Er. Piyush Pant

Features of C#

① simple → structure approach (break problem into parts)
→ a rich set of library functions data types etc.

② Modern → current trend.
→ powerful & simple for
→ building scalable, interopesable & robust appl^n.

③ object-oriented →

④ Type safe → can only access memory location it has permission to access.

⑤ Interoperability → almost anything native C++ can

⑥ scalable & updatable

⑦ Component oriented

⑧ Structured Programming Language (break program into parts using f^n)

⑨ Rich Library → In build f^n

⑩ Fast speed (compilation & Execution)

Scanned with CamScanner

Er. Piyush Part

. Net Framework

- FCL
  Framework class library
- CLR
  common language runtime.
  ↓
  an appln virtual machines
  ↓
  that provides services such as
  - security
  - memory mgmt
  - exception handling

Scanned with CamScanner

OOP.

→ strategy that provides principles for developing appln.

Problems of Modular Programming.

Reusability
Extensibility
Simplicity → Maintaibility.

Reusability	Extensibility
class and object	Inheritance Aggregation Composition
Abstraction Encapsulation Polymorphism	All of above
Simplicity	Maintaibility

Scanned with CamScanner

OOP Principles.

Encapsulation
Inheritance
Abstraction
Polymorphism

Abstraction

→ process of representing the essential feature without including background details.

→ external world लाई।
class को necessary detail मात्र provide गर्ने।

→ hiding or removing unnecessary thing.

→ abstract class.
→ Inheritance

Encapsulation.

→ The process of binding data and function together into a single unit.

→ Defining class by hiding its internal member from outside class & accessing those internal member only through publicly exposed methods or properties.

→ Data Hiding (Hide internal data from outside world)

Scanned with CamScanner

Uses.

```
Public Register()
{
    private validate()
    Private GetConnection string()
    private saveuser()
}
```

access modifiers

getter
setter

Inheritance

→ process by which members of one class are transferred to another class.

Parent Base class superclass
child Derived class subclass.

Polymorphism

→ poly → many
morph → behaviour / faces.

ability to take more than one form.

→ same f^n / operator will show many behaviours by taking diff type of values or diff no of values.

1. static polymorphism (Early Binding)
2. Dynamic polymorphism (Late ")

Scanned with CamScanner

static polymorphism

- function overloading
- operator overloading

Dynamic Polymorphism

- function overriding

Adv of OOP in C#

a) Modularity

b) code Reusability → Inhesitance.

c) Abstraction → simplifies complex system by hiding unnecessary details.

d) Polymorphism
(obj of diff class through common base class or interface. enables writing code)

e) Encapsulation
↳ restricts access to internal state of objec
→ promoting data integrity & security.

f) scalability (accept future change without modifying previous changes.

g) Readability.

Scanned with CamScanner

Methods

→ also called functions.
→ one used interchangeably
→ define logic ones & use it at many places.
→ makes maintenance of your appln easier.

[attributes] → valid or void.

access-modifier return-types method-name(~~parameters~~ parameters) ← optional

{
method Body
}

Static vs Instance Methods

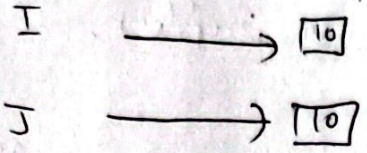
static method → method defn includes static modifier
Instance method → " " doesn't " " "

↳ multiple instance of class can be created.
↳ & each " " have its own seperate method

static method → No instance of method
↳ can invoke only that defn of static method.

Pass by value / reference.

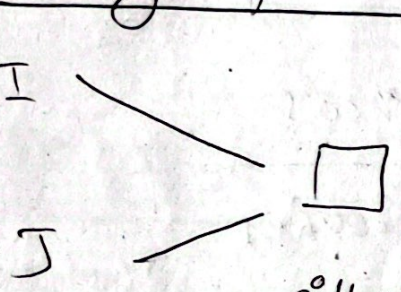
Pass by value.



→ pointing to diff memory location.

→ operation on one variable will not affect value of other.

Pass by reference



→ both I & J pointing to some memory location.

→ operations one variable will affect value of other variable.

pass by reference

```
int i = 0;
simplemethod (ref i);
Console.WriteLine (i);
```

```
public static void
    simplemethod (ref int i)
{
    j = 101;
}
```

```
public static int calculate (int FN, int SN)
{
    return FN+SN ;
}
```

output parameters

```
public static void calculate (int FN, int SN, out int sum,
                              out int product)
{
```

out

```
public static void Main()
{
    int Total = 0;
    int Product = 0;
    calculate(10, 20, out Total, out Product);
    console.writeline("sum = {0}, Product = {1}",
                      sum, product)
}

public static void calculate(int FN, int SN,
                             out int sum, out int Product)
{
    sum = FN + SN;
    product = FN * SN;
}
```

(No return req)

params

ref. type

params मा

```
int[] Numbers = new int[3];
    Numbers[0] = 101;
    Numbers[1] = 102;
    Numbers[2] = 103;

ParamsMethod();
    paramsMethod(1,2,3);
public static void ParamsMethod(param int[] Numbers)
{

}
```

Array optional बनाउन सक्छ।

← गर्न पाउछ।

params use गरिए value type.

(int a, params int[] numbers) ✓

pass हुने value → argument
method decl^n → parameters.

Class
→ consists of data and behaviour
↓ fields ↓ methods

→ complex custom type.

constructor. → invoked when initialization.
doesn't returns value.

Destructor → clean up any resources your class obj are taking

GC ले automatic Destr call गर्छ।

ctor → called automatically when ~~int~~ creating instance of class.

purpose of ctor. → initialize class fields.

mandatory to provide constructor?

→ No.

→ automatically parameterless ctor बनाउछ .Net framework ले।

Scanned with CamScanner

→ once we provide ctor Net framework दे default ctor देता है।

```
class customer
{
  string _firstName;
  string _lastname;
  public customer() : this("No FName", "No LName")
  {
  }

  public customer(string FName, string LName)
  {
    this._firstname = FName;
    this._lastName = LName;
  }
  public void PrintFullName()
  {
    Console.WriteLine("Full Name = {0}", this._firstname
      , this._lastName);
  }
  ~ customer()
  {
    // clean up code
  }
```

```
class Program
{
  public static void Main()
  {
    customer c1 = new customer()
    c1.PrintFullName();
  }
```

Purpose of class ctor.

- → to initialize class fields.
- → automatically called when instance of class is created.
- → not mandatory, if we don't create ctor default ctor is automatically provided.
- → can be overloaded by no & types of parameters.

Static vs Instance.

C1 → -PI = 3.14
-radius = 5

C2 → -PI = 3.14
-Radius = 6

Static
-PI=3.14

Instance

C1 → ~~-PI = 3.14~~
-Radius = 5

C2 → -Radius = 6

Scanned with CamScanner

→ wise to mark -PI as static

members → static keyword हुँदा static member.

~~Obj~~ Instance members → No static.

static members लाई

this keyword use गर्न सकिँदैन।

↳ यो instance को भन्ने बुझाउछ।

Static members	Instance members
→ class members includes static modifier	→ No static modifier
→ invoked using className	→ invoked using instance of object
→ ~~belon~~ only one copy of static member no matter how many instance of class are created. 3 → 1 only	→ every instance of new class have instance members in memory 3 → 3.

class members → fields, methods, prop. events, indexers, ctor.

Scanned with CamScanner

Static ctor.

→ used to initialize static fields in a class.

→ called before instance ctor.

→ 1 time only call.

static circle()

Static prop field भएपा पनि पहिला call हुन्छ।

Inhesitance

```
public class Employee
{
    String FullName;
    String Email;
    String LName;
    public void PrintFullName()
    {
    }
}
```

```
public class FullTimeEmployee
{
    float YearlySalary;
}
```

```
public class PartTimeEmplo
{
    float HourlyRate
}
```

Scanned with CamScanner

→ common code → Base class
→ specific code → Derived class.

Single class inheritance ✓
No multiple " "
But supports " interface "

Child class → specialization of BC

→ Base class are automatically instantiated before derived class.

→ Parent class ctor executes before child class ctor.

base

```
public class Parentclass
{ public Parentclass ()
{ console.WriteLine ("Parentclass ctor called");
}
public Parentclass (string Message)
{
console.WriteLine(Message);
}
public class Childclass : Parent class
{ public Childclass () ~~base~~
{ console.WriteLine ("child called");
}
```

ChildClass cc = new ChildClass();

↳ Default में

parent class called

Child class called

(PL ctor called)

we need to change default & call 2nd ctor.

controlling which ctor to call from desired class.

public ChildClass() : base("DC controlling PC")

{

}

यहाँ पर parent class को message भेजकर ctor call हुआ।

Method Hiding

```
PartTime PTE = new PartTime();
PTE.firstName = "Piyush"
PTE.LastName = "Pant"

((Employee) PTE).printFullName();
```

new keyword से child को use कर parent को hide कर सकते।

```
Employee PTE = new PartTimeEmployee();
PTE.FirstName = "PartTime"
PTE.LastName = "~~LastName~~ Employee";
PTE.PrintFullName();
```

→ parent class reference variable can point to child class obj.

→ but child class ref " cannot " " parent class obj.

→ child can fulfil all of parent.

Concepts

→ use new keyword to hide a base class ~~object~~ member.

→ you will get compiles warning if new keyword is missed.

Hidden base को invoke करें।

→ cast child type to parent type & invoke member.

→ use base keyword.

→ ParentClass PC = new ChildClass()
PC.HiddenMethod()

Scanned with CamScanner

```
public new void PrintFullName()
{ base.PrintFullName();
}
```

(parent को hidden method call गर्छ।)

Polymorphism.

Virtual override use गर्ने।

```
Employee [] employee = new Employee[4];
employee [0] = new Employee();
employee [1] = new PartTimeEmployee();
employee [2] = new FullTimeEmployee();
employee[3] = new TemporaryEmployee();

foreach (Employee e in employees)
{
    e.PrintFullName();
}
```

↳ जहिले पनि parent कै class call गर्छ।

↳ hide भएको class call गर्छ।

Now, child को call गर्न पर्यो भने.

parent मा virtual.

child मा override use गर्ने।

Scanned with CamScanner

→ polymorphism enable us to invoke derived class methods using a base class reference variables at runtime.

→ primary pillar

→ base class मा method declared virtual.
derived " " " " override

→ virtual ~~ne~~ indicates methods can be overriden

base मा virtual
child मा new
overriding > method hiding गर्न सकिन्छ।

Method ~~overloading~~	Vs Method Hiding
BCRV pointing to CCO will invoke overridden method in child class.	BCRV pointing to CCO will invoke hidden method in the base class.
	Derived class D = new Derived class(); D.print(); Derived class को method call गर्छ। Base class B = new Derived class(); B.print(); ↳ Base class को call गर्छ।

Scanned with CamScanner

→ Method overloading allows a class to have multiple methods with the same name but with a diff signature so in C#, functions can be overloaded based on the numbers, type (int, float etc) and kind (value, Ref, out) of parameters.

→ The signature → name of method
type
kind (value, ref, out)
no of formal parameters.

doesn't include → return type & params modifier

Properties.

why properties?

→ marking the class field public and exposing to the external world is bad, as you will not have control over what gets assigned and returned.

Problems with public field.

1. ID should always be non negative no
2. Name cannot be set to Null.
3. If Student Name is missing "No Name" should be returned.
4. PassMark should be read only.

Scanned with CamScanner

```
public class student
{
    private int -id;
    private string -Name;
    private int -passMark = 35;

    public void setName(string Name)
    {
        if (string.IsNullorEmpty(Name))
        {
            throw new Exception ("Name cannot be
                null or empty");
        }
        this.-Name = Name;

    public string GetName()
    {
        return string.IsNullorEmpty(this.-Name) ?
            "No Name" : this.-Name;

    }
```

Scanned with CamScanner

Getter and setter Methods.

→ Programming lang that doesnot have properties use getter and setter methods to encapsulate & protect fields.

→ In this example, we use SetId(int Id) and GetId() methods to encapsulate _id class field.

→ As a result, we have better control on what gets assigned and returned from the _id field.

Note: Encapsulation is one of the primary pillars of oop.

1. get and set accessor to implement prop.
2. get & set both → Read/write "
3. only get accessor → Read only.
4. only set " → write ".

Note: The advantage of properties over traditional Get() or Set() method is that

→ you can access them as if they were public fields.

Scanned with CamScanner

```
public int Id
    { get { return this._Id; }
      set { if (value <= 0)
            {
                throw new Exception("Id is
                    negative");
            }
            this._id = value;
    }
```

c1.Id = 101, → set. accessor invoke हुआ।

Readonly बनाउने ?

```
private passMark;
  public int passMark {
        get { return this.passMark; }
    }
```

→ read only prop.

↳ shortform

```
public string Email { get; set; }
```

↳ compiler ले automatic private field बनाउछ र run गर्छ। public

If No additional logic in property accessors then, we can make use of auto implemented properties introduced in C# 3.0

reduce amount of code to write

when you use auto-implemented properties, the compiler creates a private, anonymous field that can only be accessed through the property's get & set accessors.

struct in C#

Just like classes struct can have

1) Private fields
2) Public properties
3) Constructor
4) Methods

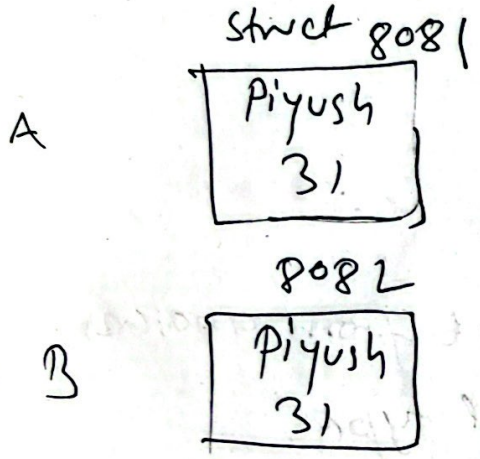
obj initialization same as class.

struct	class
→ value type	→ reference type
→ stored on stack	→ stored on heap.
→ hold value in memory where they are declared.	→ holds a reference to an object in memory
value type	reference variable
→ destroyed immediately after scope is lost.	→ destroyed after scope is lost.
	→ obj is later destroyed by GC.

Scanned with CamScanner

copy one to another
struct → new copy of struct created

modification on second doesn't effect first



value copy हुआ।

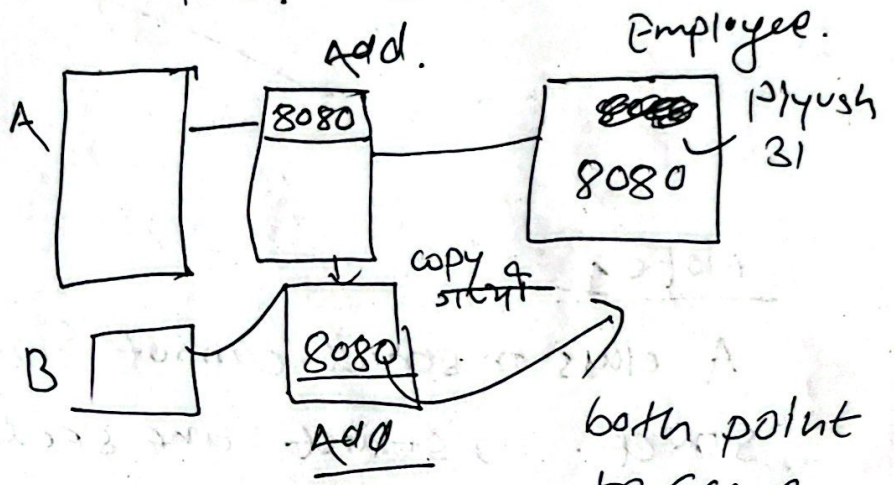
एक का change उसको नहीं डाल

class → another class

→ we only get copy of reference variable.

