12 × 120 = 1440 min 60 × 1/2 = 30

↳ 23 Hrs

↳ Basic C#

Progim technology

Mr. Bangar Raju (18.30 Hrs) Naresh IT

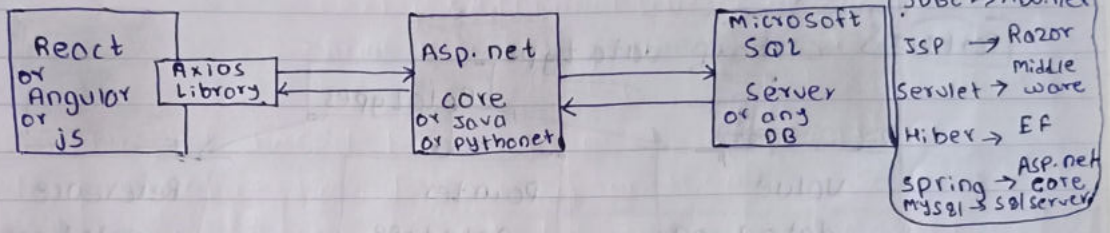
Kudvenkat → 102 vdo playlist 23-24 Hrs youtube - C#
- Asp.net core
- sql-server.

C# Basics + Advanced & Asp.net core [mvc]

offically 30+ long in C#
content in C#-sharp

complete C# Basics [core java]

1) Learn Architecture
2) Learn/Practice Progg in that long
3) Learn/Practice core of that long like OOP, Exch^n & multi thr
4) Practice above etc Peter till n.
5) Learn frameworks.



Asp.Net core is framework mode by Microsoft so it also allows cross platform & also is most widely used framework for creating web Applications.

DotNet Dev.

1) C# [Programming Lang]
2) Asp.net core
3) ~~MS~~ Sql server MS
4) Asp.net webApi
5) Entity framework
6) Linq to xml, Sql

console.writeline
↳ for displaying

console.Readline
↳ for taking keyword input.

FE → Html, Css, js
React, Typescript

Q How to Leor Programming Languages?

→ • See 18-30 min vdo of 100 vdo's
• watch on mobile, code on laptop [like Abdul bari.]
•

$12 \times 120 = 1440$ $60 \times \frac{1}{2} = 30$

↳ 23 HRS

↳ Basic C#

Mr. Bangar Raju (23 hrs) Naresh IT

Kudvenkat → 102 vdo playlist 23-24 HRS youtube - C#

→ Asp.net core

→ SQL-Server.

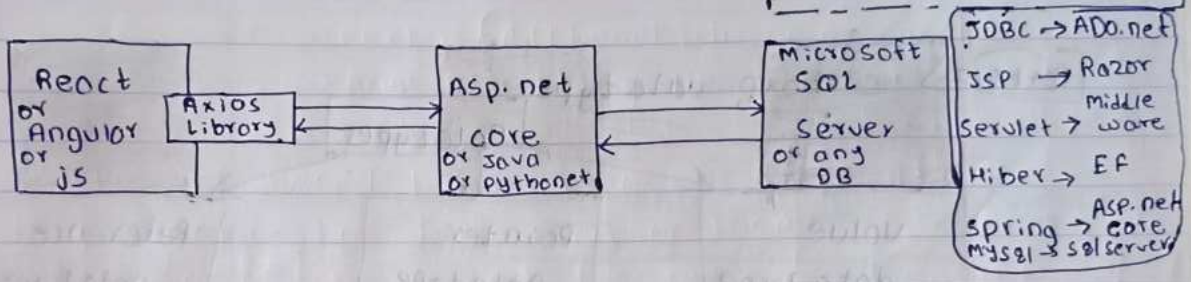
Progim technology

C# Basics + Advanced & Asp.net core [MVC]

offically 30+ long in C# content in C#-sharp

complete C# Basics [core java]

1) Learn Architecture
2) Learn / Practice Progg in that long
3) Learn / Practice core of that long like ODP, Exch^n & multi thr
4) ~~Practice~~ etc Practice above Perer till n.
5) Learn frameworks.



Asp.Net core is framework mode by microsoft so it ~~also~~ allows cross platform & also is most widely used framework for creating web Applications.

DotNet Dev.

1) C# [Programming Lang]
2) Asp.net core
3) ~~MS~~ SQL server MS
4) Asp.net webApi
5) Entity framework
6) Linq to xml, SQL

console.writeline
↳ for displaying

console.Readline
↳ for taking keyword input.

FE → Html, Css, js
React, Typescript

Q How to Leor Programming Languages?

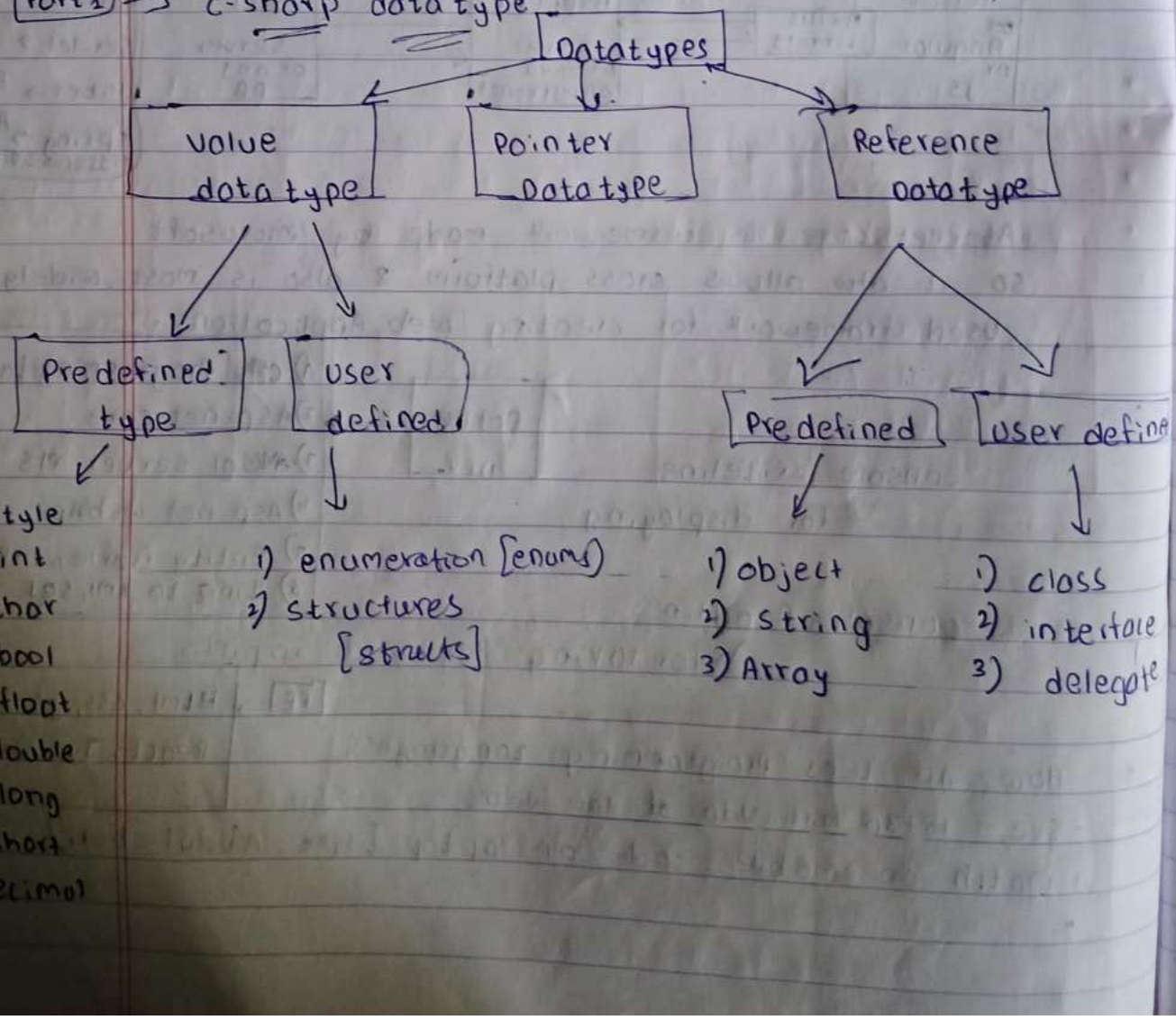
→ • See 18-30 min vdo of 100 vdos
• watch on mobile, code on laptop [like Abdul bari]
•

- .Net → 1) Desktop Application [C, C++, visual basic]
→ 2) web Application [PHP, ASP, JS, Java]
→ 3) Mobile Application [xamarin, Java swing]

variables, operators, Datatype, conditional statement

1) boolean
2) integer
3) double
4) string
} Datatypes

Part I => C-sharp data type



Part-2 => C-sharp Operators

1) Arithematic => +, -, *, /, %, ++, --

2) Assignment => =, +=, -=, *=, /=, %=, &=, |=, ^=, >>=

3) comparison => ==, !=, >=, <=, <

4) Logical => &&, ||, !

C-sharp conditional statements

1) if-else
2) else-if ladder
3) for-loop
4) while-loop
5) do while loop
6) break
7) continue
8) Switch

variables in C#

1) Local variable
2) Read-only variable.
3) constant variable
4) static variable
5) Instance variable

Fullstack .Net developer

Frontend
Javascript & ES6
Angular 2
React framework.

Backend	
C#	Asp.net web Api
Entity fw	Asp.net core
LINQ	

Database	
My SQL	RDBMS [SQL]
SQL server	
MongoDB	- NOSQL

cherry on cake
Design pattern
solid design principles
MongoDB [Non RDMS]
Source control
TFS or GIT

Pre-req	
Html	[Tailwind]
Css	[Bootstrap]

12 x 10D = 120D / 60 = 20 mins

Basic structure of c# program
what is Namespace
purpose of main method.

writing to console
Reading from console Console.ReadLine

```
main ()
{
    C.W ("please name");
    String user = Console.ReadLine();
    // C.W ("Hello" + user);          → contoctenation
}
    C.W ("Hello {0}", username);      → Placeholder
    C.W ("Hello {0} {1}", UP, LP);
    C.W ("Hello {0}, {1}", username); FN, LN);   → syntax
                F.N  L.N
```

Note: c# is case sensitive & i's pass by value bydefault, we can make it pass by ref by (ref & out)

no need of intialization

requires intialization before meth declaration

Part-

Part-3 ⇒ Build in types in C# [Data types]

① Integer type

```
main {

Bool  b = true;

  b = 12;
      ↘ error

}
```

// integer

sbyte	8 bit
byte	8 bit
char	16 bit
short	16 bit
ushort	16 bit
int	32 bit
uint	32 bit
long	64 bit
ulong	64 bit

float & Double

② Boolean Datatype

③ float - Point type table

```
int i = 0;
```

~~Bool int.MinValue~~

```
c.w ("min = {0}", int.minvalue);
c.w ("max = {0}", int.maxvalue);
```

implict & explict conversion.

④ Decimal Datatype
28-29 significant digits.

Part-4

⑤ String type

```
main {
  String name = "pragim";
  c.w (name);
```

② Escape sequence character [Backslash]

"\" Pragan\"";

C.w ("one\n Two\n Three") → new line

one
Two
three

~~Verbetion Literal~~

③ Verbatim Literal [to treat ES as regular character]

String name = @" ";

Part 5

common operator

1) Arithematic operator =
2) Assignment operator +, -, *, /, %
3) comporison operator ==, !=, >, <, >=, <=
4) conditional operator &&, ||
5) ternary operator ?:
6) null coalescing operator ??

```
main ()
{
    int m = a>b ? a : b;
}
```

Part - 6 ⇒ Nullable types in C#

Value types = int, float, double, Structs, enum
Ref types = interface, class, delegate, array etc. String

By default value types are non nullable
To make them null use ?

→ can't be nullable, can be made null by ?

value type

int ?j = 0 (j is nullable int, so j = null is legal)

Nullable types bridge Diff b/w C# types & OB types

user not so answering & saying No ⇒ so introduced nullable type.

