Protected
Entire Program	YES	NO	NO	NO	NO	NO
Containing Class	YES	YES	YES	YES	YES	YES
Current Assembly	YES	NO	YES	YES	NO	NO
Derived Types	YES	YES	NO	YES	NO	NO
Derived types within current assembly	YES	YES	YES	YES	No	YES

Public:

- Access is granted to the entire program.
- This means that another method or another assembly which contains the class reference can access these members or types.
- This access modifier has the most permissive access level in comparison to all other access modifiers.

Protected:

- Access is limited to the class that contains the member and derived types of this class.
- It means a class which is the subclass of the containing class anywhere in the program can access the protected members.

Internal:

- Access is limited to only the current Assembly, that is any class or type declared as internal is accessible anywhere inside the same namespace.
- It is the default access modifier in C#.

Note: In the same code if you add another file, the class Complex will not be accessible in that namespace and compiler gives an error.

Private:

- Access is only granted to the containing class.
- Any other class inside the current or another assembly is not granted access to these members.

Private Protected:

- Access is granted to the containing class and its derived types present in the current assembly.
- This modifier is valid in C# version 7.2 and later

Namespace & .Net Library

What is Namespace?

- namespaces are used to logically arrange classes, structs, interfaces, enums and delegates.
- The namespaces can be nested. That means one namespace can contain other namespaces also.
- The .NET framework already contains number of standard namespaces like System, System.Net, System.IO etc.
- In addition to these standard namespaces the user can define their own namespaces.

Declaring a Namespace

Syntax:

```
namespace <namespace_name>
{
        // Classes and/or structs and/or enums etc.
}
```

- **Note: namespace** keyword is used to define namespace

It is not possible to use any access specifiers like private, public etc with a namespace declaration.
The namespaces in C# are implicitly have public access and this is not modifiable.Default it provide internal access.

Creating Aliases

- **Developers can create Aliases of the namespace.**

Example:

```
using con = System.Console; // Create an alias
class MyClient
{
        public static void Main()
        {
                con.WriteLine("namespace demo");
        }
}
```

Standard Namespaces in .NET

- System: Contain classes that implement basic functionalities like mathematical operations, data conversions etc.
- System.IO: Contains classes used for file I/O operations.
- System.Net: Contains class wrappers around underlying network protocols.
- System.Collections: Contains classes that implement collections of objects such as lists, hashtable etc.
- System.Data: Contains classes that make up ADO.NET data access architecture.
- System,Drawing: Contains classes that implement GUI functionalities.
- System.Threading: Contains classes that are used for multithreading programming.
- System.Web: Classes that implement HTTP protocol to access web pages.
- System.Xml: Classes that are used for processing XML data.

Creating and adding ref. to assemblies

- An Assembly is a basic building block of .Net Framework applications.
- It is basically a compiled code that can be executed by the CLR.
- An assembly is a collection of types and resources that are built to work together and form a logical unit of functionality.
- An Assembly can be a DLL or exe depending upon the project that we choose.

  Assemblies are basically the following two types:

1. Private Assembly
2. Shared Assembly

Private Assembly

- It is an assembly that is being used by a single application only.
- Suppose we have a project in which we refer to a DLL so when we build that project that DLL will be copied to the bin folder of our project.
- That DLL becomes a private assembly within our project. Generally, the DLLs that are meant for a specific project are private assemblies.

Shared Assembly

- Assemblies that can be used in more than one project are known to be a shared assembly.
- Shared assemblies are generally installed in the GAC. Assemblies that are installed in the GAC are made available to all the .Net applications on that machine.

Working with Collections

- C# collection types are designed to store, manage and manipulate similar data more efficiently.
- Data manipulation includes adding, removing, finding, and inserting data in the collection.
- Adding and inserting items to a collection
- Removing items from a collection
- Finding, sorting, searching items
- Replacing items
- Copy and clone collections and items
- Capacity and Count properties to find the capacity of the collection and number of items in the collection

NET supports two types of collections.