→ both ref variable (obj) point to same memory location on heap

→ so change on one will affect other.



No destr
→ हुआ

→ can't inherit from another class

→ Both str & class can

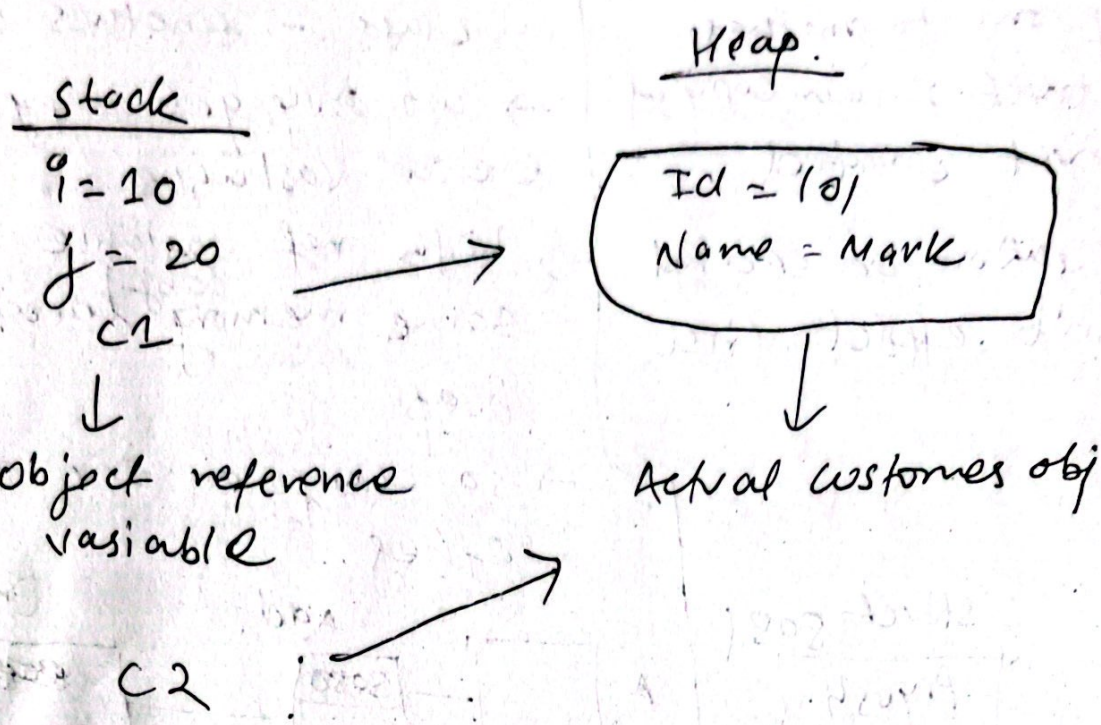
eg: int (System.Int32)
double (System.Double etc).

→ can have destr

→ explicit parameter less ctor हुआ तक।

→ can inherit from interface.

Scanned with CamScanner



Note 1

A class or struct cannot inherit from another struct. → struct are sealed type.

Note 2

How to prevent class from being inherited
↳ use sealed keyword.

Interfaces.

→ we create interfaces using interface keyword.

→ Just like class it also contains properties, methods, delegates or events, but only declaration & no implementation.

→ It is compile time error to provide implementations for any interface member.

→ Interface member are public by default, & they don't allow explicit access modifiers.

→ Interface cannot contain fields.

→ If a class or a struct

Scanned with CamScanner

Abstract Class

→ abstract keyword
→ incomplete → cannot be instantiated
→ can only be used as base class
→ cannot be sealed.
→ may contain abstract members (methods, properties, indexers, events) but not mandatory.
→ Non abstract class
↳ derived from an abstract class must provide implementations for all inherited abstract members.

Class inherits abstract class. 2 options

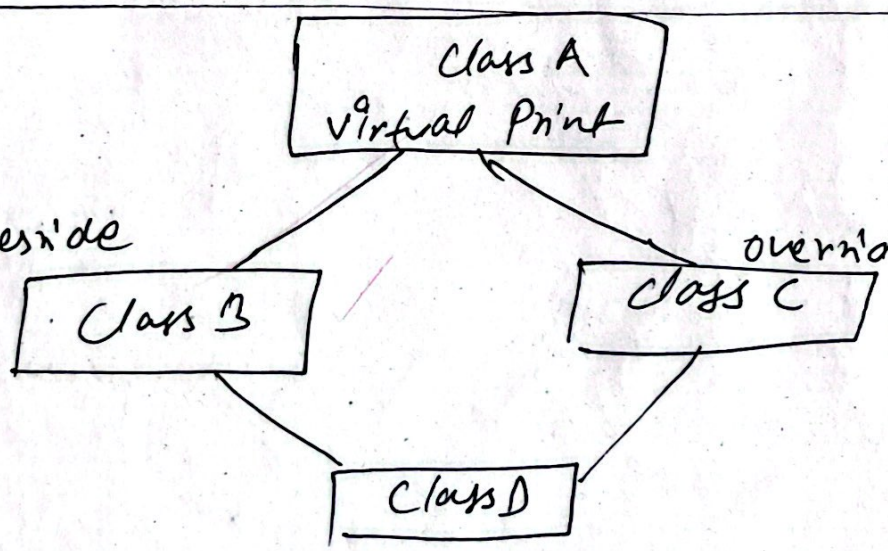
① Provide implementation for all abstract members inherited from base abstract.

② If class doesn't wish to provide implementation for all the abstract members inherited from the abstract class, then the class has to be marked as abstract.

Scanned with CamScanner

Abstract Class	Interface.
→ can have implementation for some of its members	→ can't have implementation for any members.
→ cannot have fields	→ cannot have fields
→ abstract class can inherit from another abstract class or another interface.	→ can inherit from another interface & cannot inherit from abstract class
→ multiple class बाट inherit some time मा हुन्छ।	→ हुन्छ। multiple interface बाट
→ access modifier abs class members can have	→ Interface members cannot have access modifier.

Imp



D: B, C

override गरिन्छ।

Diamond Problem

```
D d = new D();
d.Print();
```

B को call जाने कि C को ?

Exception Handling

Exception → Unforce error that occurs when a program is running.

eg:

Trying to read from file that doesn't exist : FileNot Found Exception.

read from db that doesn't exist : SqlException

Bad (actual error to end uses) due to.

1. users will be annoyed as they are cryptic & doesn't make sense to end user.
2. has info that makes hacking into your app/ne easy.

Exception class that derives from System.Exception class. The System.Exception class has several useful properties, that provides valuable information about the exception.

Message : Gets a message that describes current exception.

Stack Trace : Provides call stack to line no in the method where exception occured.

Scanned with CamScanner

finally block मा memory release गर्ने काम गर्नपर्छ।

streamReader.close();

Releasing System Resource.

try → code that can possibly cause an exception is in try block.

catch → Handles exception.

finally → clean and free resources that the class was holding once during the program execution.

(block is optional.)

Note → good practice to always release resources in the finally block, because finally block is guarenteed to execute, irrespective of there is exception or not.

Specific exception will be caught before the base general exception, so specific exception blocks should always be on top of the base exception block.

Otherwise, you will encounter a compile error.

Scanned with CamScanner

Inner Exception

→ The InnerException property returns the Exception instance that caused the current exception.

→ To retain the original exception pass it as a parameter to the constructor of the current exception.

→ Always check if inner exception is not null before accessing any property of the inner exception object else you may get NullReference Exception

→ To get the type of InnerException use GetType() method.

```
throw new FileNotFoundException( filepath +
    " is not present", ex)
}
catch (Exception ex)
{ Console.WriteLine("current exception = {0}",
    exception.GetType().Name);
}  if (exception.InnerException != null)
{
  Console.WriteLine("Inner exception = {0}",
    exception.InnerException.GetType().
    Name);
```

custom exceptions.

- Inheritance
- exception Handling Basics
- Inner exceptions.

when to create your own custom exception?

If none of already existing dotnet exceptions adequately describes the problem.

eg. 1) I have an asp.net web appln

2) appln should allow users to have only one logged in session.

3) user is already logged in & if he opens another browser window & tries to login again, appln should throw an error stating he is already logged in another browser window.

In Net framework we don't have any exception that adequately describes that problem. so we need custom exception.

Scanned with CamScanner

Why Enums

→ Enums are strongly typed constants.

→ If an program uses a set of integral numbers, consider replacing them with enums.
Otherwise program becomes less Readable & Maintainable.

→ Enums are enumerations.

→ strongly type constraint
explicit cast needed to convert from integral type ⇔ enum type.
(VV)

underline member (value) same हो तो
enum of one time cannot be implicitly assigned to an enum of another type.

→ default type → int

→ " value for first → zero
inc by 1

→ Value types.

→ enum keyword → to create enumerations
(small)

→ Enum class contains static GetValues()
& GetNames() method which can be used to list Enum underlying type values & Names.

Scanned with CamScanner

(change return type)

```
public enum Gender : short
{
    Unknown = 1,   (change st value)
    Male = 2       (change value)
    Female = 3
}
```

→ Enum.GetValues( )

→ Enum.GetNames( )

strongly type

```
Gender gender = 3   ?
```

you cannot assign value directly to enum.

```
Gender gender = (Gender) 3
int num = (int) Gender.Unknown
```

explicit cast चाहिए।

```
short.MaxValue()
```

Types	Types members
eg: customer	field, properties, methods
class, enums, interfaces, delegates	fields, properties, constructors, methods
access modifiers. private, prote	
→ can have 2 internal, public access modifiers.	can have all access modifiers
def: internal	private default

regions
↳ expand or collapse section of code manually.

Access Modifies	Accessability
→ Private (Default for type members)	only within containing class.
→ Public	Any where, no restriction
→ Protected	within containing ~~class~~ types and types derived from containing types.

Scanned with CamScanner

→ Internal (Default for types)	Anywhere within containing assembly
→ Protected Internal	" " " " and from within derived class from any other assembly.

→ compile type error आएगा।

Attributes

→ Attributes allows you to add declarative information to your programs.

→ This information can be queried at runtime using reflection.

→ Predefined attributes provided by .Net

There are few predefined.

→ Also possible to create custom attributes

Obsolete → Marks type & type members outdated.

WebMethod → expose method as XML web service method.

Serializable → indicates class can be serialized

→ It is possible to customize the attribute using parameters.

→ class that inherits from System.Attribute base class.

Reflection

→ Reflection is an ability of inspecting an assemblies metadata at runtime.

→ It is used to find all types in an assembly and/or dynamically invoke methods in an assembly.

Uses of Reflection

1) When you drag & drop a button on a winforms or an asp.net app. The property window uses reflection to show all the prop of button class. So, reflection is extensibly used by IDE or UI designers.

2) Late binding can be achieved by using reflection. You can use reflection to dynamically create an instance of a type, about which we don't have any information at compile time. So, reflection enables you to use code

that is not available at compile time.

3) consider eg where you have 2 alternative implementation of an interface. You want to allow user to pick one or the other using a config file. With reflection, you can simply read the name of the class whose implementation you want to use from the config file, and instantiate an instance of that class.

customer c1 = new customer ();

↳ compile time मा instance बनाउने → easly binding

late binding → runtime मा instance बनाउने,

Generics

→ C# 2.0 में introduced.

→ Generics allows us to design class and methods de-coupled from data types.

→ GC are extensibly used by collection classes available in System.Collections.Generic namespace

→ One way of making AreEqual() method reusable is to use object type parameters.

Since, any type in .NET directly or indirectly inherit from System.object type, AreEqual() method works with any data type but the problem is performance degradation due to boxing & unboxing happening.

→ Also, AreEqual() method is no longer type safe. It is now possible to pass integer for the first parameter and string for second. It doesn't make sense to compare string with integer.

object use करके pass करना।

System.object. (Problems).

1. Are.Equal() method is not type safe.
2. Performance degradation due to boxing & unboxing.

To make AreEqual() method generic.
we specify type using angular braces as

public static bool AreEqual<T>(Tvalue1, Tvalue2)

If code wants to invoke this method
they should specify type

If working with integer

bool Equal = calculator.AreEqual<int>(2,1);

working with string data type

bool Equal = calculator.AreEqual<string>("A","B"

→ this is method generics.

→ class, Interface & Delegates लाई पनि generic बनाउन मिल्छ।

Same logic
diff DT.

→ same logic
diff data type.

Scanned with CamScanner

Convert.ToString Vs ToString

↳ Handles null → throws a NULL reference exception.

```
customer c1 = null;
string str = c1.ToString();
```

Null Reference Exception.

```
string str = Convert.ToString(c1);
```

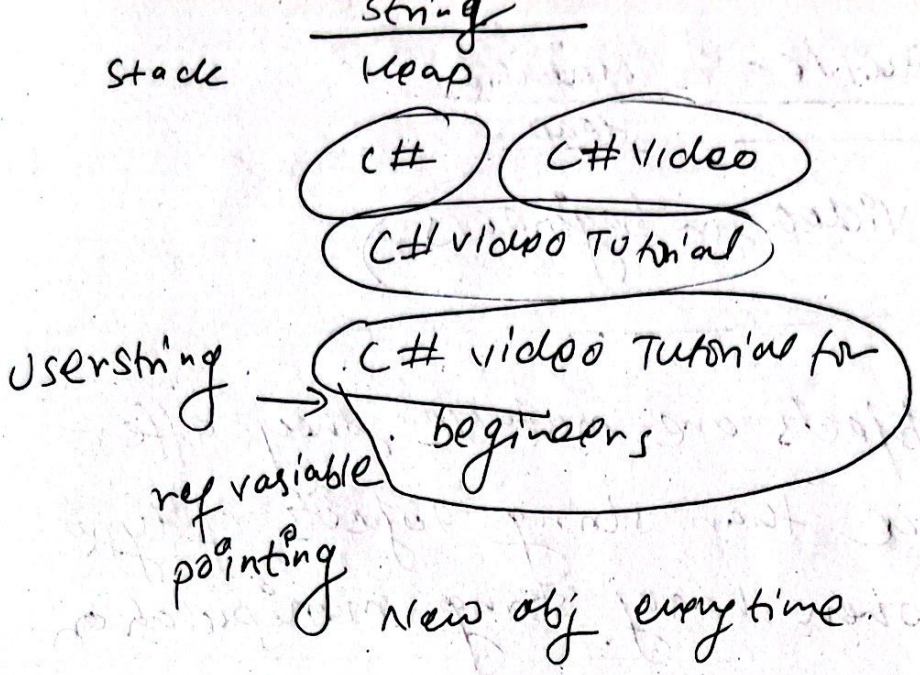
str = " "

Depending on type of appln architecture & what you are trying to achieve. Choose.

System.String Vs System.Text.StringBuilder.

System.String	System.Text.StringBuilder
immutable	→ mutable
	→ Better performance.

```
string userstring = "C#";
userstring += "video";
userstring += "Music";
```



New obj everytime.

5 objects → 4 are not used but remain in memory.