Non nullable type to nullable type

```
AvailableTickets = (int) Ticketsonsale;

Available Tickets = Ticketonsale??;

main() {
    int ? Ticketsonsale = 100; , null
    int Available tickets = Ticketsonsale ?? 0;
                            ↓
                          if null
}
```

Part-7 => Datatype conversion in C#

1) implict conversion
2) explict conversion
3) parse() & try parse()
   ↳ method

class → Noun
method → Adverb
obj → Noun

Dataset

- upcasting is implicit
- downcasting is explict

float [upcasting]

float int a = (int) b;

float

float

int

[downcasting]

Parse() → converts string to req datatype

```
String str = "123"

int value = str.pare (str);

c.w (value)
```

tryparse() → converts string to req datatype
it handles exception, as if input string is invalid it throw exception

```
String str = "123abc"   int intvalue;
if (int.Tryparse (str, out intvalue))
{
   c.w (intvalue);
}
else
{
   c.w ("conversion faile");
}
```

int.parse ( );

1) ref Parameter

- when pass a parameter using ref, you are indicating that variable being passed must be intialized before method call
- inside method can read & modify value of ref

→ recieve

← return

2) out Parameter

- no need to intialize before using in method
- method does work of intialization

← return

ref

```
void modifyvalue (ref int x)
{
   x = x + 1;
}

int num = 5;
modify value (ref num);
C.w (num);      //6
```

out

```
void modify value
 (out int x)
{
  x = 10;
}
int num;
intialize value (out num)
C.w (num);
```

Part -8 => Array

C-sharp tutorial Datatypes [Done]

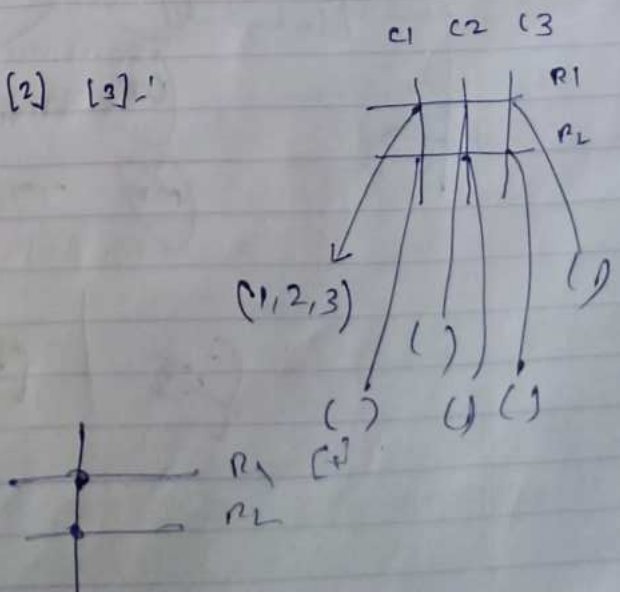
C-sharp Arrays
↳ Group of similar datatypes in one variable.

```
int [] a = new int [5];        a[0] = 2 , a[1] = 4 , a[2] = 6

int [] a = {1, 2, 3, 4, 5};
   for (int i = 0; i <= a.length -1 ; i++) {
   c.w (a[i])
   }
```

~~for (int~~

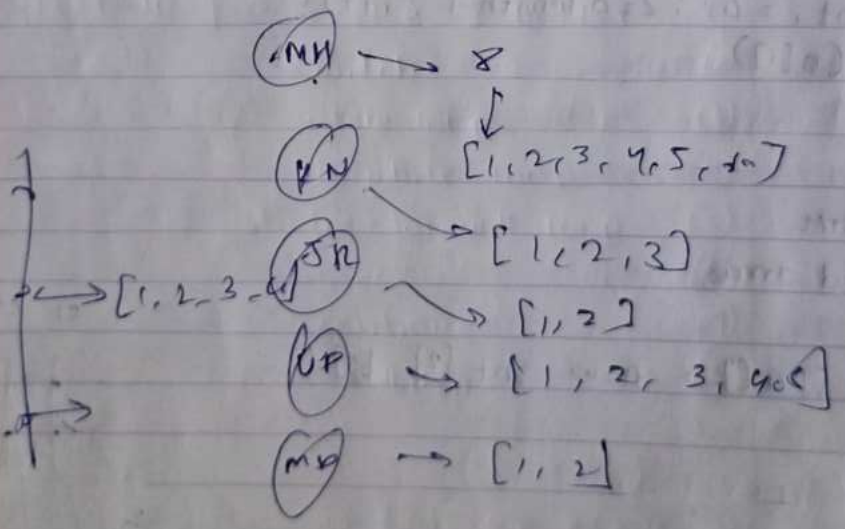
jogged array

```
int [] a [] = new int [2] [3];
```



```
foreach (int [] innerarray in jaggedarray)
{
    foreach (int value in innerarray)
    {
        C.W (value + " ");
    }
}
```

Domzons : 10 states



15 sept

Part -10 => Methods in C#

keyword → static, final, Abstract

return type

```
public void run() {
    C.w("speed");
}
```

method name

⇒ Access_Modifier datatype methodname () {

}

return type)

- .Net [30 + Language supported]
  1) Desktop Application - C, C++, Visual Basic
  2) Web Application - Java, python, .Net
  3) Mobile Application - React Native, flutte

Types of method →

1) instance methods
2) static methods
3) constructor methods
4) Destructor methods
5) Parameterized methods
6) void methods
7) value-Returning methods
8) Getter-setter methods
9) overloaded methods
10) Extension methods
11) Lambda exp method
12) Delegate methods
13) Asynchronus methods
14) iterator methods
15) Anonymous methods
16) Partial methods
17) Abstract methods

Method writing
method parameter
Method overloading
Method over-ridding
Method understanding

Mr. Bangar Raju
Mr. Kudvenkat

Constructor.

- It's special method under a class responsbe for intializing variable of that class & intializing obje
- name of constructor method is exactly same name of class in which it was present & more over it's non-value returning method.
- Each & every class req this constructor, if we want to create instance of class.
- Don't have constructor can't create ~~constructor~~ instan
- It's the responsibility of Programmer to define a constructor under his class & if he fails to do so, on behalf of programmer an implict constructor gets defined by class by compiler

```
class Test
{
  int i;
  public Test()
{
  i=0;          //intializing the variable,
}
```

String s, bool b is bydefault intiolized.

i = 0 ; integer → 0
s = null ; object → null
b = false ; boolean → false

Note :-
compiler defined the default constructor.

- implicitly defined constructor are also known as default constructor.
- implicitly defined constructors are Parameterless, then it is default
- implicity defined constructor are public.
- we can also define a constructor under class & if we can call it as explicit constructor

it can be
- parameter less
- parameterized.

} types

```
class Program
{
    int i;
    string s;

    main ()
    program p = new Demoproject.Program();
    C.W (p.i);
    C.W (p.s);
}
```

Syntax

[<modifiers>] <Name>> [Paramter list>]()

{ }

```
public  ExplictconDemo()
{
    c.w ("constructor is called");

main()
{
    ExplictconDemo obj1 = new ExplictconDemo();
    ExplictconDemo obj2 = new ExplictconDemo();
}
```

note:-

- constructor is called when obj is created to intiolize variables. ✓

Breakpoint - F9

press F11 [by pass]

- whenever create object then call goes to constructor.

calling: Implict or Explict

Nonstatic are always explictly called. but can be implicitly / explicitly created

calling must be done Explicitly, whether constructor is define explicitly or implicitly

Types of constructor

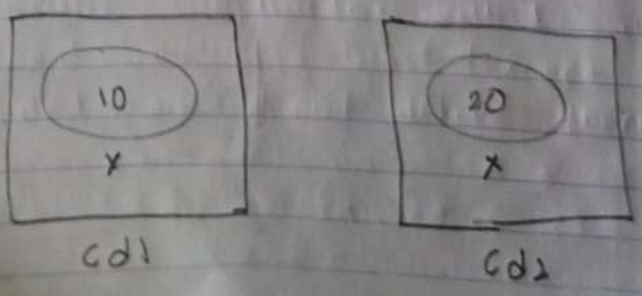
1. Default or Parameterless constructor
2. Parameterized constructor
3. Copy constructor
4. Static constructor

(1) - If a constructor method doesnot take any paramter then we call that as default or parameter less. These constructor can be defined by a Programmer explicitly or else will be default implicitly provided there is no explicit constructor in the class

- Implict constructor defined by compiler is parameterless. it is default

(2) if it is defined with Parameter ~~than~~ then explicity defined by programmer

```
class ParameterizedConDemo
 int x;
{
  public ParameterizedConDemo (int i)

  {
    c.w ("called");
  }

  static void main()
  {
     Pcd pcd = new Pcd (10);      cd1. Display();
     Pcd pcd = new Pcd (20);      cd2. Display();
}
}        void
    public display () {

    c.w ("value of x" + x);

  }
```



③ copy constructor

If we want to create multiple instance with same values then we use copy constructor, the constructor takes same class as parameter to it.

```
int x;

public copyconDemo(int i)
{
   x=i;
}

public copyconDemo (copyconDemo obj)
{
   x = obj.x;  (passing class as a parameter)
}

public void Display()
{
   c.w (x);
}

main()
{
```

Note: main method serves as a entry point to programme.

(4) Static constructor

if a constructor is explicity declare with static keyword than called static constructor all other constructor defined uptill now is copy constructor.

```
class Test
{
    static Test ()    // static constructor
    {                      defined explicitly

    }
    public Test ()    //implict default constructor
    {
    }
}
```

1) if a class contains any static variables then only implict constructor will be present or else we need to define then explicitly, whereas non static constructor will be implicitly defined in every class (except static class) provided we donot define them explicitly.

2) static constructors are responsible in intializing static variables & these constructors are never called explicitly they are implicitly called & move over these constructor are first to execute under any class.

static are explicitly created but always implicitly called

3) static constructor cannot be parameterized so overloading is not possible as implicitly called we cannot pass value.

4) they are implicitly private & no parameter as they are executed before main method, we can't intialize.

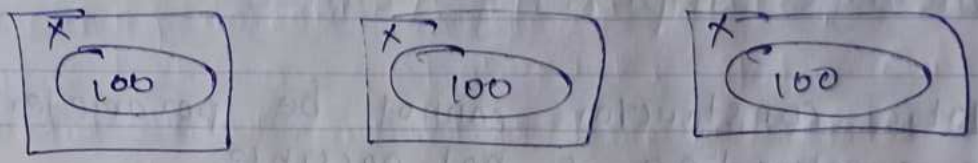
why constructor?

→ Every constructor has implicit constructor Init if we want to creat instance of class.

- Every class contains an implict cont constructor if not defined explicitly & with help of that implict container instance of class created.

Q what is need of define constructor explicitly again)

→ implict constructor of class will intialize variable of class with same value even if we create mutilpe instance of that class



f1 f2 f3

→ it will give same value for all instance, by implict constructor

→ if we define constructor explicitly with parameters then we will get chance of intializing the field or variable, with new value everytime, we are creating instance of class.

public int x; → class variable.

public st value (int x)

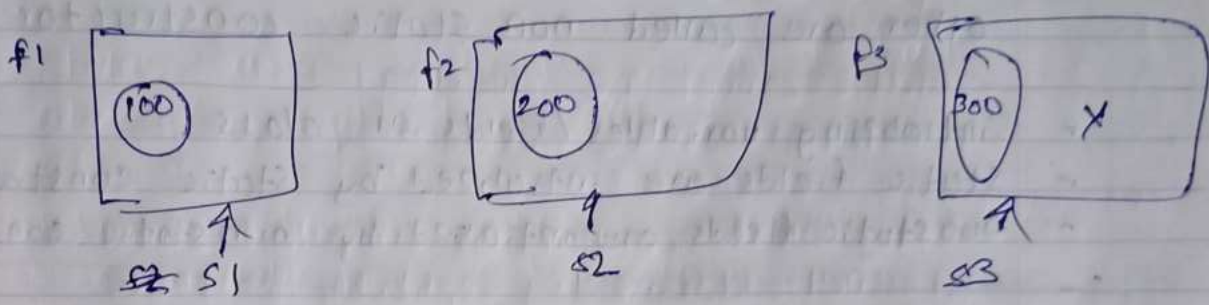
{

this.x = x;

} class variable x

local variable

value v1 = new value(10);
value v2 = new value(20);



- whenever we define a class first identify where, if class variable requires some value to executed, if they are req, define constructor explicitly, then create param constructor.

- Every time ~~one~~ instance created, then we get chance to input / passing new value.

- Generally every class requires some values for execution & the values that is req for a class to execute are always sent to that class by using constructor only

Access specifier

- special type of modifiers using which we can define scope of type & it's members

```
Public class ()
{
  Private void Test1()
  {
    c.w ("Private");
  }

  internal void Test2()
  {
    c.w ("internal");
  }
  protected void Test3()
  {
    c.w ("protected");
  }

  protected internal void Test4()
  {
    c.w ("protected internal");
  }
  public void Test5()
  {
    c.w ("public");
  }
```

```
main()
{
Program P=new Pr
P.Test1();
```

-1-2
-3
-4
-5

static constructor v/s Non-static constructor

- if a constructor is explicitly declared by using static modifier we call that constructor as static constructor whereas other are called non static constructor only.