1. Generic Collection
2. Non-Generic Collection

- Generic collections with work generic data type.
- In non-generic collections, each element can represent a value of a different type. The collection size is not fixed. Items from the collection can be added or removed at runtime.
- **Non-generic**: ArrayList, HashTable, SortedList, Stack, Queue
- **Generic:** List, Dictionary, SortedList, Stack, Queue.

Enumerations

- In C#, an enum (or enumeration type) is used to assign constant names to a group ofnumeric integer values.
- It makes constant values more readable, for example, WeekDays.Monday is more readablethen number 0 when referring to the day in a week.
- An enum is defined using the enum keyword, directly inside a namespace, class, orstructure.
- All the constant names can be declared inside the curly brackets and separated by a comma.

Example

```
enum Weekdays
{
    Monday,
    Tuesday,
    Wednesday,
    Thursday,
    Friday,
    Saturday,
    Sunday
}
```

Enum Values

- If values are not assigned to enum members, then the compiler will assign integer values toeach member starting with zero by default.

Example

```
enum Weekdays
{
    Monday,     // 0
    Tuesday,    // 1
    Wednesday,  // 2
    Thursday,   // 3
    Friday,     // 4
    Saturday,   // 5
    Sunday      // 6
}
```

- You can assign different values to enum member. A change in the default value of an enum member will automatically assign incremental values to the other members sequentially.

Example

```
enum Categories
{
    Electronics,    // 0
    Food,           // 1
    Automotive = 6, // 6
    Arts,           // 7
    BeautyCare,     // 8
    Fashion         // 9
}
```

- You can even assign different values to each member.

Example

```
enum Categories
{
    Electronics = 1,
    Food = 5,
    Automotive = 6,
    Arts = 10,
    BeautyCare = 11,
    Fashion = 15,
    WomanFashion = 15
}
```

- The enum can be of any numeric data type such as byte, sbyte, short, ushort, int, uint, long,or ulong. However, an enum cannot be a string type.
- Specify the type after enum name as: type. The following defines the byte enum.

Example

```
enum Categories: byte
{
    Electronics = 1,
    Food = 5,
    Automotive = 6,
    Arts = 10,
    BeautyCare = 11,
    Fashion = 15
}
```

Methods

Enum.CompareTo

- Compare to method is user to compare two enums.

Enum.Format(Type, Object, String)

- Converts the specified value of a specified enumerated type to its equivalent string representation according to the specified format.

Format Parameters:

- g or G – If value is equal to a named enumerated constant, the name of that constant is returned; otherwise, the decimal equivalent of value is returned.
- X or x - Represents value in hexadecimal format without a leading "0x".
- D or d - Represents value in decimal form.

Enum.GetHashCode

- Returns the hash code for the value of this instance.

Enum.GetName

- Retrieves the name of the constant in the specified enumeration that has the specified value.

Enum.GetNames

- Retrieves an array of the names of the constants in a specified enumeration.

Enum.GetValues

- Retrieves an array of the values of the constants in a specified enumeration.

Enum.IsDefined

- Returns a Boolean telling whether a given integral value, or its name as a string, exists in a specified enumeration.

Enum.Parse

- Converts the string representation of the name or numeric value of one or more enumerated constants to an equivalent enumerated object.

Exception Handling

- Exception handling in C#, supported by the try catch and finally block is a mechanism to detect and handle run-time errors in code.
- The .NET framework provides built-in classes for common exceptions.
- The exceptions are anomalies that occur during the execution of a program. They can be because of user, logic or system errors.
- If a user (programmer) does not provide a mechanism to handle these anomalies, the .NET runtime environment provides a default mechanism, which terminates the program execution.

Try catch finally

- C# provides three keywords try, catch and finally to implement exception handling.
- The try encloses the statements that might throw an exception whereas catch handles an exception if one exists.
- The finally can be used for any clean-up work that needs to be done.

Example

```
Try
{
    // Statement which can cause an exception.
}
catch(Type x)
{
    // Statements for handling the exception
}
finally
{
    //Any cleanup code
}
```

Throwing an Exception

- In C#, it is possible to throw an exception programmatically. The 'throw' keyword is used for this purpose. The general form of throwing an exception is as follows.
- Syntax: **throw** exception_obj;

Project Types

Windows App, Class Library

Windows App:

- Windows Forms is a Graphical User Interface(GUI).
- It is installed in the windows platform using windows operating system.
- Which is used to create or develop easier interface for applications of desktop, tablets etc.
- Windows Forms Application can contain the different type of controls like labels, list boxes etc.
- It can run only on windows platform.