GC run भएपछि clean गर्छ
" " गर्दाखेरि memory मा orphaned string object
बाकी रहन्छ।

StringBuilder is Mutable

Stack.

Heap

C# video for begineers

userstring →

As stringBuilder objects are mutable, they offer better performance than string objects of type system.string, when heavy string manipulation is involved.

```
StringBuilder userstring = new StringBuilder("C#")
userstring.Append("Video");
userstring.Append("Tutorial");
Console.WriteLine(userString.ToString());
```

Scanned with CamScanner

Partial class

→ Partial class allows us to split a class into 2 or more files. All these parts are then combined to single ~~file~~ class when the appl^n is compiled.

→ partial keyword can also be used to split structs and interface.

Adv of Partial class

① Main adv

↳ VS uses pc to seperate automatically generat system code from developer's code.

webforms

↳ a) webform1.aspx.cs → contains dev class

b) webform1.aspx.designer.cs

↳ contains system generated code.

eg: declarations from controllers that you drag & drop to ~~code~~ webform.

② when working on Large projects

↳ splitting class into several files → allow multiple programmers to work on it simultana

Creating Partial classes.

① All the parts spread across different files, must use the partial keyword.

② All the parts spread across different files, must have the same access modifier.

③ If any of the part are declared abstract then the entire type is considered abstract.

④ sealed → some

⑤ inherit a class → then, entire type inherits that class.
If any of parts

⑥ multiple class inheritance ~~interface~~ not supported in C#
↳ diff partial class ले diff class लाई inherit गर्न पाईंदैन।

⑦ interface ✓
Different ~~type~~ parts of partial class can specify different base interfaces, and the final type implements all of the interfaces listed by all the partial declarations.

⑧ Any members that are declared on partial definition are available to all of the other parts of the partial class.

Scanned with CamScanner

Partial Methods.

1. A partial class or a struct can contain partial method
2. A partial method is created using the partial keyword.
3. A partial method decl^n consists of 2 parts
   i) The defn (only method signature)
   ii) The implementation
4. The implementation for the partial method is optional
   → If we don't provide the implementation, the compiler removes the signature and all calls to the method.
5. Partial methods are private by default
   → & it is compile type error to include any access modifiers including partial.
6. It is compile time error to include ~~any~~ decl^n & implementation at the same time for a partial method.
7. A partial method return type must be void. Including any other return type is compile time error.

Scanned with CamScanner

(8) signature of partial method declⁿ, must match the signature of implementation.

9) Partial method must be declared within partial class or partial struct. A non partial class/struct cannot include partial methods.

10) A partial methods can be implemented only once.

Throw vs Throw ex

propogates stack trace	resets stack trace

↓

use गर्ने ? why ?

→ end uses लाई internal stack tra नदेखाउन

→ only show meaningful value lay man can understand.

Var vs Dynamic

Var dynamic typed ×

Static typed → data time known at runtime

code लेख compile. run time मा right को data दिंदै datatype change गर्ने।

```
Var x = 100;
```

Dynamic y = "Hello";

↑

यसमा datatype compile time मा देखाउँदैन,
run time मा check गर्छ।

dynamic typed.

y.length build हुँदा run गर्दा error आउँछ।

checks during runtime.

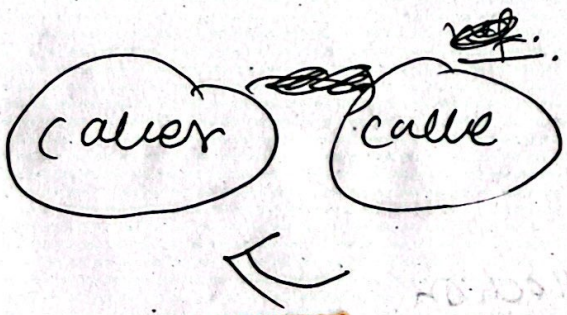
Var is static typed
Dynamic is dynamic ".

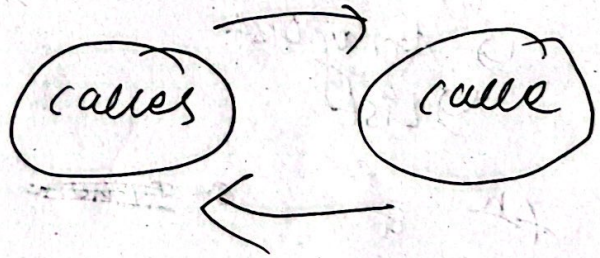
Out vs Ref.

(help to pass data by referen

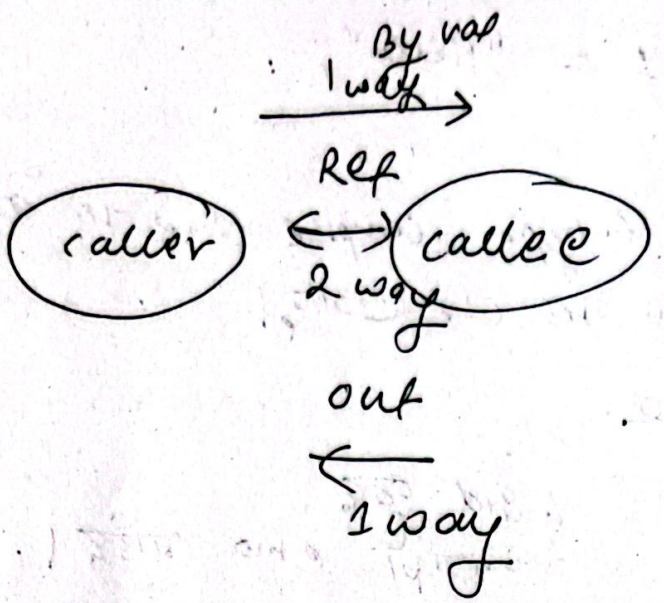
→ ~~out and ref~~

out	ref
pass by ~~value~~ reference.	pass by reference
pass one way	both way
calle to calles	





by pass



1. Out and Ref helps to pass by reference.
2. Ref is 2 way from caller to callee
3. Out is 1 way it sends data back from callee to caller & any data from caller is discarded.

IEnumerable.

→ helps to iterate without knowing the actual type → abstraction.

↳ ArrayList
↳ List

fn &

↳ hides what type of collection.

IEnumerable vs IQuerable

Piyush Paul

→ Both are interfaces to .Net collections.

→ IQueryable inherits from IEnumerable.

" can do all " can do.

There are some things that IQueryable can do which IEnumerable don't do.

Imp.

```
IEnumerable<Employee> ob = dal.Employees;
" < " > d = ob.where(x=>x.Empid==2)
            .ToList<Employee>();
```

SQL Examples → How SQL are fired on SQL server.

IEnumerable	IQueryable
All data fetch & then filter apply.	only required records sent.
→ complete data SQL → client side → apply filter then	→ filter in server side Db side using SQL.
→ In memory data collection. (RDMS) संग सरोकार नहीं	→ Database बार बार पर्ने।

const vs Readonly

const	Readonly
→ compile time constant	→ Runtime constant
absolutely constant data हुन्छ।	PI can change.
	PI = 3.1
	PI = 3.14
	depends on scnario.
→ configuration मा set गरिन्छ।	initialize पछि change नहुने।
→ stored in IL code	→ ctor मा
ILDSM tools.	

Extension Methods.

Helps you to add new methods to existing types without modifying the original code, inheriting or aggregating.

Scanned with CamScanner

COM (Before .Net Framework)

component obj model.

↳ windows app
↳ web app (VB, ASP)

Disadv

Incomplete oop → doesn't support all features of oop.
platform dependent → Windows.

.NET

→ Network Enabled Technology.
• → oop NET → internet.

.NET → through oop we can implement Internet based appln.

Framework

→ collection of many small technologies integrated to develop appln that can be executed anywhere.

Types.

.NET Framework → Windows

.NET → cross platform

Xamarin/Mono → IOS Android.

Scanned with CamScanner

.NET framework Provide

CLR — common lang runtime

responsible for converting MSIL code → Native code

provides runtime env. to execute code.

execution engine
↓
handles running applⁿ
↓
Thread mgmt
garbage collection
type safety
exception handling
etc.

CL — class libraries.
↓
building blocks of pwgram
↓
.NET framework Install
जहाँ machine में install हुआ।
APIs common fⁿ, types (string, dates, num) भी
provide करते।

Microsoft IL code
MSIL
IL
Intermediate lang code

1st compile

Source code → [Language compiles] → IL

2nd compile

MSIL → [CLR] → Native code
(code specific to OS)

JIT → Just in time compiles. (Component of CLR)

MSIL code → Native code लाने काम गर्छ,

Directly understand by OS.

Support

60 +

9 Microsoft

VB.Net

C#.Net

VC#.Net

J#.Net

F#.Net

JScript.Net

Powershell

Iron Python

Iron Ruby.

Technologies supported

ASP.Net (Active sesves Pages)

ADO.Net (Active Data object)

WCF (Windows Comm^n Protocol)

WPF (Windows Pres^n Foundation)

WWF (" workflow ")

AJAX (Asynchronous Javascript & XML)

LINQ (Lang Integrated Query)

Language.

→ acts as mediator betn programmes & system.

→ offers rules & regln.

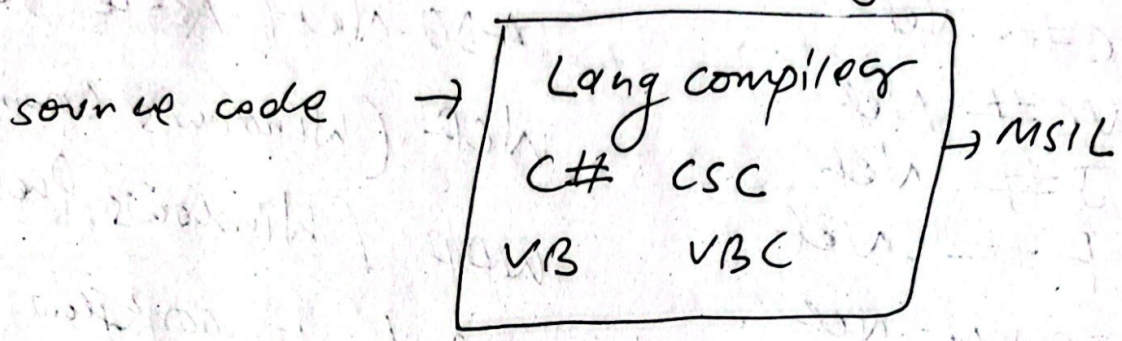
→ " libraries for writing program.

Technology needs.

→ always designed for particular purpose

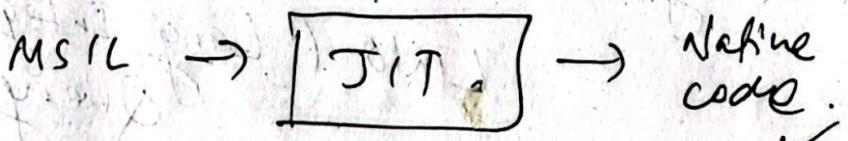
eg: → web related → ASP.Net

→ doesn't offer any specific rules for writing programs. → can't be implemented individually



↓

CLR ले MSIL code लिए JIT लाई दिन्छ।



(OS)

CLR → provides runtime env to execute .Net appln.

IL code → partially compiled code हो cannot be executed by OS.

why we need second compilation?

→ कुन env मा run हुन्छ थाहा छैन code run हुन्छ थाहा छैन।

↳ windows XP
windows 7
" 10
" 11
" server.

we don't know OS that runs our code.

CLR

1. security manager
2. JIT compiler
3. Memory manager
4. Garbage collector
5. Exception Manager
6. common lang specification (CLS)
7. common Type system (CTS)

security Manager.

2 component to manage security

1. CAS (Code Access security)
2. CV (code Verification)

CAS CV

Scanned with CamScanner

→ check privilage of current user
→ " whether " " is allowed to access assembly or not.
→ what kinds of rights / authorities this code has & whether it is safe to be executed by OS or not.

(2) JIT compiles
(3) Memory manager
↳ allocates necessary memory for variable & objects that are used by appln.

(4) Garbage collector
→ when net app runs lots of obj are created. At given point in time it is possible that some of these obj are not used by appln.
GC → is small routine or you can say background process thread that runs periodically & try to identify what obj are not being used currently by appln & deallocates the memory of that obj.

Scanned with CamScanner

5) Exception Manager.

↳ redirects control to execute catch or finally blocks whenever exception occurred at runtime.

→ If you have not handled the runtime excepti then the exception manager will throw the exception & abnormally terminates the program execution at a line where the exception has occured.

6) CTS in .Net Framework

→ Every programming lang has its own data type

→ one " " Data type cannot be understood by another PL.

→ comm^n गर्न पर्यो C# र VB # suppose के गर्ने।

↳ common understandable Data type हुन पर्यो।

Common Type System.

↳ ensures data type defined in 2 PL get compiled to common data type.

CLR → compile time मा all lang specific DT → CLR DT मा convert गर्छ।

(Common to all .Net supported PL)

Common Language Specification

lang
↳ syntax rules → " "
comm^n 2 programming "

then you need CLS

CLS → has own lang specification common to diff programming lang.

C# code → (CLS) compilation → MSIL

VB code

(common lang specification code.)

.DLL	.exe
windows console →	.exe
class library →	.dll

managed code → IL code

CLR manages the code & runs it.

VB6 मा memory deallocation गर्न छुटेमा भने → memory exception आउन सक्छ

C# मा (CLR) भएको → CLR ले deallocate गर्छ।

C++ → both managed / unmanaged code generate करते

ILDASM → Intermediate lang dis Assembler

admin mode.

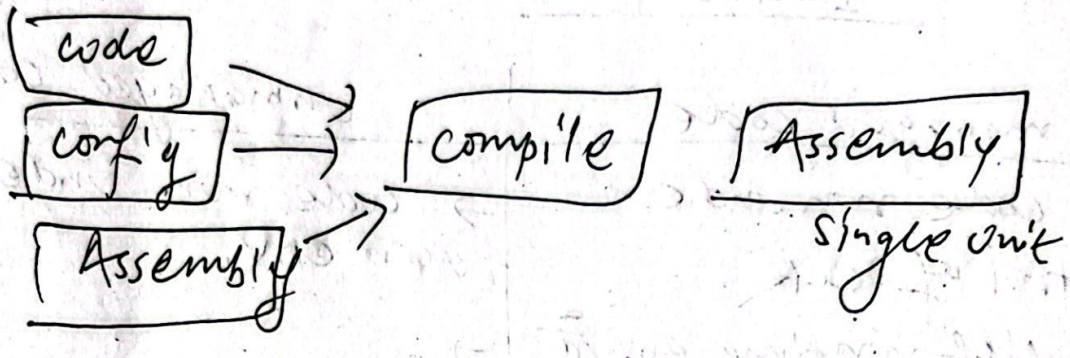
cmd. ildasm.exe.

managed code	unmanaged code
→ code run under control of CLR.	→ code run under their own env.
→ Net runtime env में run करते।	→
All CLR facilities provided (C# code)	Powerpoint, skype, Excel.
→ Lang Interoperability	→ don't require Net runtime env.
→ Automatic memory mgmt	→ can run on their own env.
→ exception handling mechanism	→ doesn't provides facilities of CLR.
→ Code Access security	
→ Garbage Collection	

Scanned with CamScanner

Assembly

→ building block of .Net Framework Appln.
→ fundamental unit of deployment.
→ Pre compiled .NET code that can be run by CLR.
→



DLL	EXE
Class library	console appln
→ can't be run by itself.	→ runs in own address space (memory space)
→ needs a consumer who is going to invoke it.	
→ runs inside other memory space.	