- intializing variables / fields of class.
- static fields are intialized by static constructor
- nonstatic fields are intialized by nonstatic constructor

- static constructors are implicitly called
- nonstatic constructor must be explicitly called.
- nonparam constructor implicitly created.
- param constructor are explicitly created

- static constructor executes immediately once execution of class starts, its first block of code to run in class
- nonstatic constructor executes only after instance of class created & is executed always when an object is created.

- Entry point is always main method, but static blocks are always executed before the main method.

- we call constructor, by creating object.

In life cycle of class

1) static constructor executes only one time

2) nonstatic constructor executes N times when N times instance ~~of~~ created ore

- Static member access in static block

- Nonstatic member of a class can never be directly accessed from static block, can be accessed by class instance

- Static copy one time called because only one copy

- Nonstatic 'y' sep memory

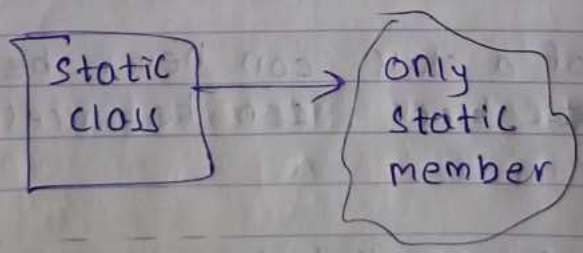
- Nonstatic const can be param, but static no param

- static connot hove ony porometer, they ore implicitly called ~~bow~~ who will pass value ??
- first block of code to run under class.
- that is only ~~bloc~~ first block how will pass value

- static cannot be overloaded
- nonstatic can be overloaded

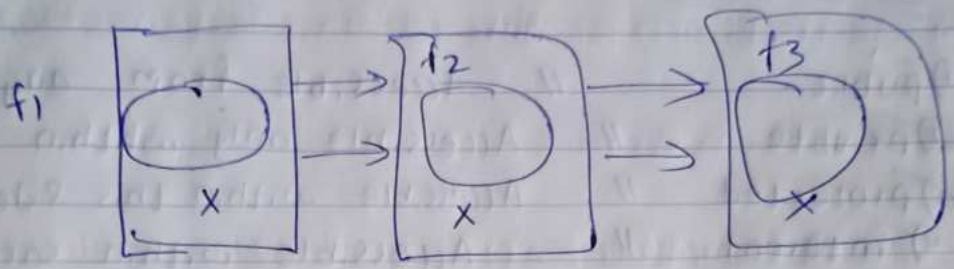
- Every class contain an implicit constructor if not defined explicitly & those implicitly constructors are defined based on.

=> Every class except static class have 1 implict ~~constructor~~ non-static constructor



- static constructor are implicitly defined only if, that class contains any static fields or else that constructor will be present at all.

f1.x



- Ref of class can be called as pointer to the instance & every modification we perform on members using instance reflects when we access those members thru reference & viceversa. it doesnot have its own memory

first f2 = f1;

1) Ref of class
2) instance of class [creating obj of class]
3) variable of class first f1;

first f1 = new first();

first f2 = f1();

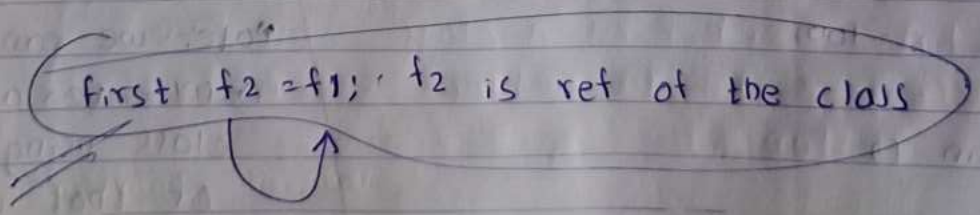
- memory allocation is happen only when instance is created

variable of eg class : copy of that class which is not intialized

→ instance of class : a copy of that class is that is intialized using new keyword. has it's own memory & never shared with other inst

— Ref of class : a copy of class that is intialized by using existing instance

- No. of instance created that many time memory is allocated
- memory allocated by one instance, is not reflected to other instance.
- Every instance unique, change in one instance, not affect other instance.



- ref doesnot have memory allocation, they will share some memory of instance assigned for

Difference between variable of class, instance of class & Reference of class

class :- user defined type

Every class is a type [user defined data type]

int = 100; X

int i = 100;

i → memory allocation

copy of type int

int → plan of House

i → House made of that plan

memory has to be allocated, class doesnot have memory

```
class first {
int x = 100;

static main ()
{
c.w (x);
        first f;
First f = new first();
c.w (f.x);
```

Note:- we can access variable of class using instance of that class.

f is variable of class

f is instance of class

Note :-
Any restriction is outside class not inside class.

```
class Two: Program1
{
    static void main()
    {
        Two t = new Two();

        t. test2
           test3
           test4
           test5
```

Note :- Default scope for members in class in private.
Private method access only within class
Every member in class has default scope as private
cannot make types private

Private
Protected
protected internal
} can't use class

public
internal
} class present.

1. public → anywhere
2. private → only same class
   internal → not in same proj
   proctected → not a child class

To make class private, make constructor of class private

Scanned with ACE Scanner

Access specifier

1) public // Accessible from anywhere
2) private // Accessible only within class
3) protected // Accessible within this & derived c
4) internal // Accessible within some assem
5) Protected internal // within some assebly or by d
6) private Protected // within some assembly by de
7) protelt private //
8) default //
   ↘ internal for types
   Private for members

will have detailed & practical lookup.

1) types (datatypes) - datamember
2) members (methods) - method member
3) fields (variables) -
4) Prop - (Get-set)

Variables

1) instance variable
2) local variable
3) static variable
4) final variable
5) constants
6) Reference types
7) value types

Detailed explaination with Practical Handson will be given

[crux]

1) public :- from anywhere accessible global

2) private :- only accessible within class

3) internal :- child & non child within same project

4) Protected :- child class within class within some Project [not allowed by non child class]

5) internal protected :- anyone allowed then allowed

case 1 :- consuming members of a class from child class of same project

case 2 :- consuming members of a class from non child class of some project

case 3 :- consuming members of a class from some class in same project

case 4 :- consuming members of a class from chid class of diff project

case 5 :- consuming members of a class from nonchild class of diff project.

Add Ref → inheritance b/w 2 classes
Present in different Assembly /Proj

Add ref of that assembly

2 two ways of consumption class

1) inheritance
2) object creation

④ internal → Default scope for class internal
child & nonchid within Project

⑤ protected internal

if any one is accessible
then both otherwise no

Case 2: Consuming member from child class of some Project

Case 3: consuming member of class from non child class of some Project

(3) Note:-

Protc ⇒ Protected method is consumed by only child class & within class

```
class program { }

class two
{
 main() {
  program p = new program();
  p.test1();
 }
```

Note:-

consuming member of class from child class

Case 1: consuming members of class from same class

Case 4: consuming member of class from child class in another project non child

Case 5: consuming member of class from child class in another project

Case 6:

During lifecycle of class,

- Static variable intiolized only one time
- instance variable intiolized N times, if N instance created
- intioliosation of instance variable is associated with instance creation & constructor calling, so instance variable intiolized by constructor

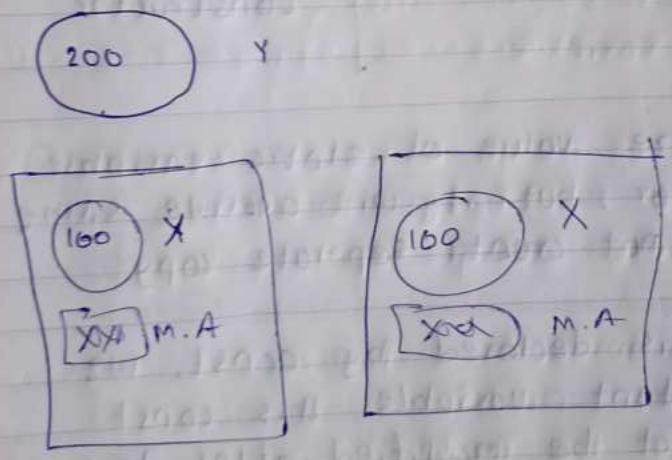
```
public Program (int x)
{
    this.x = x;
}
```

Param Constructor

- we can intiolize value of static variable by constructor, but it will overide same value but not create seperate copy
- if a variable is declared by const, we call it constant variable this const variable cannot be modified after declaration, so it's must to intialize constant variabe at time of declaration only

② Nonstatic [instance variable]

- memory allocates only when instance is created.
- Requires instance of class for both intialization & execution
- Static variables of class are intialized immediately once execution of class starts where as instance variables are intialized only after creating instance as well as each & every time instance of class is created



first instance P1

second instance P2

- Each & every time instance created that many time memory will be allocated

Different kind of variables in class

1) Non-static variable [instance variable]
2) static variable
3) constants variable
4) Readonly variable

```
class Programm
{
    int x=100; static int y=200;
    {
        main()
        {
✓       C.w(y);
        int z;
X       C.w(x);
        }
```

static variable

// ① any variable explicitly declared in static block then it is static
② explicitly declared using static keyword.
③ static variable is intialized once exec° of class start, no reqd to create instance
④ static doesnot req instance of class, to get intialized

① instance → maintains one copy for each time obj created (several copy) (modified) → inst
- 0 intialization
- 0 copy

② static → one value for whole class (one copy) (modified) → can be modified → intialized only one time

③ const → can't modify (one copy) (can't modify) → only one copy for complete Lifecycle of class → needs compulsory intializ? after declaration

④ Readonly → can't be modified (several copy) (not modify) → but after intialization not after declaration → multiple copy

CRUD

	Modified	Not modified
static		Readonly
nonstatic		const

	copy only 1	~~modified~~ copy more than 1
	static const	Readonly instance

2 copies, 2 instance by Readonly variable

Readonly bool flag;

~~this bol~~
this.flag = flag; → true
} constructor.

- modified can't be done after intialization of Readonly variable.

- Declaration time assign value, constructor assign value no change to change value once assigned.

Reodonly → copy for each instance
can't modified
nonstatic can be modified

- constant is fixed value for whole class, readonly is specific to an instance of class.

intialization after decl compulsory	not la static nonstat Readon
const	

Literal type cannot convert to float.

- No change of value, allocating memory only once happens for const
- Behaviour of constant variable, same as static i.e intialized only one time in life cycle of class, doesnot require instance of class.
- once execution start, Pi [const] intialized
- constant & static doesnot requires instance

Both same behaviour what diff static & constant

static → can be modified
constant → cannot be modified

(ii) Readonly
- By using Readonly keyword
- this variable cannot be modified, but after intialization
- it's not compulsory to intialize, readonly at time of decloration, can be intialize by constructor

Decloration & intialization same time ⇒ ① const → not modified
Decloration & intialization diff time ⇒ ② static → modified
by constructor ① non static → modified
② Readonly → not modified

[<modifier>] class <child class> : <Parent class>

~~Derive~~ Base class : Derived class

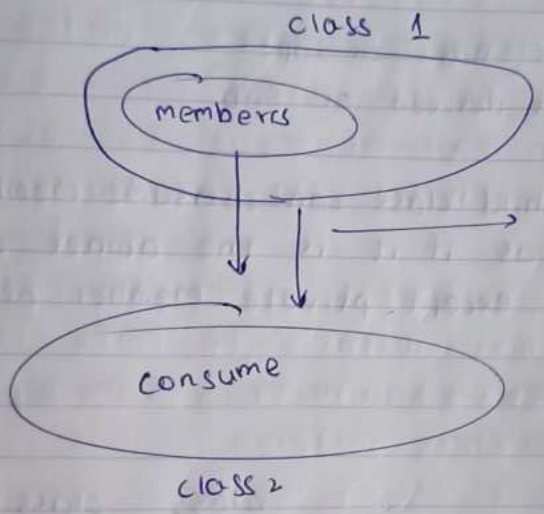
A → Parent or Base or Super
B → child or derived or Sub

- In inheritance child class can comsume members of parent class as if it is the owner of that Particular member except private member of parent class.

```
class class1                    class class2 : class1
{                               {
  Public void Test1()
{
  c.w ("m1");
}

  Public void Test2()
{
  c.w ("m2");
}
```

Inheritance

- It's a mechanism of consuming the member of one class to another class by establishing parent/child relationship b/w class



which provides reusuability & stops from ~~overridding the c~~ rewriting some code.

```
class A
{
  - members
}
```

```
class B : A
{
  - consuming members of
    A from here
}
```

OOP in C#

1) Inheritance → method overlooding (Static poly)
2) Polymorphism → method overridding (Dynamic poly)
   ↳ method hidding
   ↳ operator overlooding

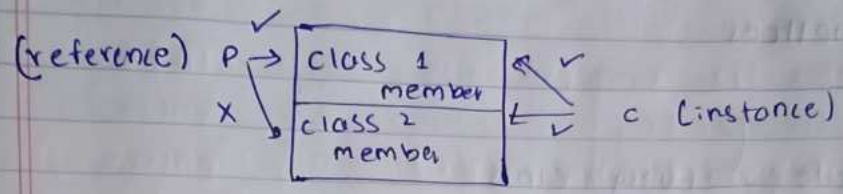
Method overloding → same ~~sign~~, name
Method overriding → same name, same sign

3) Abstraction → Abstract class
   ↳ interface

4) Encalpusalation → Get; set; [properties in C#]

Value type → structs & enums
in C#

reference doesnot have any memory



Note:-

reference doesnot have any memory

class 1 c = new class 2();

ref of parent class

obj of child class

```
c → meth 1 ()
    {
    c.w("met 1");
    }

p →   meth 1 ()
    {
     c.w ("child meth 1");
```

methods

1) ~~obj~~ are called based on obj hold

- Jvm call by obj hold
- compiler call by ref hold.