How to Create a Windows Forms Application in visual Studio?

Step 1: Open Visual Studio Go to File and then click on New.
Step 2: Click on the Project. Here we have to choose the language as C#.
Step 3: Now Select on the Windows Forms App and enter the project name and location to save.
Step 4: Click on the Ok Button.

Class Library:

- Class library defines types and the methods that are called by an Applications.
- When we create a class library it can be distribute on NuGet Package or as a component bundled with the application that uses it.

How to Create Class Library Project?

Step: 1 Create one project.
Step 2: Right-Click on the Solution in Solution Explorer and select Add > New Project.
Step 3: Choose C# language and click on the Class library.
Step 4: Write the Project name and location of the project.
Step 5: Click on the next button.

Web Application:

- Web Application is an application that runs on web browser making use of web server.
- Web application is a Client-side and server-side software application.
- We can run an application in browser.
- It can be accessed from anywhere around world using internet.
- Web application is independent of type of system.

How to Create Web Application?

Step 1: Create a new Project.
Step 2: Choose C# Language and choose windows in platform list.
Step 3: Choose ASP.Net Core Web App form list.
Step 4: Now Enter the Name and location of the project.
Step 5: Click on the Next.

STRINGS

- A STRING IS A SERIES OF CHARACTERS THAT IS USED TO REPRESENT TEXT.
- IT CAN BE A CHARACTER, A WORD OR A LONG PASSAGE SURROUNDED WITH THE DOUBLE QUOTES ".C# PROVIDES THE STRING DATA TYPE TO STORE STRING LITERALS.
- THERE **TWO** WAYS TO DECLARE A STRING VARIABLE IN C#. USING **SYSTEM.STRING** CLASS AND USING **STRING** KEYWORD.BOTH ARE THE SAME AND MAKE NO DIFFERENCE.

PROPERTIES

PROPERTY	DESCRIPTION
CHARS[INT32]	GETS THE CHAR OBJECT AT A SPECIFIED POSITION.
LENGTH	GETS THE NUMBER OF CHARACTERS IN THE CURRENT STRING OBJECT.

USE OF VARIOUS STRING METHODS

PROPERTY	DESCRIPTION
COPY(STRING)	CREATES A NEW INSTANCE OF STRING WITH THE SAME VALUE AS A SPECIFIED STRING.
COMPARE()	USED TO COMPARE THE TWO STRING OBJECTS
COMPARETO()	
CONTAINS(STRING)	RETURNS A VALUE INDICATING WHETHER A SPECIFIED SUBSTRING OCCURS WITHIN THIS STRING.
EQUALS()	DETERMINES WHETHER TWO STRING OBJECTS HAVE THE SAME VALUE.
INDEXOF()	RETURN THE POSITION OF SPECIFY CHAR OR STRING. RETURN -1 IF CHAR DOES NOT FOUND.
INSERT(INT32, STRING)	INSERT THE NEW STRING AT THE GIVEN POSITION.
TRIM()	RETURNS A NEW STRING IN WHICH ALL LEADING ANDTRAILING OCCURRENCES OF A SET OF SPECIFIED CHARACTERS FROM THE CURRENT STRING OBJECT ARE REMOVED.
TRIMEND(CHAR[])	REMOVES ALL TRAILING OCCURRENCES OF A SET OFCHARACTERS SPECIFIED IN AN ARRAY FROM THE CURRENT STRING OBJECT.
TRIMSTART(CHAR[])	REMOVES ALL LEADING OCCURRENCES OF A SET OFCHARACTERS SPECIFIED IN AN ARRAY FROM THE

	CURRENT STRING OBJECT.
TOSTRING()	CONVERTS THE VALUE INTO STRING TYPE.
TOUPPER()	CONVERTS THE VALUE INTO UPPERCASE.
TOLOWER()	CONVERTS THE VALUE INTO LOWERCASE.
TOCHARARRAY()	COPIES THE CHARACTERS IN THIS INSTANCE TO A UNICODE CHARACTER ARRAY.
SUBSTRING()	RETRIEVES A SUBSTRING FROM THIS INSTANCE.