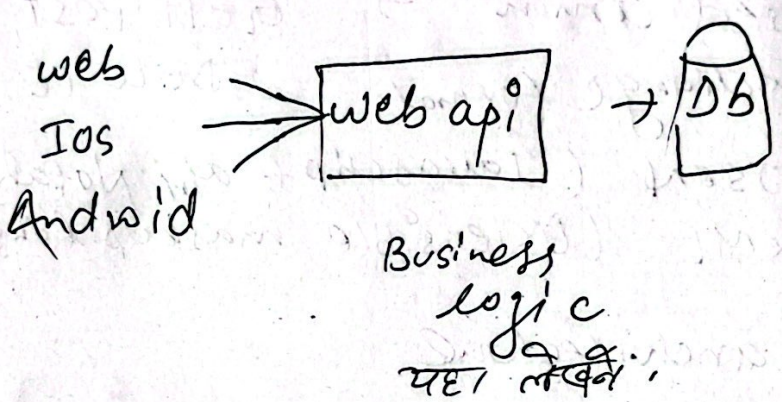
weak Assemblies	strong
→ not signed private/public key pair.	→ signed private/public key pair
→ no guaranteed to be unique	→ guaranteed to be unique & solve DLL Hell problem.

Scanned with CamScanner

Web API.

problem without web API

1. Duplicate logic for each appln
2. Error - Prone code.
3. some frontend framework cannot communicate directly to ~~web~~ database.
4. Hard to Maintain. (Many code) → web, IOS, Android App.



Adv

1. Avoid code duplication
2. Extend Appln functionality
3. Abstraction
4. security

→ Application Program Interface (Not technol.

→ concept not technology

→ that work on HTTP protocol & used to exte functionality of appln.

Scanned with CamScanner

Web APIs

→ set of rules / protocols that allows diff software appln to comm with each other over internet or network.

→ exchange of data & functionalities betn various system using HTTP as commn protocol.

enables

Characteristics	std methods
① HTTP Based commn →	GET, POST, PUT, DELETE

② Data exchange formats

std. → JSON (Javascript obj Notation)
XML (Extensible markup lang)

③ Restful architecture

follow REST principle
↳ stateless.

Rest api is
↳ uses std. HTTP methods.
→ organize resource into hierarchy with unique URLs for each resource.

④ Authentication & Authorization

→ Implement security mechanism for A&A. to ensure only authorized clients can perform specific actions.

common Authentication Methods

API keys, OAuth, JWT.

Scanned with CamScanner

Asp.Net core Web API.

↳ framework for building highly scalable & high performance Restful web services using the Asp.Net core platform.

Tools req
- OS
- .Net SDK
- editor
- Database
- SSMS
- API client → Postman, Fiddler

HTTP

↳ foundation for exchanging info betn web server and client on web.

↳ defines format for message (request & response

Characterstics.

① ↳ Basic function → request/response protocol.

client $\xrightarrow{\text{req}}$ server

HTTP page

Image

$\xleftarrow{\text{response}}$

② Stateless Protocol

↳ retain no memory of past transaction.

↳ cookies/session imp for maintaining session.

Scanned with CamScanner

Methods

to indicate desired actions.

GET (retrieve data)
POST (submit data to be processed)
PUT (update data)
DELETE (remove data)

Secure HTTP (HTTPS)

HTTP in conjunction with SSL/TLS
↳ HTTPS.
→ adds a layer of encryption, enhancing security, which is especially important for sensitive transactions like online banking.

Role in web browsing.

browser —req→ hosting site
← response
(pages)

Header & Response

HTTP message → contains a start-line header.
→ optional body

headers में req/response का info हुए।

Why we need to know HTTP

a) foundation of web comm^n

b) web Development

c) API interaction

d) Troubleshooting

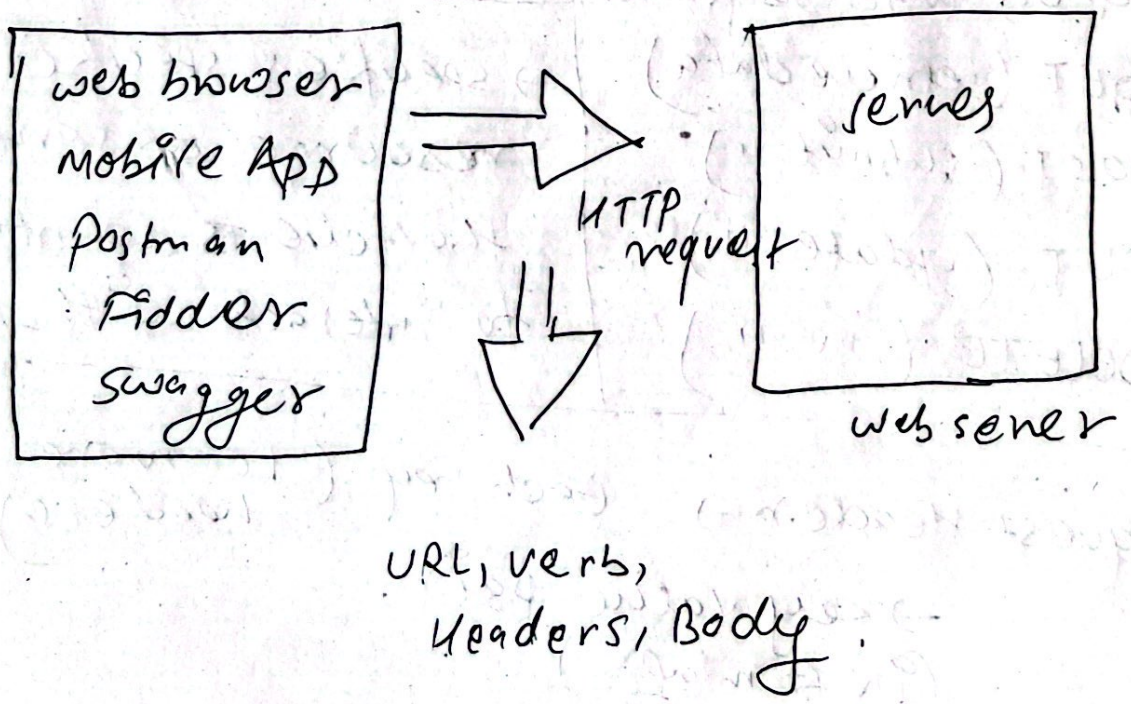
e) security.

f) Performance optimization.
(caching, persistent connections (keepalive)
efficient use of HTTP headers)
→ can significantly improve speed & responsiveness of a website.

g) → SEO Implications.

h) Educational values.

HTTP Request components.



Scanned with CamScanner

① URL (each request must have unique URL)

② verb (method) → " " " " HTTP verb.

GET PATCH
POST
PUT
DELETE

③ Header → each req can contain one or more headers

④ Body → each req can have a body.
The body contain data that we want to send to server.

Request method	Request URI uniform resource identifier
→ specifies action to be performed. GET (retrieve data) POST (submit ") PUT (update ") DELETE (" ")	→ specifies specific resource or server that clients wants to interact with.

Request Header → Each req (1 or more headers)
→ key value pair.
(in form of)

Scanned with CamScanner

common header includes

Host → domain name of server

User Agent → identifies client software initiating the request.

Accept → tells the server what content type client can handle.

content-type → when req include a body (like POST, PUT, Patch req) this header indicates the media type of body.

Authorization → contents credentials for authentication purpose.

Cookie

→ includes any cookies clients have for domain.

↳ used for state mgmt.

cache-control

→ Directives for caching mechanisms in both requests & responses.

Request body (optional)

↳ sending info data to server.

→ POST, PUT, PATCH req.

→ → form submission, file uploads, JSON/XML etc.

contain data for

Scanned with CamScanner

HTTP Response components

whatever client receive from web server
↳ HTTP response.

① Contains
HTTP code
status.

↳ 200 successful
↳ 500 → internal sever error.
↳ 404 → resource not found.

② Response Headers. → 1 or more

③ Data. → returned to client.

Status line.
↳ ① HTTP version
② status code
③ status Text.

Response Headers → 1 or more.
→ Key value pair
in format

content-type. → media type of response body.

content-length → JSON
leng of response in byte. → XML.

cache-control.
Directives for caching mechanisms in both req & response.

Scanned with CamScanner

set-cookie → server to uses agent

server → info about software used by origin server

Date → Date at which msg was sent.

Response Body

↳ actual content of response.

↳ HTML document, image, JSON data.

GET.

→ to retrieve data

↳ only be used to request data.

$\boxed{\text{request}}$ they shouldn't include data)

↳ main purpose → fetch data from server.

(search functionality).

API may expect some data.

POST

↳ make new entry to database.

↳ or new resource in app/n.

PUT

↳ update prop for current resource in db.

PATCH

↳ update few prop of " " " ".

eg: 4 coln

DELETE

↳ delete existing resource from db.

Soft delete. **Is Deleted** **Active**

↳ update जाएगा।

patch use करें

Hard Delete

↳ Delete use करें

HTTP status code

↳ issued from server & give info about ~~server~~ response.

→ when client get any response, we must have 1 status code.

① 1XX (Informational Response)

↳ server received client req & is continuing process.

↳ used for Informational purpose

↳ do not req action from client.

② 2XX Successful Response.

→ req is successful.

→ server successfully received, understood & accepted client request.

Scanned with CamScanner

③ 3XX (Redirectional)

→ some redirectional is happening on server

→ client need further action to complete this req.

④ 4XX (client Error Response)

→ some problem with your request.

→ req not successful due to some error in clientside.

⑤ 5XX (server Error Response)

→ server encountered error while processing request.

Frequent

100 continue : everything OK, client continue or ignore if finished.

200 OK : req processed, returning resource in response body.

→ If you are searching for some data you got data properly.

201 → new resource created.

(POST) ले data insert गर्यो।

URI → of newly created resource is included in response header under

204 → No content.

↳ req succed but is not returning any content.
↳ update or delete ~~HT~~

301 → Moved Permanently

→ resource you are looking is moved permanently to the URL given by location header

302 → Found. (moved temporarily)

→ resource you are looking is moved " to the URL given by location header.

400 Bad Request.

Issue with client req.
→ suppose req contain wrong data.

- → malformed req syntax
- → invalid req message
- → missing req parameters
- → invalid data.

401 Means Unauthorized

username
password

Invalid Authentication

trying to access you don't have access to

403 Forbidden

↳ authenticated but not authorized.
↳ insufficient permission to access the requested resource.

404 Not Found

↳ resource that doesn't exist.

405 Method Not Allowed.

→ server knows request method but that is not supported by target resource.
server ~~m~~ POST method & trying to access using GET method.

500 Internal server Error

↳ Generic error.
↳ error in server, you get 500 internal server error.

503 service Unavailable.

→ server not ready to handle request.
→ maintainance / overloaded
down for

504 Gateway Timeout

→ server while accessing ~~resource~~ gateway/proxy didn't get a response in time from upstream server [illegible]

Scanned with CamScanner

Parameter Binding

↳ how web api binds incoming HTML data (query string or request body) to parameters of action methods of a Web API controller.

complex
primitive

1 or more

Default

	Query string	Req body
Get	primitive, complex	
post	primitive	complex
put	"	"
patch	"	"
Delete	"	NA

complex Type

bool, double - URI (route data / Query string)

[FromUri] Employee employee ←

Default.

FromURI route data
Query string

~~Change~~

FromBody complex Type
Employee.

Change
~~के~~ गर्न
↳ Enforce गर्न
use it
in method.
- like above.

Unit Testing.

→ Developers performs.

→ white box testing → code is visible while testing complete code (SO)

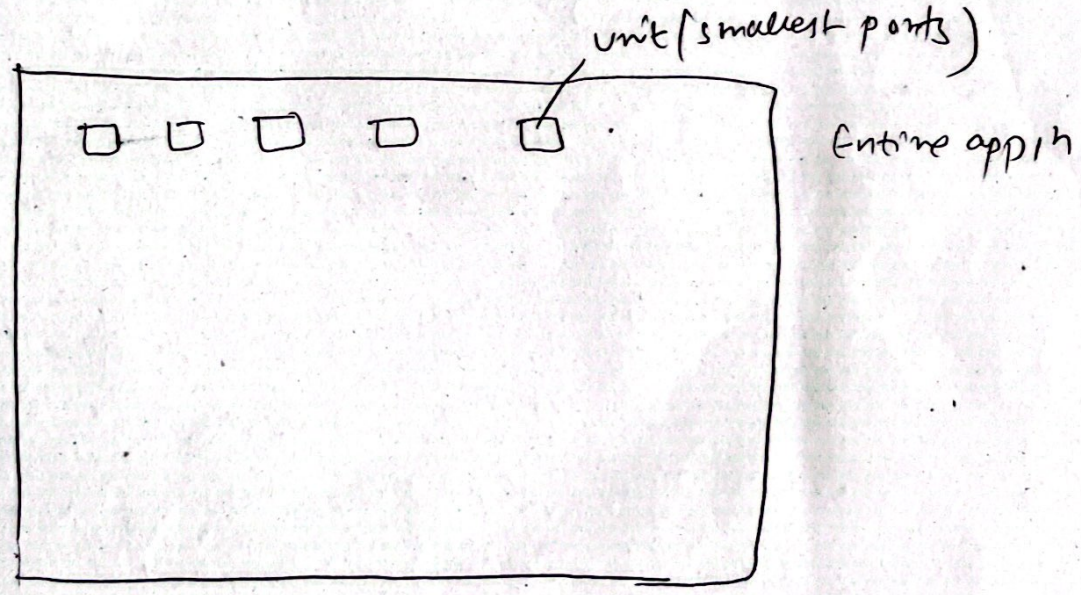
broken

smaller parts which cannot be broken further.

↳ broken down unit.

↳ will be used individually by passing

→ pass valid & invalid input for testing coverage.



calculator Program

↳ small parts.

↳ one of unit → code for addition

another → code " subtraction.

Advantage.

→ cost of fixing defects in Unit Testing is less.

→ Defect can be found early & cost of fixing cheap.

→ Unit testing can be automated and run multiple times as part of Regression testing (selenium)

Unit test can be automated with UI test(

Scanned with CamScanner

Routing → responsible for mapping incoming HTTP request to particular controller action method.

API	MVC.
HTTP method.	URI path

① Conventional Based Routing
② Attribute Routing

Conventional	Attribute
→ use routing template to determine which controller and action to hit.	→
→ at least one route must be added to routing table.	

config.

Scanned with CamScanner

CORS → if app needs to fetch data from diff origin, the SOP would prevent it from doing so. we need CORS

Same origin Policy.

↳ CORS (Cross origin ~~Reques~~ resource sharing)

→ mechanism that allows many resources on a web page to be requested from another domain out side the domain on which resource is created.)

```
EnableCorsAttribute cors = new EnableCorsAttribute
                                ("*", "*", "*");
config.EnableCors(cors);
```

Routing variations

```
[AcceptVerbs("GET", "HEAD")]

[ActionName("Image")]

[NonAction]
```

Scanned with CamScanner

Routing / Attribute Routing.