6) It should be accessible because, it intializes members of Parent class

(II) In inheritance child class can access Parent class members but parent classes can never access any member that is purely defined under child class.

X class 1 doesnot call class 2 constructor

(III) we can intialize a parent classes variable by using child class instance to make it as reference, so reference will be consuming memory of child class instance

this class cannot call pure child class members by using that ref.

class 1 P;

↳ P is variable [local variable]

[unitialized copy of class 1]

↓

intialized can be done using new keyword.

```
class 1 P;
class 2 C = new class 2();

P = C;
P.Test1();
P.Test2();
```

P → ref of Parent class created by using child class instance.

1) Parent class constructor must be accessible to child class, otherwise not possible inheritance
if constructor is Private of Parentclass then we cannot Perform inheritance.

2) Child class constructor implicitly calls Parent class constructor

3) Default scope for member in c-sharp is private

4) Execution always starts with Parent class constructor

class 1) ⟶ class 1 constructor called first
z)
:)
:)
)
)
9)

5) if class 2 has child class then that constructor has to be public

F9
F10 → run [step over]
F11 → forward [step into]

(VI) class 2 unable to call class 1 constructor if it is parameterized & value is not intialized

```
public class 2() : base (10);
{
 c.w ("class 2 constructor called")
}
```

Static value passing

Child class construct will b implicitly call Parent class constructor but only if constructor is parameter less, where as if constructor of parent class is parameterized child class constructor can't implicitly call Parent class constructor, so to overcome the problem it is responsibility of programmer to explicitly call parent class constructor from child class & pass value to those parameters to call parent's constructor from child class we use base keyword

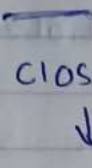
```
Public class 2 (int a) : base (i)
{                          10 ↑
 c.w (" ");                10
}
```

Dynamic value passing

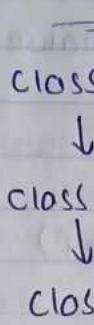
```
class 2 i = new class 2 (10);
```

Note:- constructor needs to be explicitly called by child class if parent class constructor is Parameterized using base keyword

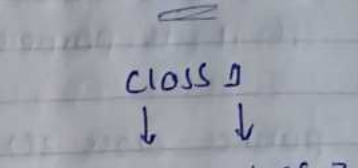
Single



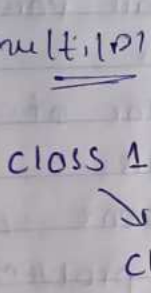
multi-level



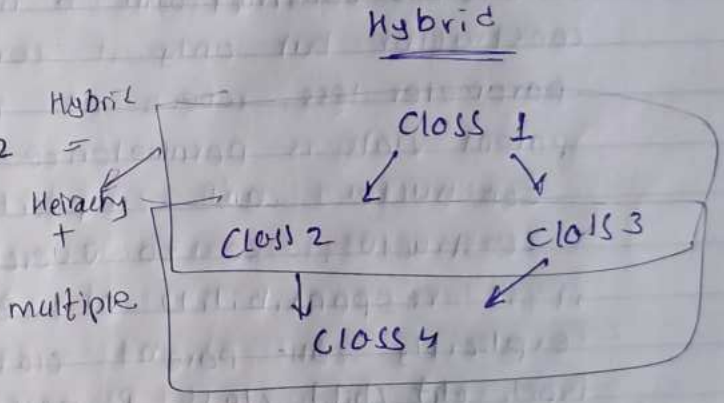
Hierarchal



multiple



Hybrid



1) single inheritance
2) multiple inheritance [more than 1 immidate parent]

(I) In csharp we dont have support for multiple inheritance only supported single inheritance.

(IV) Every class is predifened in library of long has default ↑ inherited from object class it has 4 methods

parent class i.e object class from System nomespace

Equals gethashcode full qualified name → gettype tostring	Every class has this 4 method accessible for all classes in .net

At time of compilation a class not inheriting from any class gets inherited from object class.

object { may be predefined or user defined father of all }
↓
class 1
↓
class 2

(V) Types of inheritance

No. of parent class a child class can have & vice versa

OBJECT ORIENTED PROGG CAME FROM C++

1) multi-level
2) multiple
3) Hierachal
4) Hybrid
5) single

identify common attributes of each entity & put them in hierachical order

Polymorphism

Method overloading

- more than one method with same name but different signature
  - ↳ parameters

✓ Public void Test () {}
✓ public void Test (int i) {}
✓ public void Test (String s) {}
✓ public void Test (int i, String s) {}
✗ public string Test () //invalid {}
✓ Public void Test (string s, int i) {}

only parameter change, takes into consideration.

- Parameter type must be changed due to ambiguity

note:- two method with same name & same paramter is amguigity

P Test();
↳ which method to call ??
is ambiguity

Q Why do we overload the method?

→ Polymorphism

Same name change behaviour based on input.

Behaving different — Based on — input change.

- multiple behaviour to a method (same name)

String str = "Hello world";

s.Indexof ('o', 5); } method overloading example.

Problem if no overlooading

- Each method different name, Headache for Programmer.

Approach of defining method with different behaviour where behaviour change depends on type, order, & no of Parameter.

Entity → Associated with set of Attributes

customer [Entity] → Shop, Bank

student [Entity] → School

living or non-living object associated with set of Attributes

Step 1 :- Identify entity that are associated with application we are developing

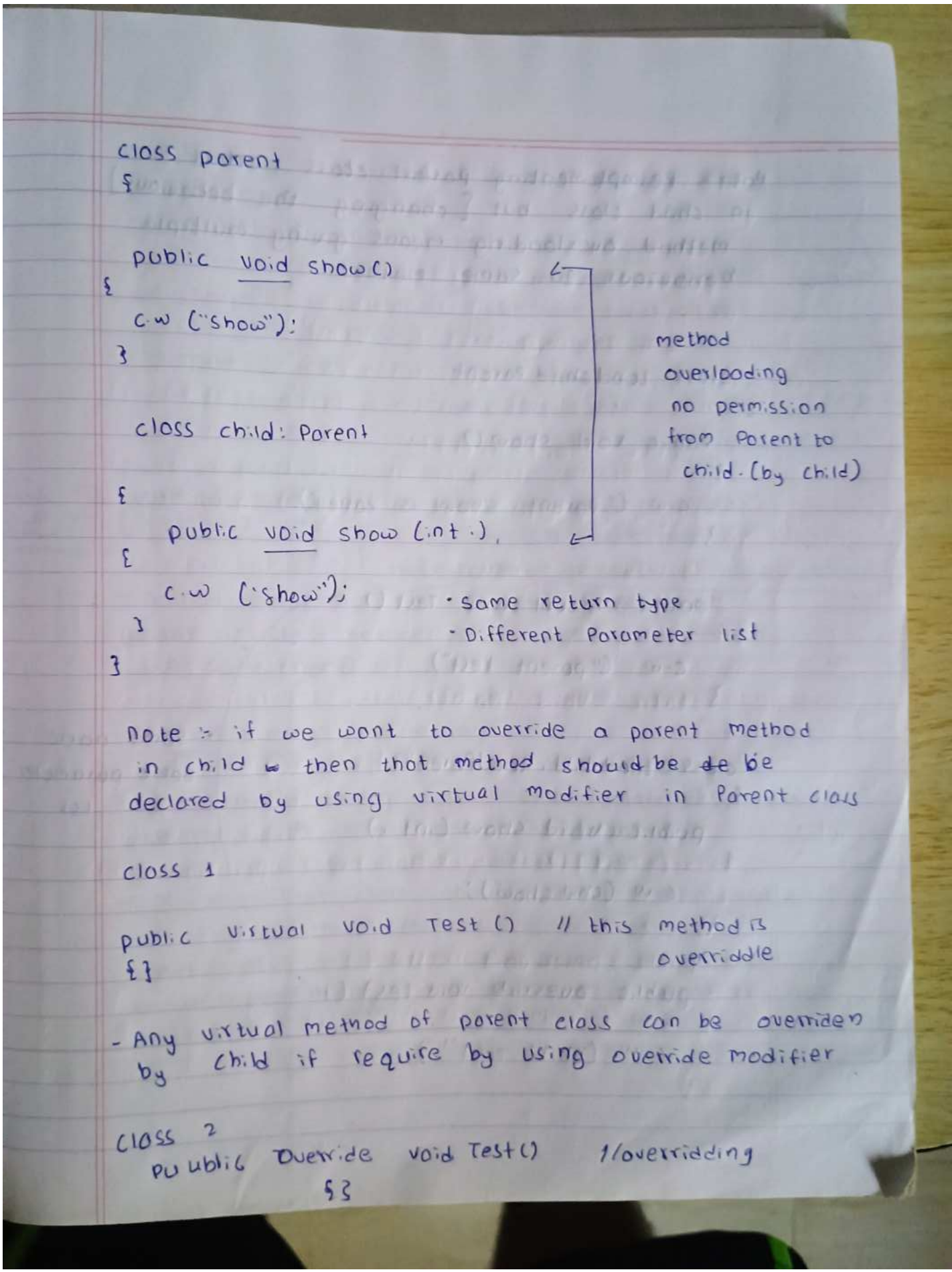
identify attributes of each & every entity

student	TStaff	NTstaff
id	id	id
Name	Name	Name
Add	Address	Address
phone	phone	phone
class	Designation	Designation
marks	Salary	Salary
Grade	Qualification	Dname
fees	Subject	

Note: Reimplementing parent class method in child class but [changing the behaviour] method overloading means giving multiple behaviour to same method

```
class LoadChild Parent
{
  Public void show()
  {
    c.w ("parent show is called");
  }

  public virtual void Test()
  {
    c.w ("parent test")
  }

class Loadchild: LoadParent
{
  public void show (int i)
  {
    c.w (Child show");
  }

  public override void Test()
  {
    c.w ("child Test")
  }
```

- Same name
- Same parame list



```
class parent
{

  public void show()
{
  c.w ("show");
}

  class child: Parent

{
    public void show (int),
{
   c.w ("show");
  }
}
```

method overloading no permission from Parent to child. (by child)

- same return type
- Different Parameter list

Note :- if we want to override a parent method in child then that method should be ~~de~~ be declared by using virtual modifier in Parent class

```
class 1

public virtual void Test ()  // this method is
{}                              overriddle
```

- Any virtual method of parent class can be overriden by child if require by using override modifier

```
class 2
  public override void Test()   //overridding
        {}
```

Method overriding in C#

Reimplementing parent class method under child class with same signature

```
class 1
 Test ()

class Test 2 : Test 1
 Test ()
```

	• overlooding	overriding
1)	multiple methods with same name by changing parameters	1) multiple methods with same name & same parameters
2)	This can be performed within same class or parent-class child	2) only parent-child requires inheritance never in same class.
3)	static polymorphism	3) Run-time polymorphism
4)	while overlooding parent class method under child class, no permission from Parent class	4) if done in same class ambiguity permission req from its parent class.
	Public string Print (int x) Public int Print (int x) NO overloading coverloading same return type but diff parameter list	5) overridding must have covariant return type or less restrictive access modifier in sub class.

