# Variables

**Instance variable :-**

(maintain multiple copies for each instance of the class created. Initialized only when instance created **N** instances **N** copies **0** instances **0** copies)

**Static :-**

Maintain one value for the whole class can be modified but will not create multiple copies initialized one and only one time

**Const :-**

These variable can’t be modified after declaration must assign the value at the time of declaration can’t modified maintain one and only one copie through out life cycle of the class.

**Readonly :-**

These variables also can’t be modified after the initialization (after declaration maintain **N** instances **N** copies **0** instances **0** copies)

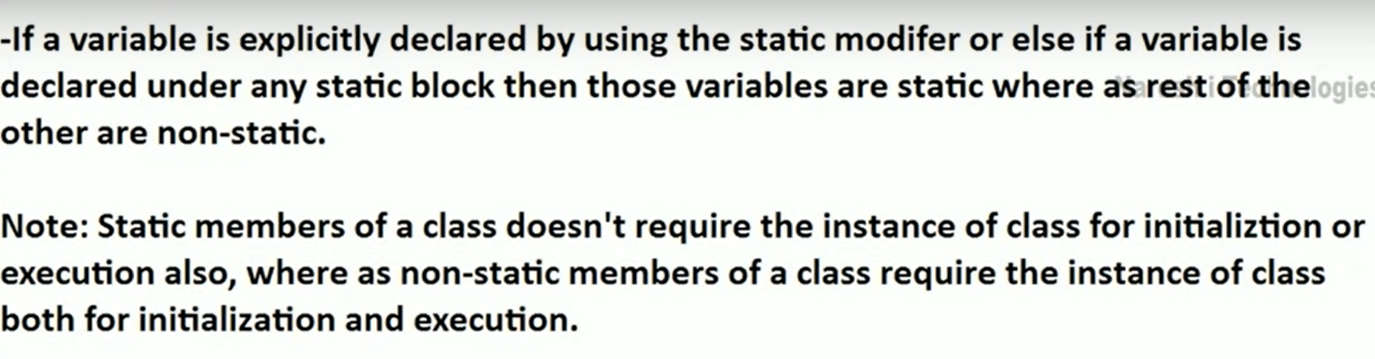
**Modifiable Non-Modifiable:-**

**Static :-** single copie modifiable static

**Constant** single copie non modifiable static

**Instance:**-Multiple copies modifiable

**ReadOnly** :- Multiple copies non modifiable



Static variables of a class are initialized immediately once the execution of class starts where as instance variables are initialized only after creating the class instance as well as each and every time the instance of class is created

If a variable is declared by using the keyword “const” we call it as a constant variable and these constant variable can’t be modified once after their declaration, so it’s must to initialize constant variable at the time of declaration only.

Const float pi=3.14f;

The behaviour of constant variable will be similar to the behaviour of static variable i.e. initialized one and only one time in the life cycle of a class and doesn’t require the instance of class for accessing or initializing.

The only difference between a static and constant variable is static variable can be modified where as constant variable can’t be modified.

If a variable is declared by using the readonly keyword we call that variable as a readonly variable and these variables also can’t be modified like constant but after initialization. It’s not compulsory to initialize a readonly variable at the time of declaration, they can be initialized under the constructor

The behaviour of readonly variable will be similar to the behaviour of non-static variables, i.e initialized only after creating the instance of class and once for each instance of the class created.

The only difference between readonly and instance variable is instance variable can be modified but not readonly variable.

Constant variable is a fixed value for the whole class where as readonly variable is a fixed value specific to an instance of class.

# Inheritance

It’s a mechanism of consuming the members of one class in another class by establishing parent/child relationship between the classes which provides reusability

How to perform the inheritance Consuming the members of A from B

Class A

{

Members

}

Class B:Class A

{

Consuming the members of A from B

}

Consuming the members of A from B

(A=>Parent or Base or Super)

(B=>Child or Derived or Sub)

Note:- Child class can consume members of it’s parent class as if it is the owner of those members(Except private members of parent of parent).

child can Inherit all the (business/properties) of my father but not job (because he get his job based on his qualification)parent qualification and child qualification not same

Copy and paste the code i.e(copy parent class methods into child class it’s not reusability reimplementing the code) Impact the size of the application(let us perform the inheritance)

**Important Points:-**

* Parent classes constructor must be accessible to child class, otherwise inheritance will not be possible

Class1 and class 2 both contains implicit constructor(every class contains implicit constructor if not define explicitly)