Routing is used to handle incoming HTTP request based on the URL.

http://localhost:1234/home/index/100

controller action id

Attribute based routing is the ability to manipulate the behaviour of URL by Route attribute.

eg: http://localhost:60995/Home/NewIndex

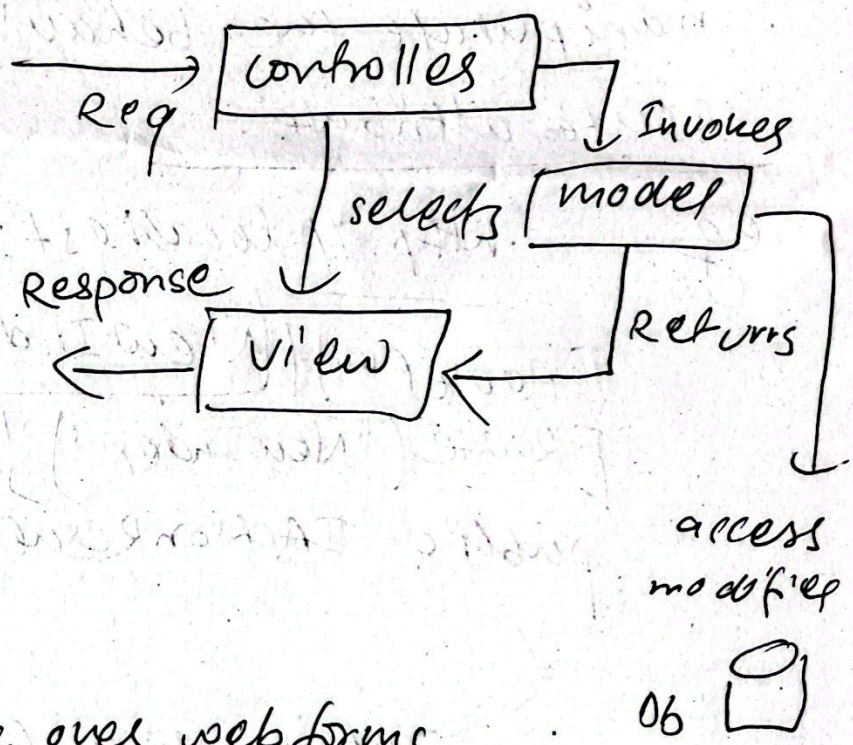
```
[Route("")]
[Route("NewIndex")]
public IActionResult Index()
{
    return view();
}
```

MVC → framework for building web appln.

model → reps data
view → displays data
controller → acts as interface betn model and view components to process all the business logic and then renders the final o/p to view..

Flow

Client side



Adv of MVC over webforms

→ one aspx files → will have 1 aspx.cs file
→ tightly coupled.

→ In MVC one view can interact with multiple controllers & controllers can interact with multiple views.

Scanned with CamScanner

1) seperation of concerns.
UI, BL, model Data.
loosely coupled.

2) Multiple view support.

3). Change Accomdation
New views add गर्दा model मा खासै affect पर्दैन।
can add more seperate view.

4) More control over view.

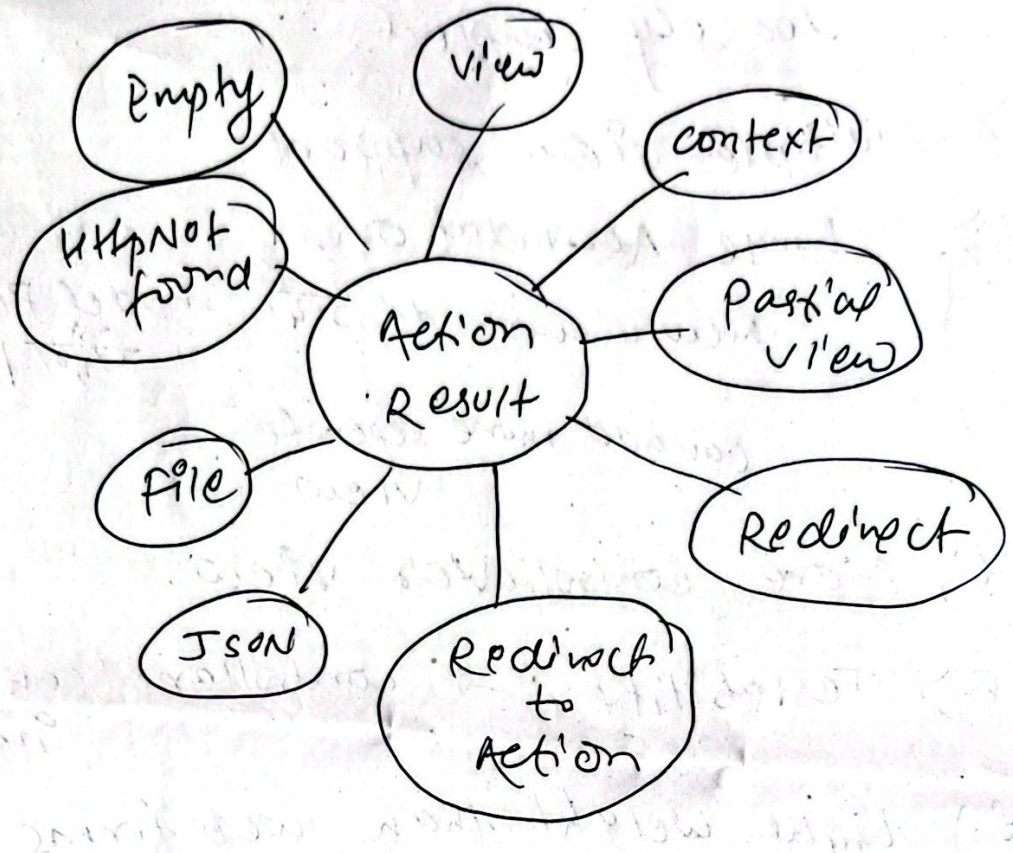
5) Testability → controllers (can test independently)

6) Light weight than web forms
(viewstate use गर्दैन।)

7) Full features of web forms → MVC to.

Scanned with CamScanner

Return Types of controller Action Methods.



ViewResult → we page

✓ PartialViewResult → part of view

Redirect Result → to any other controller action method.

RedirectToRouteResult → any other action.

✓ ContentResult → HTTP content

✓ JSON Result → JSON

JAVASCRIPT Result

File → Binary o/p

EmptyRESULT

Scanned with CamScanner

filter and their types in MVC.

filter → custom class where you can write custom logic to execute before/after an action method executes...

```
[Authorize (Roles = "admin")]
public ActionResult DoAdminstuff()
{
  return view();
}
```

Action filter Types

- **Authorization filters**
  - authentication/auth
  - IAuthorizeFilter
  - [Authorize]
  - [RequireHttps]
- **Action filters**
  - before/after ex of action methods
- **Result filters**
  - IResultFilter
  - [OutputCache]
  - before/after ex of views
- **Exception filters.**
  - IExceptionFilter
  - [HandleError]
  - some oper if unh exce thrown MVC pipeline during As.

Authentication → who are you (username pwd)

Authorization → what are your rights.
(admin
Normal user.)

Authentication → process of obtaining some sort of credentials (usesname, pwd) from the users and using those credentials to verify user's identity.

Authorization → process of allowing an authenticated user access resources.

Authentication always preceds to Authorization.

Types of Authentication.

Form Based Authentication.
→ based on cookies
→ authentication & permission settings are stored in cookies eg: login Page.

Passport Authentication

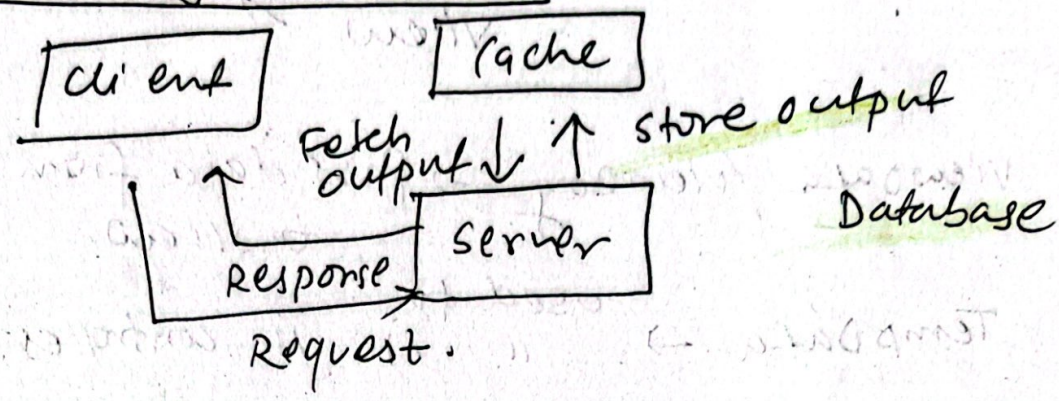
→ centralized authentication service provided by microsoft.

Scanned with CamScanner

Windows Authentication

→ limited no of users.
intranet environment.

Output caching in MVC

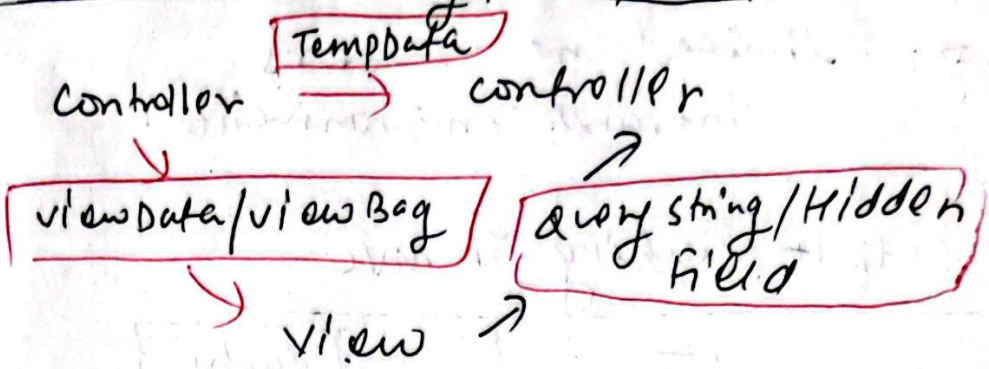


The output cache enables you to cache the content returned by a controller action.
This is kind of Result filter

→ [OutputCache (Duration=10)]

```
public ActionResult Index()
{
    viewBag.Time = DateTime.Now.ToLongTime String ();
}
```

ViewData, ViewBag, TempData



viewData /viewBag → Pass data from controller to view.

TempData → used to " " " controller to ".

viewData → requires type casting for complex data types

look. viewBag → doesn't req type " " "
" ".

Partial views → which renders a portion of view content.

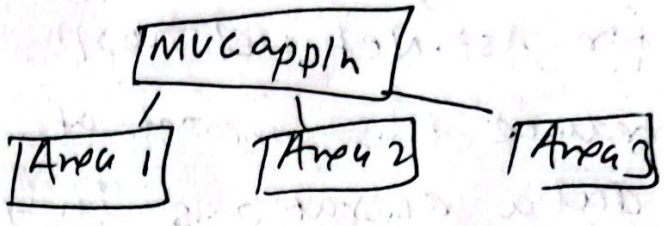
user control of web appln

→ partial can be reusable in multiple views.

→ helps us to reduce code duplication

Scanned with CamScanner

Areas in MVC.



Just a way to divide modules of a large appln in multiple seperated in MVC.

Validation in MVC:

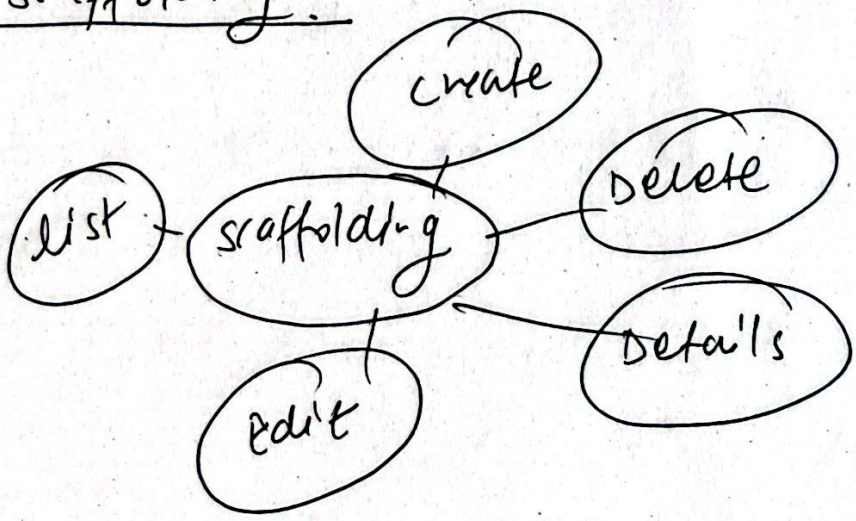
```
[Required]
public string studentName {get;set;}

[Range(10,20]
public int Age.{get;set;}
```

Data Annotation Attributes.

→ validation can be done using DAA in MVC.

MVC Scaffolding.



Scanned with CamScanner

Scaffolding is a code generation framework for Asp.Net web appln. using scafolding can reduce the amount of time to develop std data operations in your project.

Bundling & Minification in MVC.

Bundling → lets us to compile multiple JS files or multiple CSS files. so that they can be downloaded as a unit rather than making individual Http requests.

→ minification

minification → It squeezes out whitespace and performs other types of compression to make the downloaded files as small as possible.

Scanned with CamScanner

V Imp. How to implement ~~ent~~ security in web appl^n.

1. Error Handling must setup custom error page.
2. Cross Site Request Forgery.

   Bank Acc
   email → link. Hacking, Bank Acc.

   avoided by using Anti Forgery Token.

3. Cross site scripting (XSS) attack.

   (JS) → हाकर hack गर्ने। text box मा

4. Malicious File Upload
5. SQL Injection Attack.
   - validate Input
   - use parametrized queries
   - use ORM (Dapper, EF)
   - Use Stored Procedure.

   In line queries are dangerous.

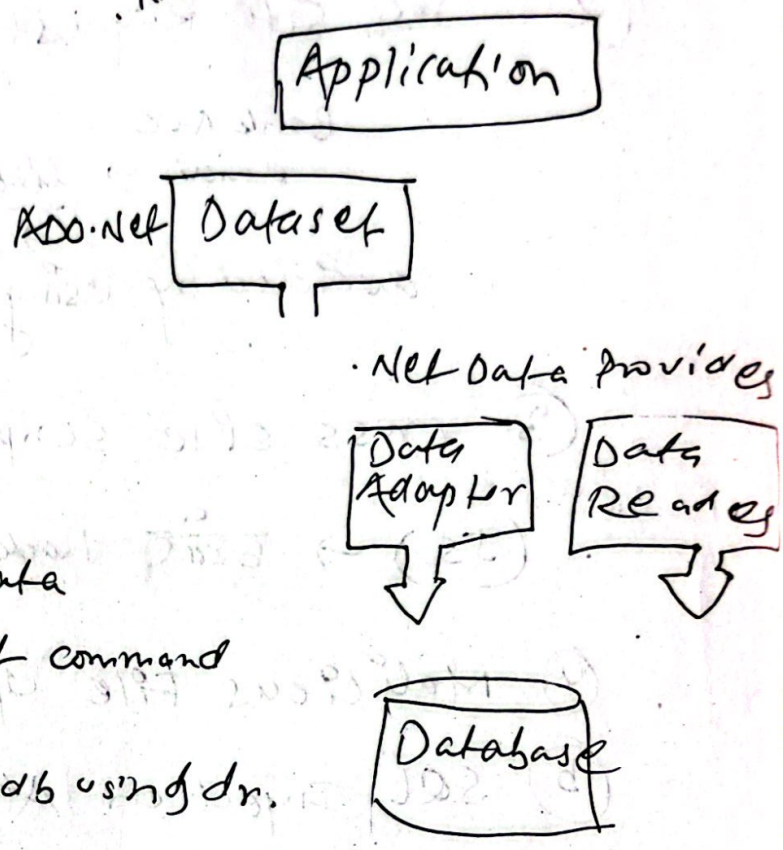
6. Token Based Authentication
7. save pwd in encrypted form in db.
8. Use Https → global level security check.

main component of ADO.Net.