Note:- Method can be reimplemented without using virtual keyword, but that concept called method hiding - we can/or not ~~ith~~ use new keyword in child class method that was overrided from parent class

- new [helps in remembering] that some method is ~~d~~ implemented in above class, we intinetionally ~~does~~ doing, re-method implementation is done.

polymorphism in C#

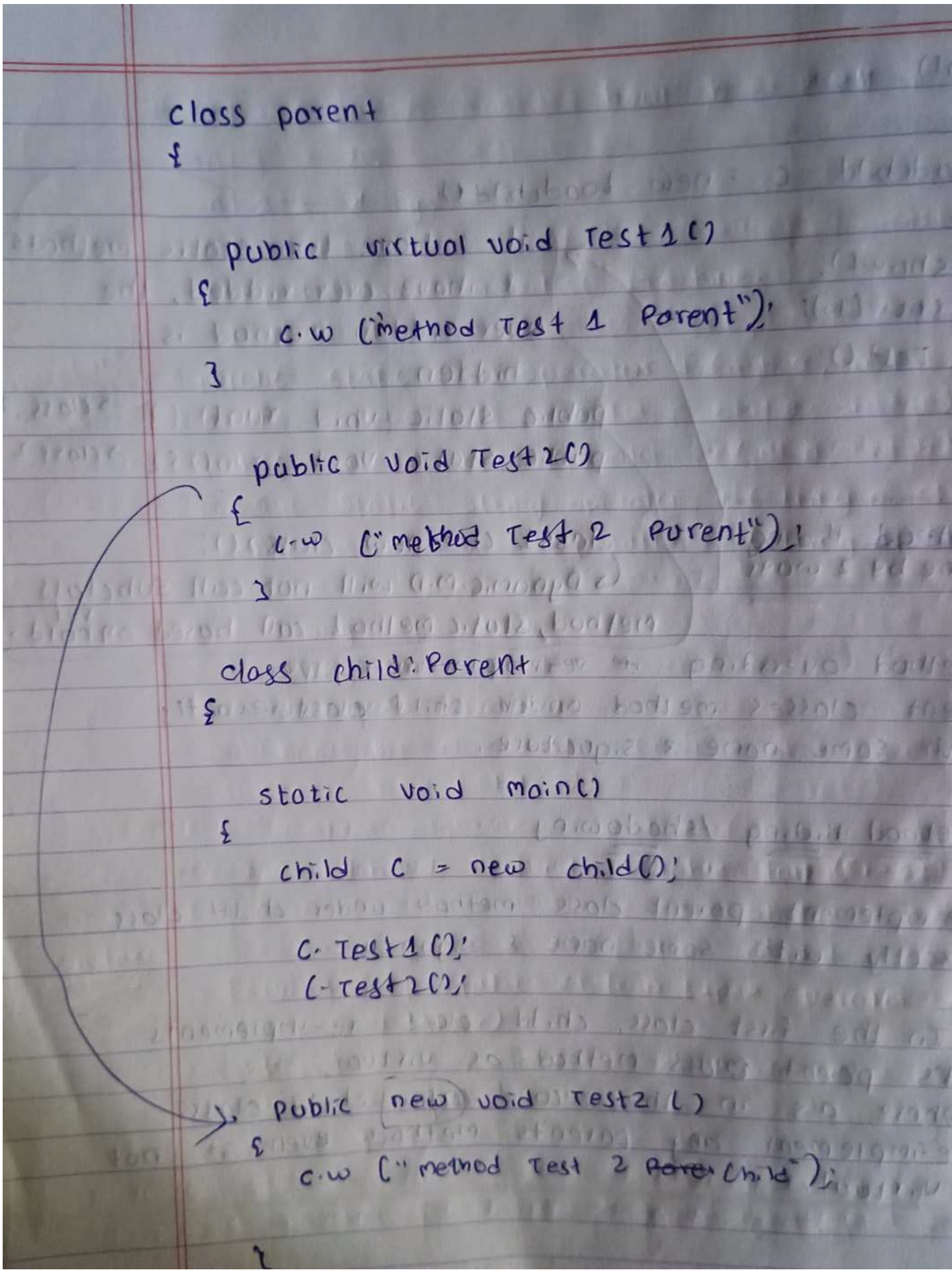
1) method overlooding
2) method overriding
3) operator overlooding:

- method overlooding is approoch of defining multiple behaviour to method & those behaviour based on parameter of that method

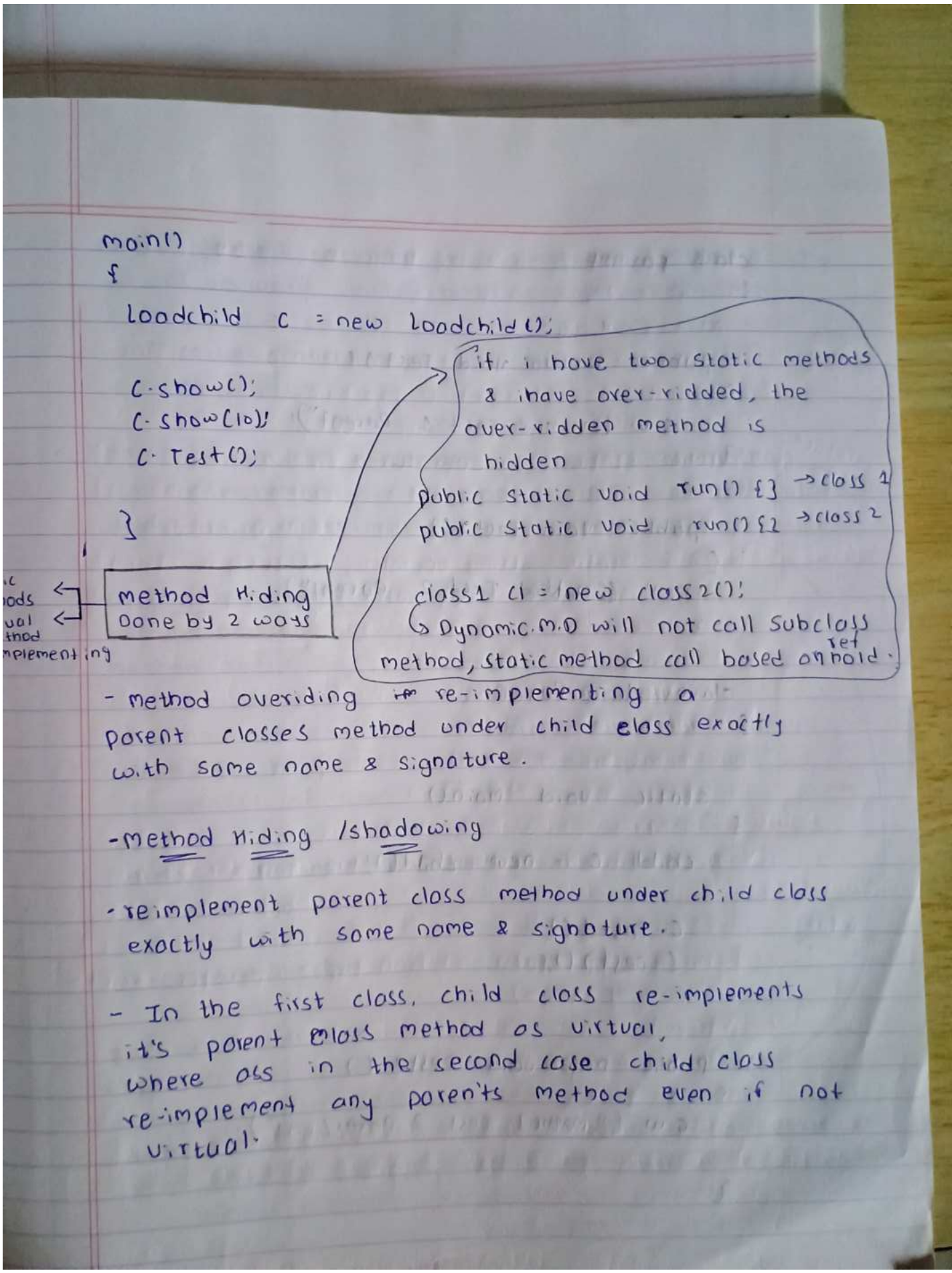
```
String str = "Hello how are you";
str.substring (4);          // you
str.substring (10);         // are you
str.substring (10, 3);      // are
```

operator overlooding:
approoch of defining multiple approach to behaviour & thos behaviour will vary based on the operand types between which the operator is used. For example + is an addition operator when used between 2 string operands.

Number + number => Addition
String + String => conroletion



```
class parent
{

  public virtual void Test1()
  {
    C.W ("method Test 1 Parent");
  }

  public void Test2()
  {
    C.W ("method Test 2 Parent");
  }

class child: Parent
{

  static void main()
  {
    child C = new child();

    C. Test1();
    C. Test2();


  public new void Test2()
  {
    C.W ("method Test 2 Child");

  }
```



- method overiding re-implementing a parent classes method under child class exactly with some name & signature.

- method Hiding /shadowing

- reimplement parent class method under child class exactly with some name & signature.

- In the first class, child class re-implements it's parent class method as virtual, where as in the second case child class re-implement any parent's method even if not virtual.

Note :- Designer of library not gave that method

Operator ~~Method~~ overloading

```
Public static matrix + (matrix obj1 , matrix ob
{
    matrix obj = new matrix (obj1.a + obj2.a ,
                             obj1.b + obj2.b ,
                             obj1.c + obj2.c ,
                             obj1.d + obj2.d)

    return obj;
}

Public static matrix - (matrix ob1 , matrix obj
{
    matrix obj = new matrix (obj1.a - obj2.a
                             obj1.b - obj2.b
                             obj1.c - obj2.c
                             obj1.d - obj2.d)

    return obj;
}
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

$a_{11} + b_{11} = 6$

```
class matrix
{
  int a, b, c, d;

public matrix (int a, int b, int c, int d)
 {
  this.a=a
       b=b
       c=c
  }    d=d

class Testmotrix
 {
  main()
{
  motrix m1 = new motrix (20, 18, 16, 14);
  motrix m2 = new motrix (12, 14, 16, 18);

  motrix m3 = new motrix
        (m1.a+m2.a, m1.b+m2.b, m1.c+m2.c)

  motrix m3 = m1 + m2;  } no implementation of logic
                          in library
   }
    Predifined method
    not available
```

Scanned with ACE Scanner

```
int x = 10;          bool b = x > y
int y = 20;
int z = x + y;
```

computer is brainless machine.

operator overloading

```
public static int operator + (int a, int b) {}
public static int operator - (int a, int b) {}
public static bool operator > (int a, int b) {}
public static String operator + (String a, String b)
Public static bool operator != (String a, String b)
```

```
String s1 = "Hello";
String s2 = "world";
String s3 = s1 + s2;
bool b1 = s1 == s2;
bool b2 = s1 != s2;
```

No predifined method available for (s1 - s2)

Syntax:

```
[<modifier>] static <return type> operator <opt>
(<operant types>)
{}  {
      -logic
    }
```

- The concept of Abstract method is near similar to method overriding

```
class  class1
{
  public  virtual  void show()
  {}

class  class2 : class1
{
  public  override  void show()     //optional
{
  Re-implementation
}
```

Note:-
virtual method can be reimplement but not compulsory

```
abstract class Program
{
  public  ~~ abstract  void  Test();

    public  void  Test()
    {
      C.w ("we are indian");

}
```

Abstraction → Abstract class
↳ Interface

Abstract method : a method without any method body is known as abstract method.

Abstract classes & Abstract methods

- method w/o any method body is known as Abstract method, what the method contains is only declaration of method

→ public void Add (int x, int)
{}

→ public abstract void Add (int x, int y);

```
abstract class Math
{
  public abstract void Add (int x, int y);
}
```

- if a method is declared as abstract, child class has to implement this method compulsory without fail

• Abstract class, variables, methods [imp modifier]
class → public &
methods → abstract, public
variables → instance, static, nonstatic, final

C → Users → HP → Source → repos → Dotnet