SPLIT()	RETURNS A STRING ARRAY THAT CONTAINS THE SUBSTRINGS IN THIS INSTANCE THAT ARE DELIMITED BY ELEMENTS OF A SPECIFIED STRING OR UNICODECHARACTER ARRAY.

DATE TIME CLASS

- C# DATETIME IS A STRUCTURE OF VALUE TYPE LIKE INT, DOUBLE ETC.
- IT IS AVAILABLE IN SYSTEM NAMESPACE AND PRESENT IN MSCORLIB.DLL ASSEMBLY.
- DATETIME HELPS DEVELOPER TO FIND OUT MORE INFORMATION ABOUT DATE AND TIME LIKE GETMONTH, DAY, YEAR, WEEK DAY.
- IT ALSO HELPS TO FIND DATE DIFFERENCE, ADD NUMBER OF DAYS TO A DATE, ETC.
- IT INITIALIZES A NEW INSTANCE OF DATETIME OBJECT.
- AT THE TIME OF OBJECT CREATION WE NEED TO PASS REQUIRED PARAMETERS LIKE YEAR, MONTH, DAY,ETC

EXAMPLE:

- DATETIME DATE1 = NEW DATETIME(2015, 12, 25);

File

- C# includes static File class to perform I/O operation on physical file system.
- The static File class includes various utility method to interact with physical file of any type
- e.g. binary, text etc.
- Use this static File class to perform some quick operation on physical file.
- It is not recommended to use File class for multiple operations on multiple files at the same time due to performance reasons.
- Use FileInfo class in that scenario.

Points to Remember:

1. File is a static class to read\write from physical file with less coding.
2. Static File class provides functionalities such as create, read\write, copy, move, delete and others for physical files.
3. Static Directory class provides functionalities such as create, copy, move, delete etc for physical directories with less coding.
4. FileInfo and DirectoryInfo class provides same functionality as static File and Directory class.

Important Methods of Static File Class

Method	Usage
AppendAllLines	Appends lines to a file, and then closes the file. If the specified file does not exist, this method creates a file, writes the specified lines to the file, and then closes the file.
AppendAllText	Opens a file, appends the specified string to the file, and then closes the file. If the file does not exist, this method creates a file, writes the specified string to the file, then closes the file.
AppendText	Creates a StreamWriter that appends UTF-8 encoded text to an existing file, or to a new file if the specified file does not exist.
Copy	Copies an existing file to a new file. Overwriting a file of the same name is not allowed.
Create	Creates or overwrites a file in the specified path.
CreateText	Creates or opens a file for writing UTF-8 encoded text.
Decrypt	Decrypts a file that was encrypted by the current account using the Encrypt method.
Delete	Deletes the specified file.
Encrypt	Encrypts a file so that only the account used to encrypt the file can decrypt it.
Exists	Determines whether the specified file exists.
GetAccessControl	Gets a FileSecurity object that encapsulates the access control list (ACL) entries for a specified file.
Move	Moves a specified file to a new location, providing the option to specify a new file name.
Open	Opens a FileStream on the specified path with read/write access.
ReadAllBytes	Opens a binary file, reads the contents of the file into a byte array, and then closes the file.
ReadAllLines	Opens a text file, reads all lines of the file, and then closes the file.
ReadAllText	Opens a text file, reads all lines of the file, and then closes the file.
Replace	Replaces the contents of a specified file with the contents of another file, deleting the original file, and creating a backup of the replaced file.
WriteAllBytes	Creates a new file, writes the specified byte array to the file, and then closes the file. If the target file already exists, it is overwritten.
WriteAllLines	Creates a new file, writes a collection of strings to the file, and then closes the file.
WriteAllText	Creates a new file, writes the specified string to the file, and then closes the file. If the target file already exists, it is overwritten.