Dataset class.

→ container gets data from 1 or more tables.

disconnect architecture



DataReader class.

allows you to read data returned by select command

read only.

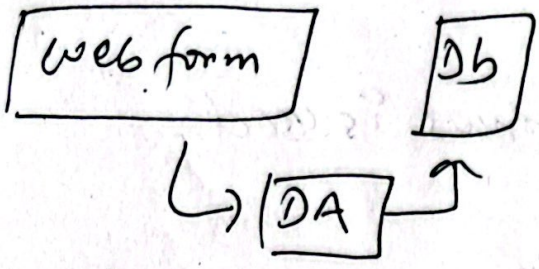
cannot update db using dr.

Data Adapter.

bridge the gap betn disconnect Dataset/DataTable obj & the physical db.

Dataset uses DataReader ↑ DA to load code in

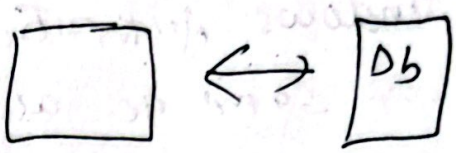
Disconnected



always connect to db पाठिए data to Data Adapter & then result.

Result आएपछि disconnect गर्न सकिन्छ।

connected Architecture



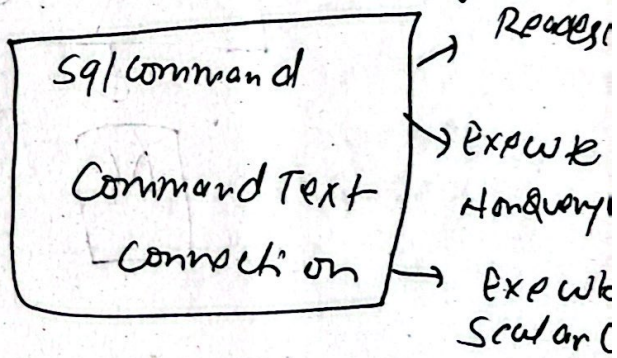
web form

Directly interacting with Db.

Executes methods in ADO.Net.

Execute scalar →

→ Returns single value from dataset.



Execute NonQuery.

Returns a result set from db and it has multiple values.

Insert, update, Delete गर्न use गरिन्छ

executeReader

→ Get Data from Database.

Scanned with CamScanner

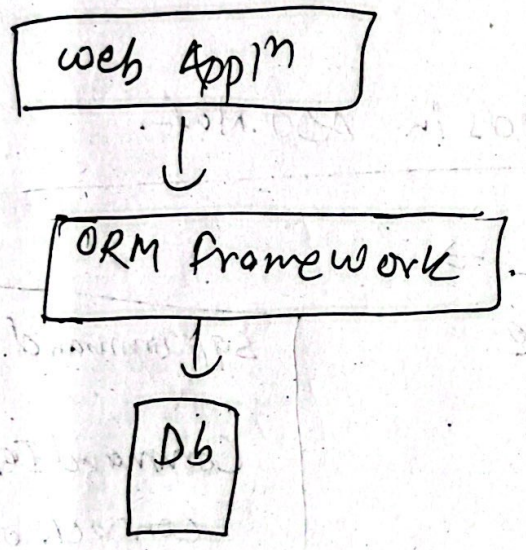
SQL server connect गर्नी technique

Windows Authentication
same ac as your windows is used.

SQL server Authentication
windows + SQL server.

web config मा Db को credentials हाल्न पर्ने।

ORM.



→ ORM is for mapping (obj) in your appln with database object.

→ like wrapper to make db calls simple & easy.

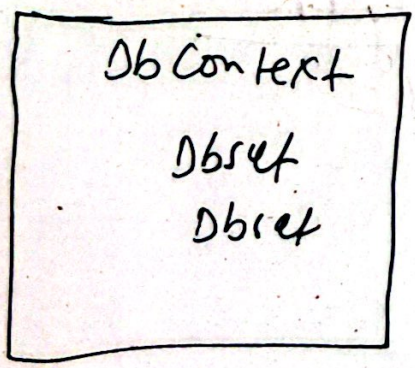
Types.

EF6 → open source orm from micros
EF core → wrapper on ADO.Net oft.
N Hibernate → minimize coding effort
Dapper.

EF

ADO.Net	Entity Framework
write code for DAL, IL & mapping code.	usorcontextDb.Add(user); ;; . savechanges();
	Reduces coding
	Automatic creates code for DAL, IL & mapping codes.
Faster	why. ↳ EF internally used ADO.n ↳ might be doing something extra which is not req.

Dbcontext → class in EF that helps to commn be the db and domain /entity class.



↔ Db

Scanned with CamScanner

Dbset class reps entity sets that can be used for create, read, update & delete operation.

Approaches in Entity Framework.

Db First
→ create entity model from db.

Code First Approach.
→ create model → which creates db.

Model First Approach. (least use.)

model class & [physical db will be generated using this model]
→ relation designed by ORM designer

(LINQ to SQL) & EF.

→ allows you to modify SQL db using LINQ query.

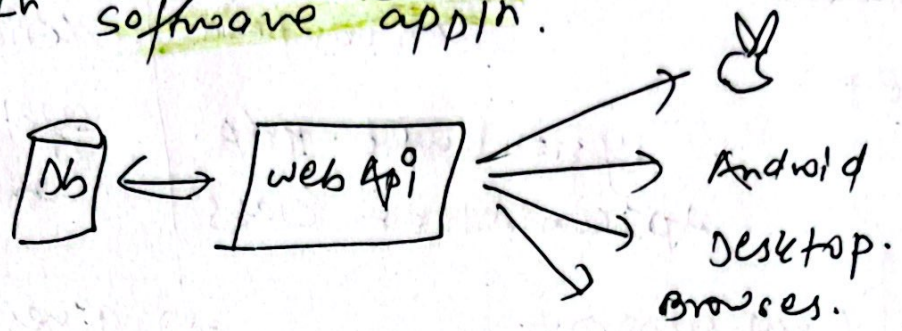
LINQ to SQL → DBML file
EF → .EDMX File

LINQ to SQL	entity FW
old & slow	fast
limit to SQL	

Scanned with CamScanner

web api.

→ Appln programming Interface provides interacti betn software appln.



Adv. over WCF & Web services.

→ every business needs both mobile app & web app.

→ better option.

loosely coupled.

1. → works the way HTTP works.
2. open source.
3. Light Weight Architecture (cell phones लाई राम्रो)
4. Controller pattern similar to MVC controller pattern
5. easy conf settings as comp to WCF.

Scanned with CamScanner

web API vs MVC controller.

~~HTTP~~ method		
GET	derives from system.web.Http.ApiController class	derives from system.web.mvc.Controller class
	→ no view support	→ gives view support

Test web Api

Fidder
Postman
swagger.

Return Types. (web API)

→ void
→ HTTP ResponseMessage → response to Http message.
→ IHttpActionResult
→ (Internally calls ExecuteAsync to create an HttpResponseMessage.)

Other types. (custom)

Scanned with CamScanner

Content Negotiation in web API

client ले चाहेको format मा data return गर्ने।

JSON
XML

Response codes.

1xx Informational
2xx Success
3xx Redirection
4xx Client error
5xx Server error

Basic Authentication

user → username, password → to post req

credentials valid → session will initiate
अस्थायि
to accept the subsequent request
without validating ~~req~~ user again

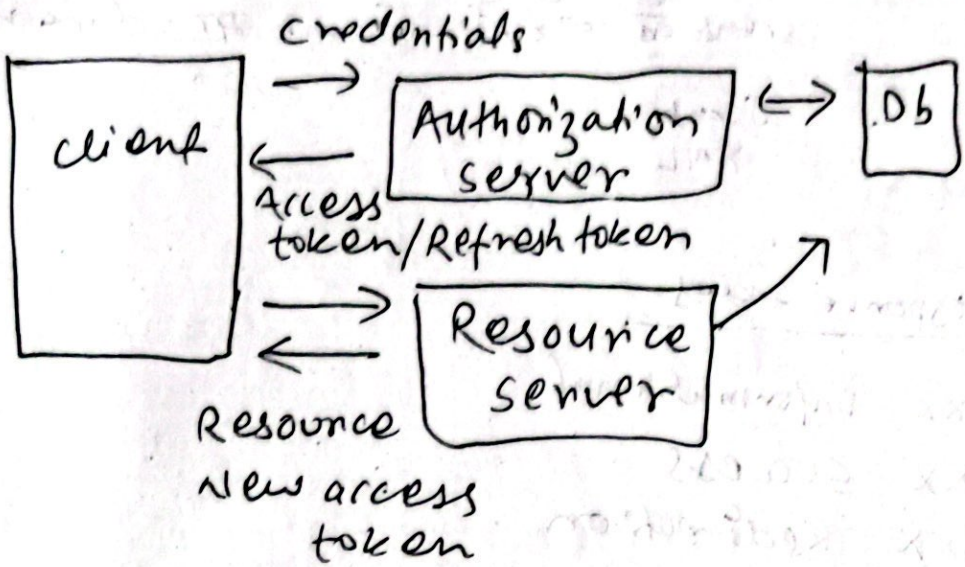
API key authentication.

API owner → share API key with user
& this will authenticate user of that API
key

V. Imp

Token Based Authentication.

Piyush Pandey



→ ① Client sends req to authentication server with a valid credentials.

→ ② Authentication sends access token to client as a response.

→ This token contains enough data to identify a particular user & it has a expiry time.

③ The client appl^n then uses the token to access the restricted resources in the next requests until the token is valid.

④ If the access token is expired, then the client appl^n can request for a new token by using Refresh token.

Scanned with CamScanner

OAuth.

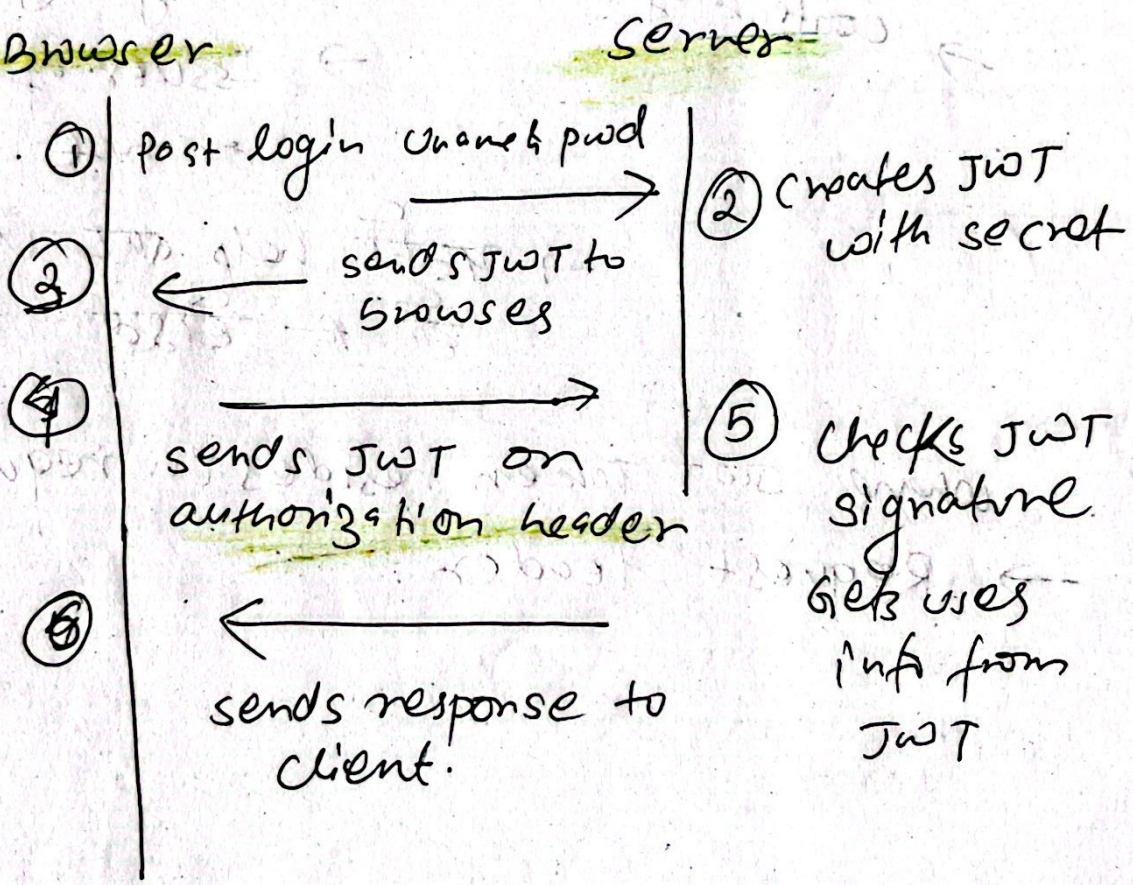
→ Standard for creating token based authentication and authorization

→ set of rules → TBA बनाउन.

JWT follow OAuth std.

VImp JWT Authentication.

→ token based ,, where JWT is token format.



Parts of JWT Token.

Encoded JWT Token → Decoded.

a) Header → ① Type of token ② Algorithm used to generate token

b) Payload ← Contains claims → registered claims → content of token.

c) Verify signature → Issuer of JWT is correct or not.

→ पक्की help करे valid token है कि होकर identify करे

Where JWT Token resides in requests?

→ Request Header.

Scanned with CamScanner

HF256.

DBMS	RDBMS
→ stores data as file	→ stored data in tabular form
→ No relⁿ betn data	→ Data stored in form of tables. related to each other.
3) Normalization not present	3) Present
4) Deals with small quantity of data	4) Large
5) XML	5) SQL series, oracle. etc

Imp. Constraint in SQL.

→ used to specify rules for the data in a table

→ Constraint are used to limit the type of data that can go into table.

→ Primary key → uniquely identify each row in table

→ NOT NULL constraint.

→ FK " → field which can uniquely identify each row in another table.

→ Check " → validate values of colⁿ to meet a particular condⁿ.

Scanned with CamScanner

(5) Default constraint → Default value if no value is given.

(6) Unique constraint

pk	uk
→ No Null values	1 Null value allowed.
→ automatically cluster Index बनाता है।	→ You have to create non cluster index if you need.
→ only 1 pk in table	→ many uk in table.

Views

virtual table → contains subset of data contain in 1 or more tables.

query → stored
but data is not stored in views

(1) Why views.
Indexed views to improve performance

(2) Extra security
→ DBA can hide actual table & expose only views.

Subquery | Nested | Inner Query

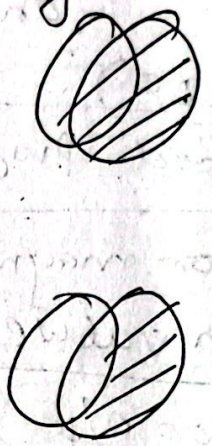
→ is a query within another SQL query & embedded within the where clause.