```
Public void static void writeline (object value)
{
string typeName = value.ToString();
                    ↳ name of the class.
writeline (TypeName);
}
```

returns name of class

c.w (m1)
↳ invoke writeline method takes obj as a parameter

public

Note :- so we override the tostring methods & change implementation & change it's original behaviour & tell it to return value.

why override tostring?
→ when we pass instance of class to c.w(c); it calls tostring method & returns name of class, so we override & change implementation of class & add obj

Interface

class is a user defined datatype
interface also user defined datatype

- class :- Non Abstract methods only → implict/explict constr
- Abstract class :- non Abstract + Abstract methods, I/E co
- interfaces :- only ~~non~~ Abstract methods only

Note : Every abstract method of an interface should be implemented by child c

Remaining Topics in C#

Complete Dotnet Backend Expert

- ✓ Interface in csharp
- ✓ structs & enums in csharp [value type]
- ✓ properties in chasharp [get-set]
- ✓ Delegates & types
- Extension methods & string classes
- ⊙ Generics & typesafe ~~ArrayList~~ ⟷ List<int> & collections
- ✓ Lambda expressions [short hand] & Anonymous method
- ⊙ collections [Dictionary, List, set]
- ⊙ Linq to sql, xml
- ✓ Exception Handling [try - catch - finally]
- ✓ multithreading & task
- ⊙ File IO in csharp complete

⊙ Kudvenkat → Lec 48, 52, 53, 54, 55, 56, 57, 58, 61, 62, 63, 101, 102

web dev

'Asp.net core + Asp.net web Api + Ms sql server + Entity framework + middleware

C# | Aspnet | Frontend | chemical.

SQL + Domain C# + Asp.net + Html f CSS JS + CHE

- Abstract class connot be static, static classes cannot be overridded, not depend on instance specific data
- Static class are made for singelton Pattern

- we connot create object of incomplete class ie Abstract class or interface

Q can Abstract method be static ?

→ NO, because we call it by not creating object then usecase is finish

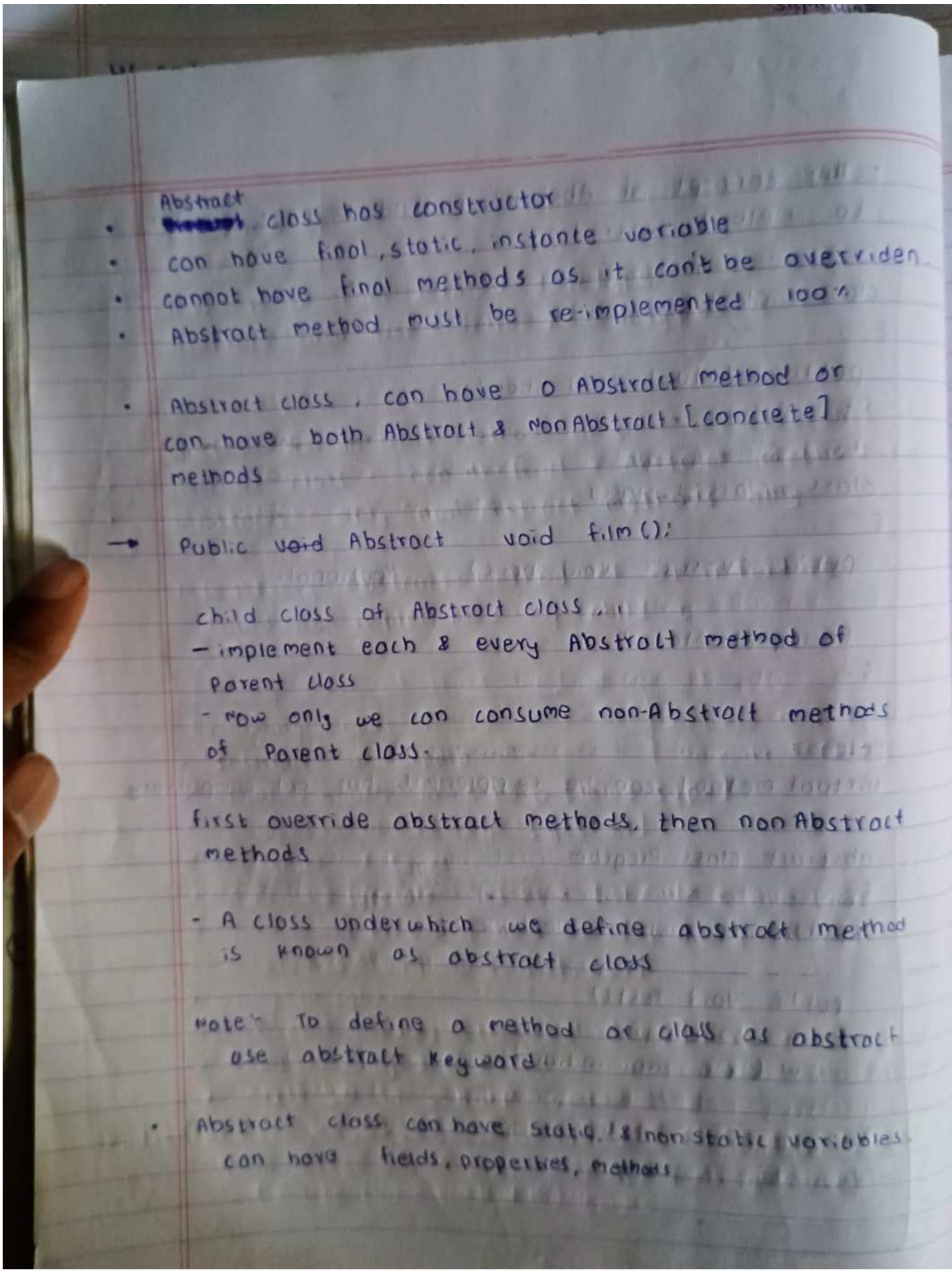
- STATIC METHODS connot be overriden as they belong to class, not any instance & this methods cannot be overrided or inherited, if we override static method it is called ~~or~~ method hiding
- tell we connot overide abstract method we connot create of ~~obj~~ obj of child class, that is inherited from parent class.

- STATIC METHODS Belong to class & not to particular instance, we can call them directly by class name

ClassName.methodName();

- we can create ref of parent class

- PARENT CLASS Ref can only call overrided method of Parent class but can never purely call, child class method with Parent ref.



Abstract class has constructor
- can have final, static, instance variable
- cannot have final methods as it can't be overriden
- Abstract method must be re-implemented 100%

- Abstract class, can have 0 Abstract method or can have both Abstract & NonAbstract [concrete] methods

→ Public void Abstract void film();

child class of Abstract class,,
- implement each & every Abstract method of Parent class
- now only we can consume non-Abstract methods of Parent class.

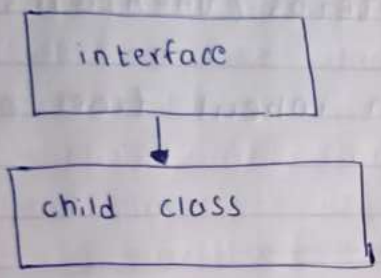
first override abstract methods, then nonAbstract methods

- A class underwhich we define abstract method is known as abstract class

Note: To define a method or class as abstract use abstract keyword

- Abstract class can have static & non static variables can have fields, properties, methods

) partial 6) Generic 7) Nested 8) Delegate
) operator overloading 6) final method



Note :- Interface have implicity defined constructor which is responsible for the inherited class which has overriden Abstract meth of interface

Interface A : Class B [
void run();
void run()
{cw("I run");

• Class B constructor give implict call to Interface A

Note :- Every abstract method should be implemented of interface in child class

• Generally a class inherited from other class to consume members of parent, where as if a class is inherting from an interface it is to implement members of its parent

• A class can inherit from a class & interface at a time.

ITest interface
[std name convection]

[<modifier>] class [<name>]
{
- Define members
}

[<modifier>] interface [<name>]
{
Abstract members dec
}

1) class :- 1) Public 2) internal 3) static 4) Abstract 5) seal 6) part
2) methods :- 1) Instance 2) static 3) constructor 4) Property 5) opera
3) variables :- 1) Instance 2) Readonly 3) const[final] 4) Readonly
5) Local variable [only within scope] 6) static

```
Interface KFC
{
    const int a = 20;        ✓
    Readonly int b = 30;     ✓
    , int c = 40;            x
    static int d = 50;       x
```

public abstract void dinner();

↓

By default all methods are public & abstract.

```
    void breakfast();
    void burger();
}
```

- Default scope for members in interface is public
- —"— in class is private.

	Interface	Abstract class
1)	contract of methods, properties, events or indexers that implementing class must provide, used to define set of operations that multiple unrelated classes can support.	1) Blueprint for other class contain mix concrete + abstract methods, prop fields & constructors,
2)	only method sign, prop, events & indexers cannot contain fields constructors or method implement	2) can contains field & constructor (with w/o param) implemented method, prop & abstract methods
3)	multiple inheritance	3) NO, multiple inheritance allowed.
4)	NO default implement^{n}	4) Def impl^{n} for some of members, ~~Derived classes~~
5)	NO instance	5) NO instance, but can create local variable or ref class Abstract A;
6)	share code & functionality amog derived class. Constants & Readonly	6) can be declare any variable instance Readonly static constant,
7)	ONLY ~~STATIC~~ VARAIBLES ARE PRESENT.	

```
using system;
namespace demoProject
{
struct [class] Mystruct
{
   public void display()
{
   c.w ("method in class");
}

  main()
{
  Mystruct m1 = new Mystruct();
   m1. Display();
   C.R();
 }
}
}
```

Diff b/w class & struct

1) class reference type (memory allocation on managed heap)
2) struct value type (memory allocation on stack)

1) memory allocation for instance of class on Mang heap
2) memory allocation for instance of struct on ~~not~~ stack

Structures in C-sharp (Structs)

=> value type

Structure is value defined type
- class is user defined type

In C-language procedural we learned structure, replaced by class in object-oriented

- Structure in C-language contain only field
- Structure in C# contain, most of member what a class can contain
  Fields, methods, constructors, properties, Indexers, operator methods etc

```
[<modifiers>] structs <nam
{
  -define member
}
```

use code file [blank.cs] file
↳ we can define class, interface, struct

- Default scope for method in interface is public
- Every member is abstract, no need abstract modifier again
- implicitly public & abstract.
- WE CANNOT DECLARE ANY FIELDS / VARAIBLES UNDER INTERFACE
- IF REQUIRED INTERFACE can inherit from another Interface.

```
public class shape
{
}
public class Recotangle circle : shape
{   // metho d of cal. area
}
public class Rectangle : shape
{   // method of calculate area
}
```

```
shape c= new cricle();
shape r= new rectangle();
```

polymorphism allows to treat object of ~~obj~~ both circle & Rectangle shape

```
c. calculate area ();
r. calculate area();
```

& oppropriate calculate area called based on actual type of object at runtime.

Con

1) we ~~can't~~ create instance of class w/o intializing instance variable that can be done by constructor after obj creation or initialize when declare.
   1) new keyword - used defout constructor
   2) int i = 100; declaration & initializ^n some time

2) ~~ob~~ we can't create instance of struct w/o intializing instance variable that can be done by

   1) new keyword - use default constructor
   2) m1. i = 10 - use explicit value
      int i = 100; X can't be done by struct allowed in class.

1) we can define param or non param constructor if no constructor defined their will be implict default constructor

2) In struct, parameterless cannot be defined b'coz it has defined constructor we can only define parameterized constructor as in class, once on defining the Parameterized constructor the no Param constructor is vanished but this doesnot happen in structure ie struct.

(II)

1) fields of class can be intialized of class at time of declaration
2) fields in struct not possible, if we want to intialize
   - constructor
   - after object intialization give value

(III)

1) Intialization of value in class can be done by constructor or by reference.
2) In struct intialization of variable can be done by reference or create instance by new keyword if fields are present