If a constructor defined implicitly by default public

If I defined class1 Explicit constructor whenever child class instance created it will call parent class explicit constructor

Private constructor inaccessible due to protection level

* In inheritance child class can access parent classes members but parent classes can never access any member that is purely defined under the child class.
* We can initialize a parent classes variable by using the child class instance to make it as a reference, so that the reference will be consuming the memory of child class instance. But in this case also we can’t call any pure child class members by using the reference.
* Every class that is defined by us or pre-defined in the libraries of the language has a default parent class i.e Object class of system namespace.(Object class is parent class for every class it will inherit implicitly and have four default methods we can create the instance for object class with out inherit any class and access any child class instance)

Object

Class1

Class2

Class3

Supports all classes in the .NET Framework class hierarchy and provides low-level services to derived classes. This is the ultimate base class of all classes in the .NET Framework; it is the root of the type hierarchy.To browse the .NET Framework

* Type of Inheritence

No of parent classes a child class have or the no. of child classes a parent class have.



Single inheritance

Multiple inheritance

If at all a class has 1 immediate parent class to it we call it as single inheritance and if it has more than 1 immediate parent class to it we call it as multiple inheritance

In Csharp we don’t have support for multiple inheritance thru class. What we are provided is only single inheritance thru classes.

* In the first point we learnt when ever child class instance is created. Child class constructor will implicitly call its parent classes constructor but only if the constructor is parameter less where as if the constructor of parent class is parameterized, child class constructor can’t implicitly call it’s parent’s constructor, so to overcome the problem it is the responsibility of the programmer to explicitly call parent classes constructor from child class constructor and pass values to those parameter

Class parentclass{

public parentclass(int i){

console.WriteLine("Print the parent class constructor value"+i);

}

}

Class ChildClass{

public ChildClass():base(a){

}

// we can pass like this also

public ChildClass(a):base(a){

}

Static void Main(){

childClass CC=new childClass(20);

}

}

# Inheritence 3

Our application mainly deals with this entities

ApplicationType Entite

Bank Customers

School Student

Bussinessconcern Employee

Every application there will be a set of entities that are associated with that application so mainly deal with the entities only now.

It’s a living or a non living object which is going to be associated with set off attributes

Entites

Phone(NonLiving entities) color,campixel,company

Customer:- name,address,phonenumber

Entity:- it’s a living or non living object associated with a set of attributes.

Step1:-Identify the entities that are associated with the application we are developing

Example : School application : Student,TeachingStaff, NonTeaching staff

Step2 :- identify the attributes of each and every entity.

**Student**

Id

Name

Address

Phone

Class

Marks

Grade

Fees

**TeachingStaff**

Id

Name

Address

Phone

Designation

Salary

Qualification

Subject

**NON TeachingStaff**

Id

Name

Address

Phone

Designation

Salary

DName

MgrId

**Database** tables are used to represent the entities.

**Procedural** structures (draw back doesn’t support inheritance so need to declare three structure with duplicate properties)are used to represent the entities.

**OOPS** classes for representing the entities.

Step3 : Identify the common attributes of each entity and put them in a hierarchical order.

(just identify the common properties and put into parent class)

**Parent(identifying the common attributes and put them in hierarchical order)**

Public class **Person**{

Int id;

String name,address,phone;}

================ Child 1==================

Public class **student**:**Person**{

Class, Marks, Grade,Fees }

==============================

Public class **Staff**:**person**{

Designation , salary }

============================

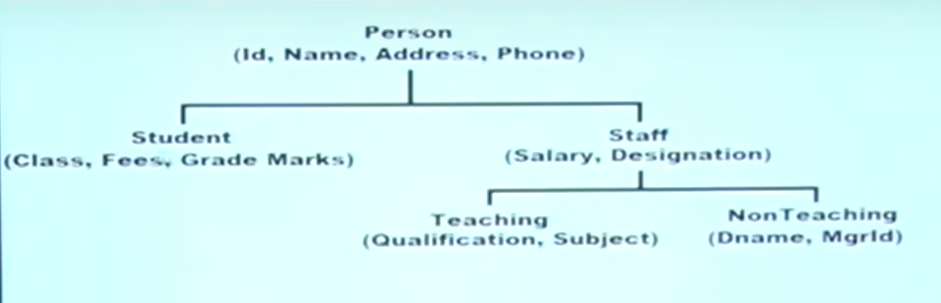
Public class **teaching**:**Staff**{

Qualification,Subject}

===================================

Public class **NonteachingStaff**:**Staff**{

MGRID,Dname}



# Method Hiding:-