Types of Joins SQL server.

Left outer Join

where B.key is Null

Right outer Join



where A.key is Null

Full

where a.key is Null
OR b.key is Null

Inner Join

Self Join → Join of table to itself.

Scanned with CamScanner

Emp Id	Mgr Id	Name
1	Null	Piyush
2	1	Parth
3	1	swapnil

→ select E.Name from Employee E
left Join Employee M
on E.Id = M.Id

VVI

Stored Proc	Function
→ may or maynot return value	→ always return values.
→ I/p or o/p parameters	→ only i/p parameters.
→ call fn inside sp.	→ cannot call sp inside fn.
→ cannot use sp in SQL statement	can use in inline queries.
select *, fn from city;	
→ sp मा transactions use गर्न सकिन्छ	→ No No trans.
→ try catch use गर्न सकिन्छ।	No.

Indexes.

→ Indexes are used in relational db to retrieve data very fast.

→ Improve performance of query also.

similar to index at start of books.
↳ purpose → find topic quickly.

Index

3
6
12

Table T1

12	- -	
3	- -	
27	- -	
12	- -	

Clustered Index.

→ defines the order in which data is physically stored in table.

→ only one clustered Index in table.

→ eg: Dictionary

→ PK or col^n → automatically creates cluster index in SQL server only

→ ⊠

Non Clustered Index.

NCI stored at one place & table data is stored at other place.

→ this index is not physically stored.

→ similar to book index.

Adv of OOPS.

1. Reuse of code → inheritance
2. Flexibility → through polymorphism
3. principle of data hiding / Encapsulation helps the programmer to build secure programs.
4. oop system can be easily upgraded to large system.
5. Modularity for easily troubleshooting (Divide करो)

Limitation of OOPS.

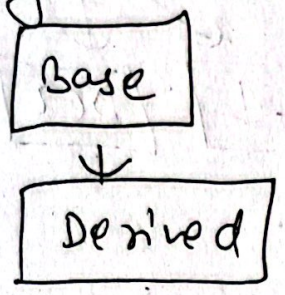
→ Not suitable for small appln.

→ " " Enterprize ".

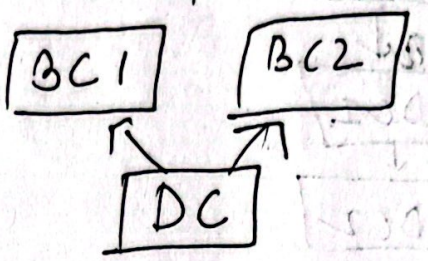
√Imp

Diff Types of Inheritance.

Single



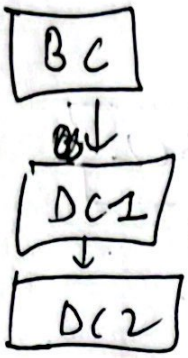
Multiple



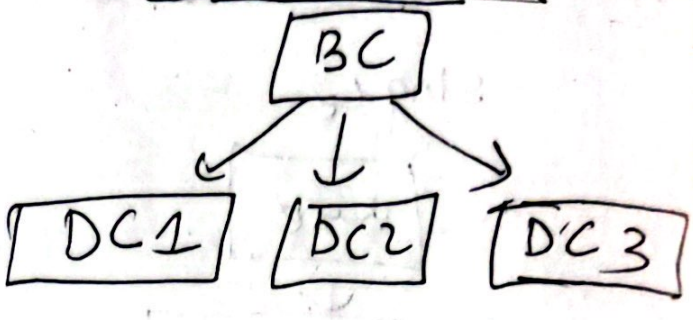

```
class BC1
{ public void Animal()
  {
    console.writeline("Animal");
  }
}
class Interface1
{ public void Fly();
}

class childclass : BC1, Interface1
{ public void Eagle()
  {
    console.writeline("Eagle");
  }
}
```

Multilevel



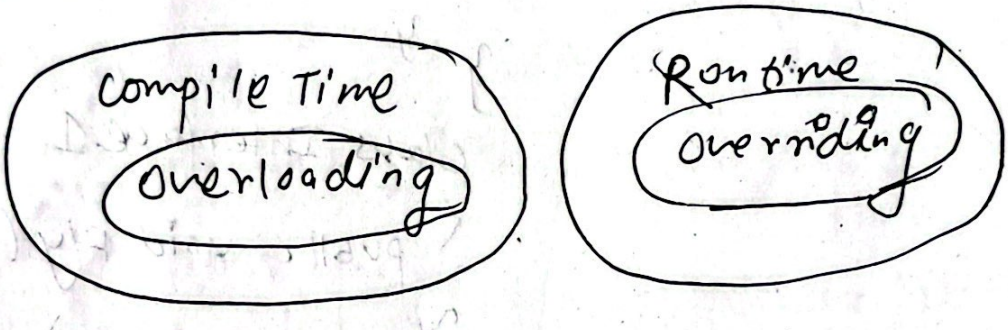
Hierarchical (most use



→ prevent class from inherited → sealed keyword.

Polymorphism

→ ability of variable, obj or fn to take multiple roles.



Q) method overloading? How many ways can be inherite

→ compile time polymorphism

→ in which we can create multiple methods of same name in class & all method work in diff ways.

Compile time → .Net compiler knows these 2 methods are diff at Compile time.

Scanned with CamScanner

→ obj ref not set to an obj
↳ run time error → compiler not able to catch that before running appln

3 ways.

→ same method name diff
a) No of parameters
b) Type " "
c) order of " are diff.

overloading	vs	overriding
No inheritance req. within same class. possible		creating method in derived class same name & signature as a method in base class.
		Base class obj = new subclass(); obj.Greetings();
		Virtual (BC) override (DC) → child की call गई।
		→ Virtual र override राखेर भने BC method use गर्ने।

overloading	overriding
Compile time polymorphism	run time polymorphism
possible in same class	req BC & DC.

V. Imp.

Abstract Class	Interface
→ Contains both Declaration & defn of method.	→ Interface contain only declaration of methods. → no defn or body.
public abstract class car { public abstract void Engine (); (method declared) public void Dashboard() → (method defined) { console.writeline("Dashboard"); } }	Interface car { void Engine () void Dashboard (); } (only method decl^n)
Keyword : Abstract	Interface
→ can contain methods, fields constants, constructor, static members, methods	→ only undefined methods. (No field, No const)

Scanned with CamScanner

→ No Multiple Inheritance	Yes.
Q) when to use what?	
when you are sure some methods are concrete/defined & must be implemented in the same way in all derived class.	you know method have to be there but it can be implemented differently by independent derived class.

public abstract class USATaxSystem { public abstract int calculateTax();
public string TaxCurrency()
{ return USD;
}
calculateTax → Just Declare because it is diff for individual or company.
TaxCurrency → always USD so concrete
DC will use already defined method.
	~~public~~ interface WorldTaxSystem
	{
	public abstract int calculateTax();
	public string TaxCurrency();
	Diff country → diff currency
	so, we need to implement it in DC.
	→ we prefer
	→ Gives us flexibility to modify things.

Why to create interfaces?

World

India Eng USA

new

China Brazil

add.

by using interface new → obj will know methods.

→ maintain consistency in DC of similar types.

Q) Do interface have ctor?

→ No.

→ obj creation हुआ ctor हुआ

Q) Can you create instance of Abs class or Interface

→ No.

→ No obj creation.

out/ref.

By using out/ref → we can pass parameter by ref.

By value. → copy हुए value.

When to use out/ref.

→ Normally return only 1 value if we want to return multiple values.

Scanned with CamScanner

out	ref
→ No need of initialization before passing it	Need to initialize before passing.
→ Need to initialize before returning	→ Not necessary.
→ when you want to get new value from called method (1 way)	→ when you want to modify value caller → (value) → callee ← (returned value) 2 way

params.

→ used as parameters which can take variable no of arguments.

→ useful when program don't have prior knowledge how many params are used.

Access Modifier.

keyword to specify accessibility of class, method, prop, fields.

public → Anywhere
private → Same class
protected → ~~class~~ DC
Internal → Assembly.
ProtectedInternal → DC & Assembly

Scanned with CamScanner

class III default → P

exception Handling

→ used to manage errors.

try { → error can occur here.
}
catch { → if error is passed here.
}
finally {
} → used to execute given set of statement whether exception shown or not.

throw

↳ pass stack trace to 1 level up.

can we write multiple catch block.

↳ yes but only 1 executes.

↳ finally block is executed irrespective of result.
connection close राखन सकिन्छ।

only try block? → yes दुई सक्छ।
गरिँदैन।

finally भित्र con.close();
↳ यहाँ राख्ने

· Unit Testing.

Unit Test → single class or small unit of functionally related class.

→ no uses (UI) click to db. (all)

→ quickest in execution

Integration → potentially less in depth than unit test

→ cover greater range of system

→ test appln with ext dependencies.

→ take longer to execute. because of ext. dep. write/read operation.

User Interface Test → great range of appln.

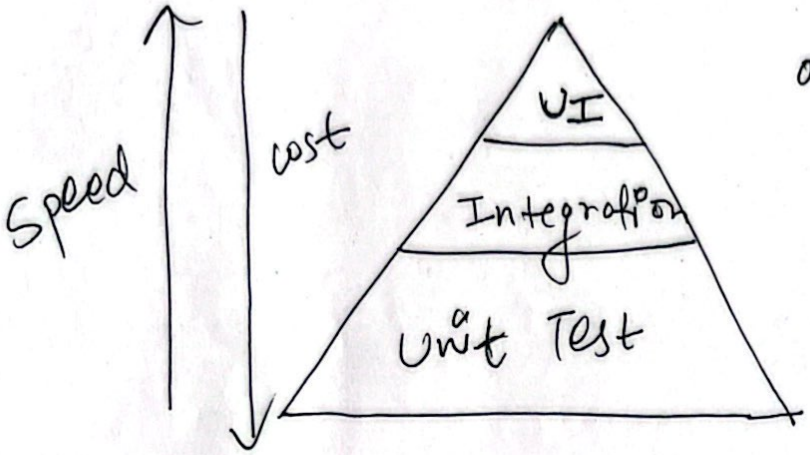
→ uses click button to subsystem to db level.

→ covers wide range but not depth.

→ cannot cover all functionality

→ slowest and brutal. (small change on appln can break this.).

Testing Pyramid.



only test happy path
not all errors - test

most of appln (unit test)
→ (पो unit test ही) हुई।

AAA.

* Arrange → Initial setup. → Create obj instance.
  → Create test data
* Act → execute appln code
* Assest → call methods.
  → check result
  → Test Pass/Fail.

MS Test.

[Test class]
class

[Test Method]
→ fn ही

Arrange. calculator calc = new();
Act. int result = calc.AddNumbers(10,20);
Assest. Assert.AreEqual(10, result);

Scanned with CamScanner

Nunit.

attribute is same.

[TestFixture]

same as MsTest. (Assest).

[Test]

Assest.AreEqual(30, result).

Assest.IsFalse(isodd)

Assest.IsTrue(isodd)

Assert.That(isodd, isEqualTo(false));

Assertion Model.

classic model.

↳ uses a seperate method to express each individual assestion of which it is capable.

Assest.AreEqual(30, result);

Assest.IsTrue(result);

Constraint Model. (Assest.That)

→ single ~~mode~~ method of assert class for all assertions.

AssertThat(30, IsEqualTo(result));

Assert.That(result, Is.False);

Scanned with CamScanner

Assest.AreEqual (customer.GreetMessage, "Hello, Ben stokes");

Assest.That (fullName, IsEqualTo ("Hello ");

Assest. That (fullName, Does.contain(","));

Assert.That (fullName, Does. startWith("Hello");

Assest. That (fullName, Does.endsWith("spark"));

" . " (" ; Does.Match("Hello, [A-Z][1]
[a-z] + [A-Z][1] [a-z]+

Range.

↳ list मधे compare करतो,

AssestThat(is.EquivalentTo(exp oddRange));

Assert.AreEqual(expectedRange, result);

↳ Others

Does.contain(7);

Is.Not.Empty EqualTo

Is. ~~Equal To~~ (3);

Has.No.Member (6)

Is.Ordered

Is.Unique

Scanned with CamScanner

Multiple Assert.

If there are multiple assert statement and any one fails, then it doesn't continue to test other statement

→ If we want all of test to complete & get result

```
Assert.Multiple(() =>
{
    Assert. - -

    - -

});

Assert.IsNotNull(customer.
```

```
var exceptionDetails = Assert.Throws<ArgumentEx
    ception>(() => customer.GreetAnd
CombineNames("", "spark");
```

```
Assert.That("Empty First Name", exceptionDetails.
                                Message);

Assert.That(() => customer.GreetAnd combine
Names("", "spark"), Throws.ArgumentException.
    with.Message.EqualTo("Empty First Name"));
```

Scanned with CamScanner

Unit Test.

→ test that exercises individual software compone. or methods, → "Unit of work"

→ should only test code within developer's contr

→ don't test infrastructure concerns.

Interacting with database
file systems
Network resources

Integration Test.

→ exercise 2 or more software component

→ ability to f^n together → Known as integration

→ operate on boarder spectrum of the system under test.

→ Unit test focus on individual component.

Load Tests.

→ Aims to determine whether or not a system can handle a specific load.

eg: No of concurrent users using an appl^n & the app's ability to handle interactions responsibly.

Scanned with CamScanner

throw ex vs throw.

- DivideByZero

$\frac{0}{0}$

↳ error stack trace.

throw gives more detail → better for developer.
preserves whole stack race.

↳

Loop types for C#

while	Do while	for Loop.
Initialize condn Increment	do{ }while(condn)	all initialization condn inc on 1 line

statement must execute at least one time

foreach

→ iterate collections.

continue	break.
↳ skip remaing statement inside loop ↳ transfer execution to loop.	→ not execute remaining part & transfer execution out of loop. → exit loop.

Scanned with CamScanner

Array	ArrayList
cat[] catArray;	ArrayList catAList;
CatArray = new cat[10];	catAList = new ArrayList();
catArray[0] = moggy1;	catAList.add(moggy1);
catArray[1] = moggy2;	catAList.add(moggy2);
1) → Array is strongly typed. → Array can store only specific type of items/elements. → eg: CAT.	→ ArrayList can store ANY type of items/elements.
2) Array can contain fixed no of items. Defn मा count of item राख्न पर्छ।	2) Any no of items.

ArrayList vs Hashtable

```
ArrayList arrList = new ArrayList();
arrList.Add("seven");
arrList.Add(789);
```

only value

value with key

key value

```
Hashtable hashtbl = new Hashtable();
hashtbl.Add("Number", 1);
hashtbl.Add("cat", "ferrari");
```

Scanned with CamScanner

this keyword

↳ to refer the current instance of class.

```
class student {
    public int Id;
    public string name;

    public student (int id, string name) {
        this.id = id;
        this.name = name;
    }
```

this.id → यो class को fields
id → value received from parameters.

(confusion हटाउन)

VImp: using को purpose

1. using directive
2. using statement → ensures that Dispose() is called even if an exception occurs.

```
private class SomeDesposible : IDisposible
{ Dispose()
{
}
```

```
using (SomeDisposibleType u = new DisposableType()) {
    operateonType(u);
}
```

→ mostly used for sql connection.
→ automatically dispose SQL connection after statement.