```
class mystruct [struct mystruct]
{
   int i;

 public void display()
 {
    C.W (i);
  }

  main()
 {
    myStruct m1;
     m1.i =10;      x [allowed in struct, not in class]
     C.R();  m1.display();
 }          C.R();
```

1) class used for representing entity for large volume data
2) struct for small volume of data

Note:- All Pre-defined datatype under libraries of our language which come under ref type category E.g string & object are classes, whereas all predefined data types

nor 32 b.t value type → structure [int, bool, char]
nor not define Reference type → class [string, object]

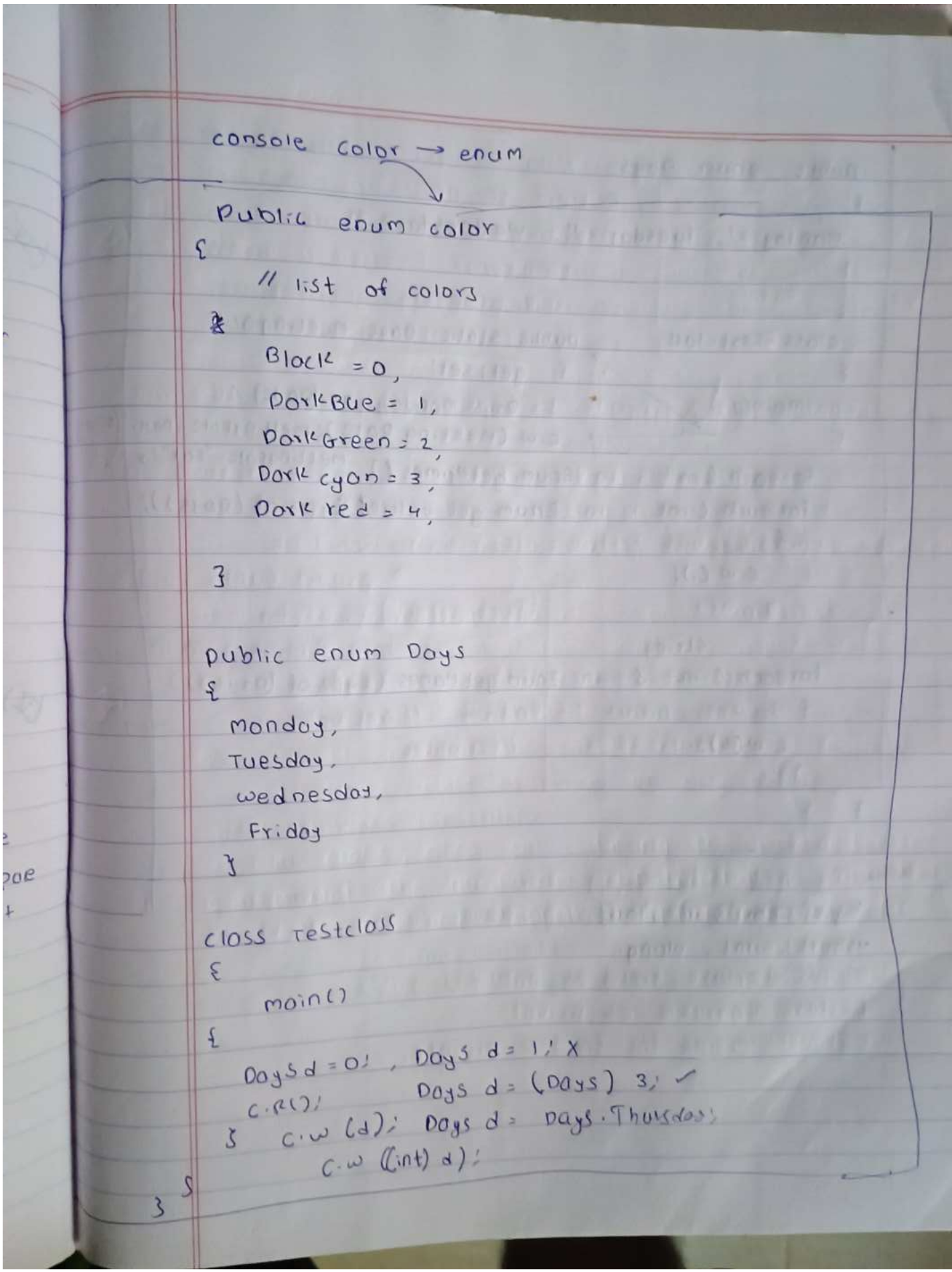
```
main()
{
  mystruct p1();
  p1.display();
}
```
✓ struct

```
main()
{
  mystruct p();
  p.display();
}
```
✗ class

(1) 1) In case of struct instance creation by new not compulsory.
2) In class compulsory

```
Public enum Days
{
  monday = 1, Tuesday = 11, wednesday = 21
}

class Testclass                    Public Static Days meetingDate
{                                  { get; set;
    main()                           } = Days.monday; or (Days).1;
{                                    c.w (meeting Date); mettingDate = Days.
   foreach (int i in Enum.getNames(1)   mettingDate = "sat";
   foreach (int i in Enum.get value(1 type of (Days));
     {
       c.w (i);
     }
          String
   foreach ( int s in Enum.get Names (type of (Days)))
    {
      c.w(s);
    }
}
```

byte, short, int, long
ushort, uint, ulong.



console color → enum

```
Public enum color
{
    // list of colors
    {
        Black = 0,
        DarkBue = 1,
        DarkGreen = 2,
        Dark cyan = 3,
        Dark red = 4,

    }

public enum Days
{
  Monday,
  Tuesday,
  wednesday,
   Friday
 }

class testclass
{
    main()
{
   Days d = 0;  , Days d = 1; X
   c.R();         Days d = (Days) 3; ✓
 }  c.w (d);  Days d = Days.Thursday;
       c.w ((int) d);
}
```

HS 256 → cryptographic Algorithm f...

7) (i) if zero constructor are defined in class after compilation where will be 1 constructor (implict) & if we define 'n' constructors in class after compilation 'n' constructors only

(ii) In structure if we define zero constructor after compilation 1 constructor, if define N constructor then after compilation (N+1) constructor

8) i) class can be inherited by other class

ii) structures can't be inherited by other structure, it does not support inheritance

9) i) class can implement interface

ii) structure also can define interface

collection of named constant ⟸ **Enumeration or Enum types** ⇒ value type user defined datatype

- [<modifier>] enum [<Name>]
  {
  - list of named constant values
  }

directly define under namespace class or struct possible to define

Note:- class, interface, delegate, Array are reference type

3 options:
1) only get
2) only set
3) Both get set

} By methods [Provide access to value]

Note:- Property combination of two method

```
[<modifiers>] <type> <Name>
{
  [get {stmts>}] // Get Accessor
  [set [stmts>}] // set Accessor
}
```

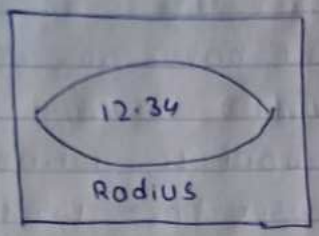
```
public double Radiusprop
{
 get { return Radius; }
 set { Radius = value; }
}
```

- only one ↑ name for getting & setting the value
- no need to declare variable
- combine both get & set in one block
- variable modifier implicitly declare
- Get accessor represent value return method w/o parameter
- ~~nor~~ set accessor represent nonvalue return method with parameter
- 2 member cannot have same name.

```
// Assign value

main()
{  Circle c = new Circle();
   double radius = c.RadiusProp;
   c.RadiusProperty = 56.78;
}
```

call get accessor of Property

call set accessor of property



Q How to access?
→ public radius
or
use property get; set;

```
public class Circle
{
  public double Radius = 12.34;
}
  public class TestCircle
  {
    main()
    {
      Circle C = new Circle();
      double radius = C.Radius;  // Get value
       C.Radius= 56.78 ; //set value
    }
  }
```

[Get access] public double GetRad
{
get return Radius;
}

~~prot~~ public void SetR
[set access] {
Radius = value;
}

Q why get set?
→ to provide restrictions, never declare field as public

- Never declare field as public
- Declare variable as private
- Default scope for field is private.

Q why Enum?

→ use Enum as type, so user donot have any chance of picking any value

→ once we create Enum, when we want to define property with set of values, user have limited opportunity

Properties in C# → Encapsulation.

- Property is member of class which can expose values associated with a class to outside environment

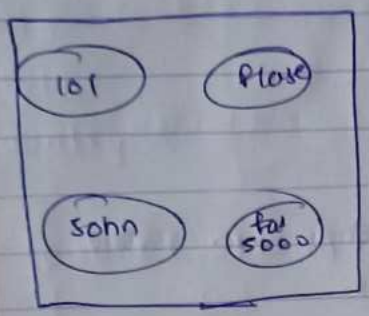
entity → Public class Circle

{

attribute → double radius = 12.34 [Default scope private]

}

```
public class TestCircle
{
    main()
    {
      Circle c = new Circle();
      c.radius(); [not accessed as private]
    }
}
```



obj
[memory allocation]

```
public bool status
{
  get { return _status;}
  set { _status = value; }
}

public string cname
{
  get { return _cname}
  set {_cname = value}
  }

public double Balance
{
  get { return _Balance: }
  set { _Balance = value: }
  }
```

Note:

1) if property has get set then it behaves like public accountnum!
   anyone can get, anyone can set [but we can validate]
2) if i provide only get property then we can't modify the value
3) Both get & set can have conditions.

COP + Python
COP + C-Lang
COP + Core java
COP + C#

=> Learn Basics of Programming first then Learn Advancement.

variable

```
public class customer
{
  int _custid;
  bool _status
  String _cname;
  double _Balance;
```

} never be public, if then accessible to anyone.

```
public customer (int custid, ...... )
{ this.customerid = id;
    this.customername = cname;
    this.Balance = Balance;
    this.status = status;
}
```

field & method name cant be same so -

properties

```
  public int custid
  { get { return _custid; } set { _custid = value; }
```

```
class Testcustomer
{
  main()
customer obj = new customer (101, false, "John", 5000)
  obj. x can not accessible
  c.w (obj.custid);    obj.custid = 12;
                       obj.status = false;
}                      obj.cname = "Sushant";
                       obj.Balance = obj.Balance - 1400;
                       Obj.Balance = ...
```

Scanned with ACE Scanner

if 100 fields expose only 10 then How to do

double _Radius [Prefix with underscore, if we wont to create Prop to that field]

As connot have two fields with some name

More adv of properts

1) conditional access & conditional assignment, if we wont to assign the value only if value greater than previous radius then write condition in set

```
set
{
  if (value > _Radius)
    _Radius = value;
}
```

1) Encapsulation
2) data validation
3) Access control

properties in class
1) get
2) set
3) get-set
4) conditional get
5) conditional set

Customer [Entity]

- custid
- status
- cname
- Balance

} Attributes

indexers in C#

- if we give indexer to class, it start behave like virtual array
- member of class
- starts behaving like virtual array
- only get value but not assign, it is diff from array

Employee → Entity
- Eno
- Ename
- Job
- Salary
- Dname
- Location

Attributes

indexer

```
public object this [int index]
{
  get { if (index == 0)
          return Eno; }
        else if (index == 1)
          return Ename;
        else if (index == 2)
          return job; return null; }
```

```
public class Employeee
{
  int Eno;
  String Ename, job, Dname, location
  double salary;

  public Employee (int Eno, string job, double slary ...)
  {
    this.Eno = Eno
    .
    .
    .
  }
```

HS 256 → cryptographic Algorithum for creating

city
state
country
} 3 attributes in city Entity

```
Public string city
{
   get { return _city; }
   set
   {
      if (_status == true)
   {  if (value == "Delhi" || value == "mum" || value == "kol" || value
       _city = value;
   }
}
}
```

value = Cities.Ben

main()

Create enum for city
{
   obj city = Cities.Bengaluru

enum

public enum cities
{ Bengaluru, Hyderad ---- }
}

Got options for selecting cities