"is" vs "as". (operator)

is → operator is used to check type of object

```
o1 is P
```

```
P o1 = new P();
Console.WriteLine(o1 is P);
```

(true)

whether obj is of any class or not.

as operator → used to perform conversion betn compatible reference type

```
object[] o = new object[1];
o[0] = "Hello";
String str1 = o[0] as string;
```

is boolen | as is not of boolean.

keyword.

Readonly	constants
① using readonly we can assign value at decln or well as Ctor part.	① you can only asign value in decln part.
② value of readonly field can be changed. (Ctor III)	② cannot be changed.
③ Readonly keyword	③ constant keyword
④ ~~Compile~~ Run time const	⑤ compile time constant

VImp.

Boxing	UnBoxing
int num = 23; object obj = num;	object obj = num; int i = (int) obj;
process of value type → Ref type	Ref type → value type
Implicit conversion	Explicit conversion.

Scanned with CamScanner

String	String Builder
String str1 = "Interview" " str2 = "Happy" str1 = str1 + str2;	String Builder str1 = new StringBuilder("Interview"); Str str2 = "Happy"; str1.Append(str2)
→ Immutable → every time you assign value it create new string.	→ mutable → new obj बनाते है।
	If you want to change string multiple type

Nullable Types.

```
int j = null;
Nullable<int> j = null;
int ? j = null;
```

→ variable types don't hold null values.

If we req variable to hold " " we need nullable Types.

Scanned with CamScanner

Assembly is a unit of deployment EXE or DLL

→ create a code & build soln

→ .Net framework converts it into assembly.

inside bin folder

3 types.

Private Assembly

→ single appln ने use गर्ने।

→ not accessable outside (private)

Shared Assembly

opp of PA.

→ libraries/code which multiple appln can use.

→ global assembly cache मा stored हुन्छ।

Satellite Assembly

resx files.

→ Defined as an assembly with resource only not executable code.

GAC Global Assembly cache

→ place where shared assembly get saved in system.

Scanned with CamScanner

Imp: GC ? → Gurbage collector manages the allocation & release of memory.

→ memory release गर्छ
→ automatically
→ no need to code
→ routinely run हुन्छ।

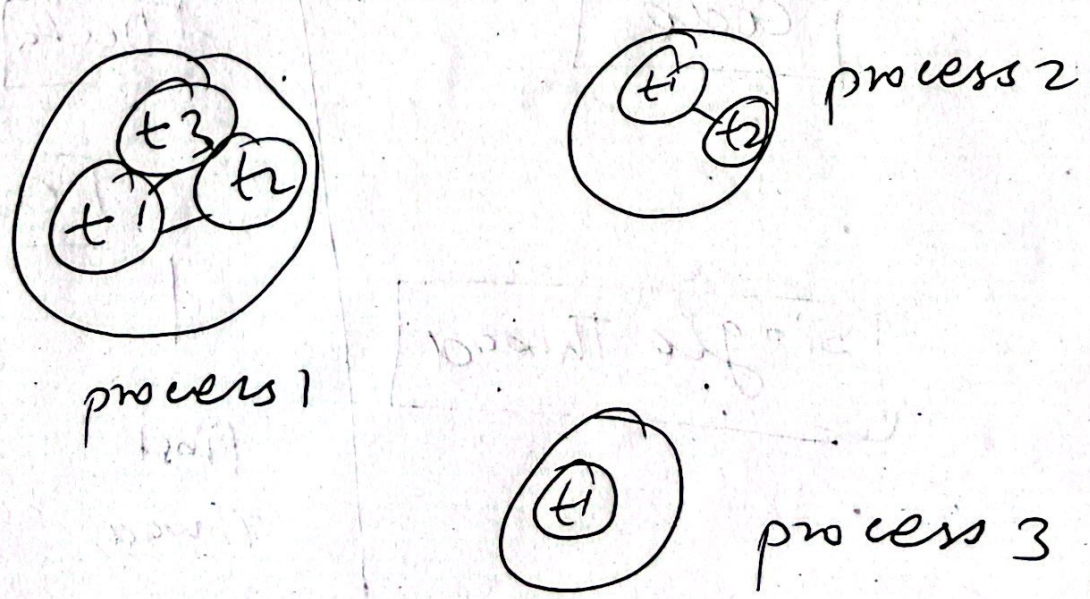
when GC runs all unused obj is released & memory is freed.

→ Can we force GC to run?
Yes. GC.Collect() but not recommended

prior to dispose

particular obj dispose गर्न dispose method use गर्न सकिन्छ।

Process vs Thread.



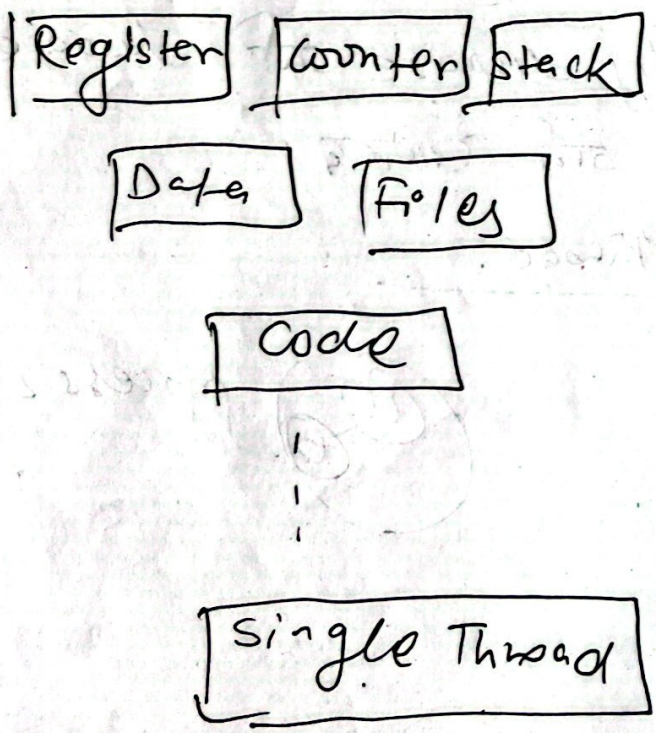
→ A process is an execution of a specific program /appln.

→ A thread is a smallest unit of process that can be performed in OS.

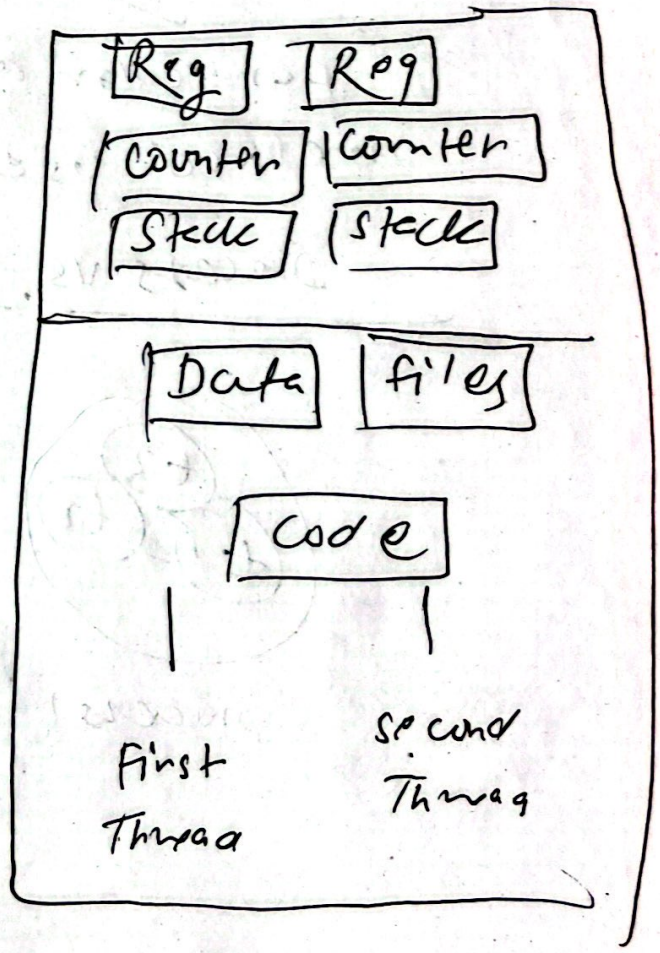
process are independent, thread can be interlinked with each other.

Multi Threading

Single Process Single Thread



Single Process Multiple thread.



~~multiple threads wor~~

→ Multithreading is a process in which multiple threads works simultaneously.

→ way to achieve multitasking

→ save time.

System.Threading Namespace

Reflection

→ Ability of code to access the metadata of the assembly during runtime.

→ metadata is info about data.

Book → Data

Author, booksize → MetaData है।

when is required?

① suppose you have to check version of assembly at runtime → use reflection.

Scanned with CamScanner

SOLID

design principle → helps us to manage most of software design problems

understandable
flexible
maintainable. → software Design

Single Responsibility Principle
open closed "
Liskov substitution "
Interface segregation "
Dependency Inversion "

SRP

→ class should have only one reason to change.
→ Every module or class should have responsibility over a single part of functionality provided by software
→ And that responsibility should be entirely encapsulated by class.

Liskov subs Principle

Scanned with CamScanner

SRP

Motivation

→ maintainability
→ testability
→ flexibility and extensibility
→ Parallel Development
→ Loose Coupling

→ SRP plays key role in maintaining all these.

class/module → focus on single task at a time
everything in class → related to that single purpose.

can be many members in a class → as long as related to single responsibility.

with SRP → class becomes smaller & cleaner.
→ code is less fragile

Login → LogException → send Email

```
interface IUser
{
    bool Login (string username, string password);
    bool Register (string username, string password,
                   string email);
    void LogError (string error);
    bool sendEmail (string emailcontent);
}
```

break this into.

```
interface IUser
{
  bool Login( string usesname, string password);
  bool Registes ( string usesname, string pwd,
        string email)
}

interface ILogger
{ void LogErrr (string error);
}

interface IEmail
{
  bool sendEmail (string emailcontent);
}
```

Scanned with CamScanner

Interface segregation Principle.

R.C.Ma (Xerox)

→ No client should be forced to depend on methods it does not use.

→ One fat interface → split into small & relevant interface so that clients can know about the interfaces that are relevant to them.

```
public interface IPrinttasks
{
    bool PrintContent (string content);
    bool ScanContent (string content);
    bool FaxContent (string content);
    bool PhotoCopyContent (string content);
}
```

HP printer द → सबै implement गऱ्यो।

Canon MG 2470 → print, scan, photocopy (no fax.).

↳ FaxContent → implement गर्न सकिन्छ

→ we need to handle.

(problems.).

Suppose: ADV भयो tech मा

```
bool PrintDuplexContent (string content);
```

↳ suppose यो add गर्न पऱ्यो

सब implementation मा असर

के गर्ने ?

(common)

```
interface IPrintScanContent
{
    bool PrintContent (string content);
    bool ScanContent (string content);
    bool photoCopyContent (string content);
}
```

(Fax).

```
interface IFaxContent
{
    bool FaxContent (string content);
}
interface IPrintDuplex
{
    bool PrintDuplexContext (string context);
}
```

OCP

(Bob martin M I principle of oop design)

"software entities such as class, modules, functions etc should be open for extensions but closed for modification".

→ New functionality should be implemented by adding new class, attributes & methods instead of changing current ones or existing ones.

Implement Guidelines

(Req change हुआ)

1. Implement new functionality on new derived class.
2. Allow client to access original class with abstract interface.

If Not followed in Req enhancement.

→ End up testing the entire functionality
→ QA need to test entire flow.
→ Costly process for organization.
→ Breaks Single Responsibility as well.
→ Maintainance overhead increase on classes.

Scanned with CamScanner

Liskov substitution Principle. (LSP)

→ if S is subtype of T, object of type T may be replaced with obj of type S.

→ Derived ~~class~~ types must be completely substitutable for their base types.

→ LSP is a particular defn of a subtyping relation, called strong behavioural subtyping.

→ Barbara Liskov.

→ extension of OCP.

Implementation Guidelines.

→ No new exceptions can be thrown by the subtype.

→ clients should not know which specific subtype they are calling.

→ New derived class just extend without replacing the functionality of old classes.

Scanned with CamScanner

suppose, contract employee → bonus calculate

Employee बाट. Temporary Employee, permanent Employee ले गर्ने।

Contract मा → Bonus calculate गरेर भने।

↳ NotImplementedException();

Then, Employee र EmployeeBonus लाई इन्टरफेस implement गर्ने।

(code हेर्ने, गरेर।)

Dependency Inversion Principle

R.C. Martin

(DIP.)

→ High level module should not depend on low level modules, but both should depend on abstractions.

→ Abstraction should not depend on details. Details should depend on abstractions.

→ Interaction betn high level modules and low level modules should be thought as abstract interaction betn them.

Scanned with CamScanner

High level

Present layer → Business layer → Data Access Layer.

Button click → Employee save → save Data in DB.

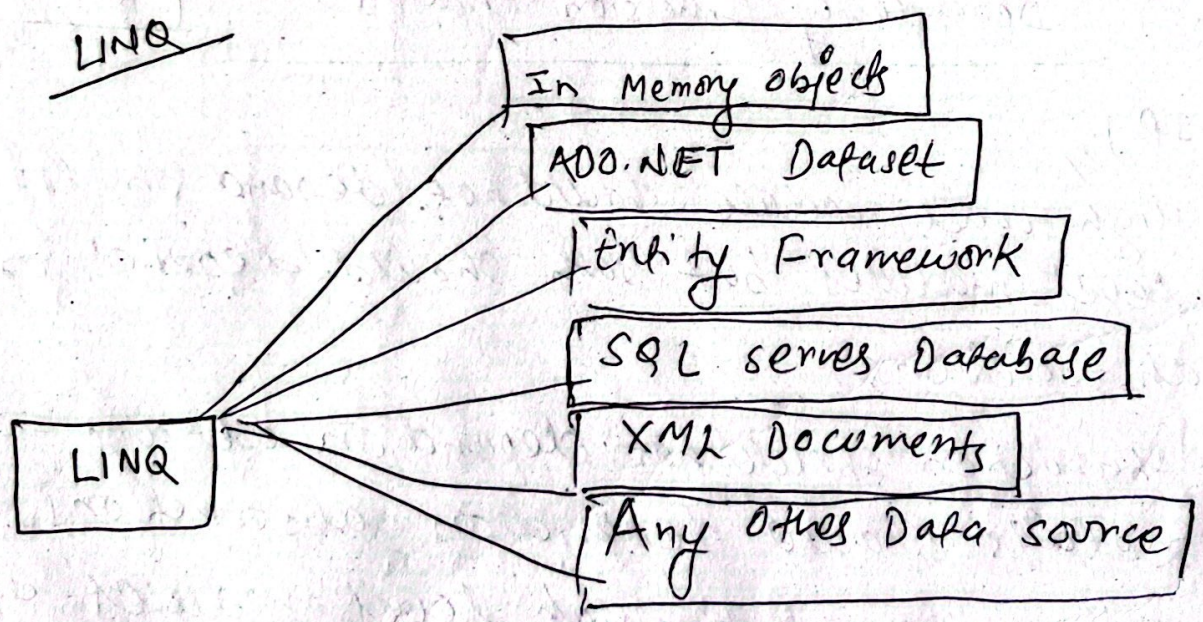
→ Unit test गर्न hard,

→ Couple भएर।

→ Abstraction (Interface) use गरेर decoupling गरिन्छ।

IRepository acts as abstraction

LINQ



Scanned with CamScanner