```
Public string cname
{
  get { }
  set
  {
    if (_status == true)
      _cname = value;
  }
}
```

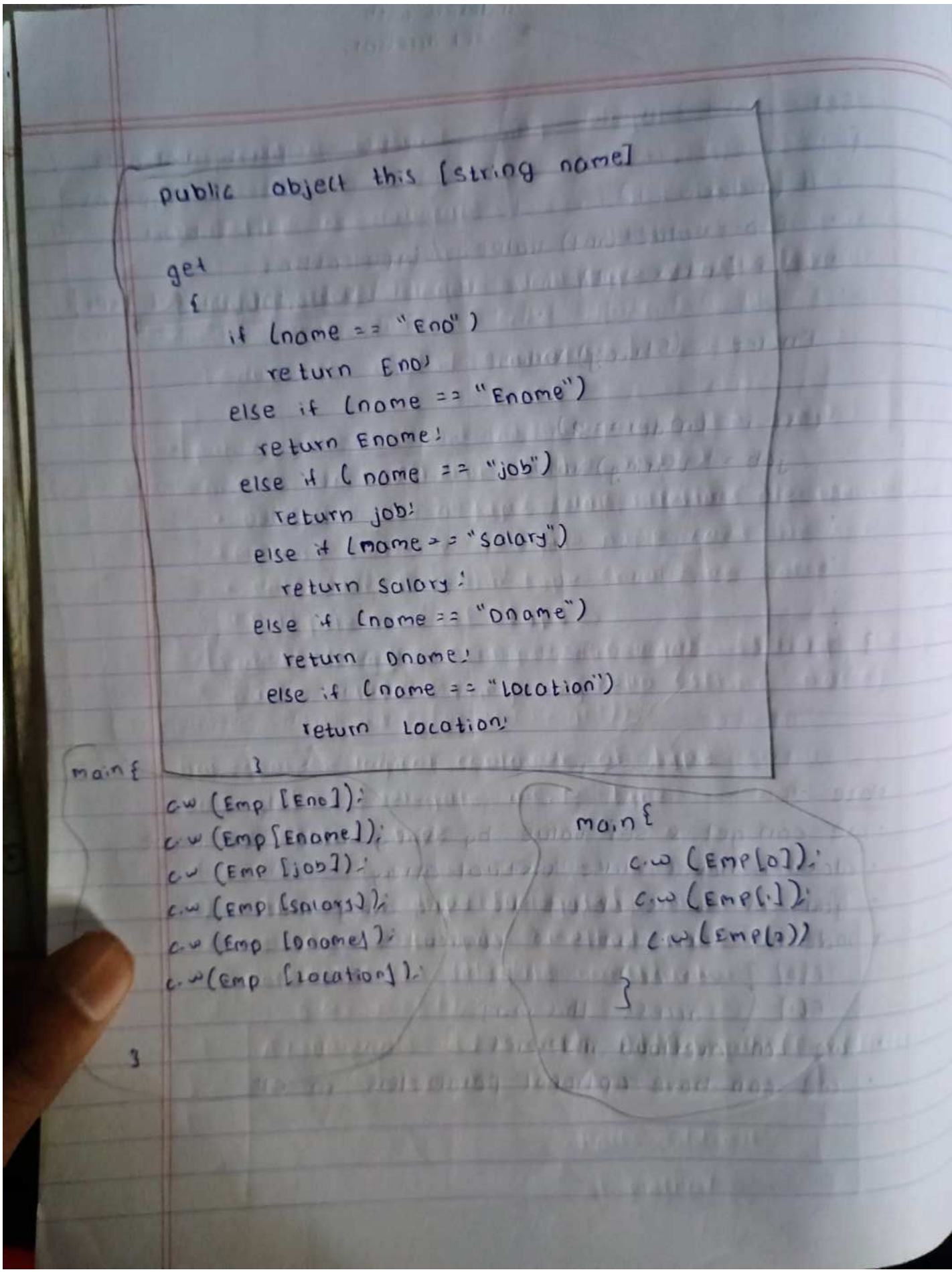
Note: we can provide condition, i.e if account inactive we cannot modify name.

```
public double balone
{
  get { }
  Set
  {
    if (_status == true)
      _Balance = value;
  }
```

Note: we are giving validations.

```
set
{
  if (_status == true) {
  if (value >= 500)
  {  _Balance = value;

  }
}
```

e]



```
public object this [string name]

get
 {
   if (name == "Eno")
     return Eno;
   else if (name == "Ename")
     return Ename;
   else if (name == "job")
     return job;
   else if (name == "salary")
     return salary;
   else if (name == "Dname")
     return Dname;
   else if (name == "Location")
     return Location;
 }
```

```
main{
c.w (Emp [Eno]);
c.w (Emp [Ename]);
c.w (Emp [job]);
c.w (Emp [salary]);
c.w (Emp [Dname]);
c.w (Emp [location]);
}
```

```
main{
  c.w (Emp[0]);
  c.w (Emp[1]);
  c.w (Emp[2])
}
```

indexers with set accessor.

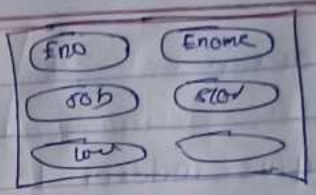
```
Set
{                                   value → implicity variable
                                            same datatype as
   if (index == 1)                          object
      Eno = (int) value; // type casting
   else if (index == 2)                [ value type to Ref type]

      Ename = (string) value;

      else if (index == 3)
         job = (string) value;
      .
      .
      .
      .
      .
}
```

Note :-

- we can get & set value by using index or name
- class behaves like virtual array.
- you can access object by using array index.
- indexers have pass 6 parameters
  - E[0] } we can
  - E[1] } access this way.
- we can overload indexeres
- we can have optional parameters in C#

Note :- In real project we pass only one value other loaded from database.

```
class TestEmployee
{
    main()
    {
        Employee Emp = new Employee
        (1001, "scoot", "manager", 2000, "sales", "mumbai");
```



Emp.

Emp. x [default scope is private]

Emp[0]; [if it was array]

```
Emp[3] = "manager";   C.W [Emp[0]]; C.W (Emp[2]);
Emp[2] = "rohan";     C.W [Emp[1]]; C.W (Emp[3]);
}
```

Note:- set get

value of class by index position

Syntax :-

```
[<modifiers>] <type> this [int index]
                          <Parameter list>
{
    [get <{stm}>]
    [set <{stm}>]
}
```

```
public object this [int index]
{
    get{if (index == 0)
            return Eno;
        else if (index == 1)
            return Ename;
        else if (index == 2)
            return job;
        else if (index == 3)
            return
        else if null; }
```

→ indexers with get accessor

Note :- WE CAN NEVER ACCESS NONSTATIC MEMBES UNDER STATIC BLOCKS WE NEED to CALL BY INSTANCE OF CLASS.

```
SayDelegate sd = new sayDelegate(Program.Say Hello);
```

3) call the delegate by passing req parameter value, so that internally the method bound with delegate get executed

```
od.(100,50);
String str = sd("Raju");
sd.Invoke(100,50);
```

Note:

1) Delegate is not a method or doesnot have any body
2) it is reference pointer to a method
3) passing a method to delegate constructor & invoking delegate, internally invoves that particular method which was passed as Parameter.
4) Bind a method with delegate.

- To call a method using delegate

1) Define a delegate

[<modifiers>] delegate void | type <name> [<parameterlist>]

[same as method signature]

→ public delegate void AddDelegate (int a, int b); — Delegate

→ public void AddNums (int a, int b) — method

- Delegate is a type, class, interface are user defined ty
- method name & delegate name can't be same
- Parameter & return type should be same
- Delegate is ref type

→ Public delegate string SayDelegate (string str) — Deleg

→ public static string sayHello (string name) — meth

namespace :- logical container of type
can define delegate inside type
i.e Nested type

2) instance of type [Delegate]

- we require to pass method name as parameter to delegate constructor

```
AddDelegate ad = new AddDelegate (P.addnums);
```

Holds address of method

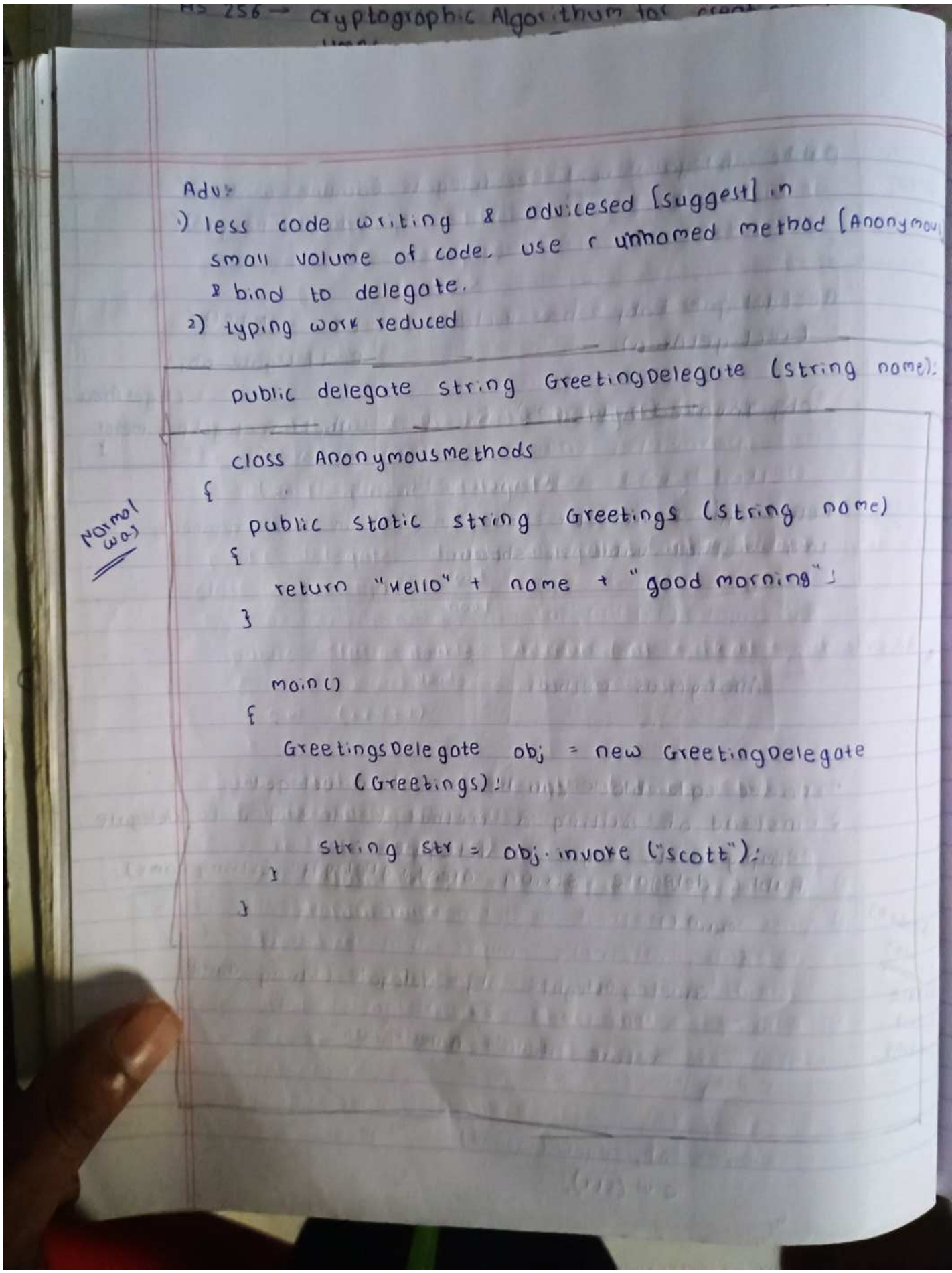
Delegates

Delegate:- type safe function pointer

holds reference of a methods & then calls the method for execution.

1) If call method by creating object if non static
2) —"— by not creating object if static
3) via delegates.

```
class Program
{
    public void AddNums (int a, int b)
    {
        C.W (a+b);
    }

    public static string SayHello (string name)
    {
        return "Hello" + name;
    }

    main()
{
    Program P = new Program();
    P.AddNums (100, 50);
    string str = Program.SayHello ("Raju");
    C.W (str);
    C.R();
}
```



Public delegate void Rectdelegate (double width,
double height);

```
Rectdelegate obj = new Rectdelegate
    (rect.getArea);

obj.Invoke (12.34, 56.78);
```

or

```
main() {
    Rectdelegate obj =
        rect.getArea;
    obj += rect.getPerimeter;
}
```

- Return type & parameter should be same, in value returning last method value is shown.

Anonymous methods

- unamed code block can be binded to delegate
- instead of writing a method & binding it to delegate simply write logic in delegate

```
Public delegate string GreetingDelegate (string name);

main()
{
    GreetingDelegate obj = delegate (string name)
    {
        return "Hello" + name + "sus";
    };

string str = obj.Invoke ("Scoot");
c.w (str);
}
```

Multicast Delegates

- Return type same } of 2 methods
- Parameter same } use multi delegate

- Hold a reference of more than one method
- can call method with same signature with same delegate

```
class Rectangle
{
    public void GetArea (double width, double Height)
    {
        C.W (width * Height);
    }
    public void Getperimeter (double width, double Height)
    {
        C.W (2 * (width + Height));
    }

    main()
{
    Rectangle rect = new Rectangle();
    rect.getArea (12.34, 56.78);
    rect.get Perimeter (12.34, 56.78);
    C.Readline();
}
}
}
```

```
Func <int, float, double, double> obj1 = (x,y,z) =>
{
    return x+y+z;
};

main()
{
    Delegate1 obj1 = new Delegate1 (AddNum1)
    Func <int, float, double, double> obj1 = AddNums1;
                                              o/p
    double result = obj1.Invoke (100, 34.5f, 193.465);
    C.W (result);

    Delegate2 obj2 = AddNum2
    obj2.Invoke (...);

    Delegate3 obj3 = checklength;
    bool status = obj3.Invoke ("Hello");

    obj3.Invoke (status);
}
```

[non value return]

Action <int, float, double> obj2 = AddNum2;
i/p i/p o/p

Predicate <int, float, double, bool> obj3 = checklength;
i/p i/p i/p o/p

func → value return method
Action → void (non value) return method
Predicate → bool return method.

Note:
Generally

comb^n of delegate & anony & lambda exp.

```
Action <int, float, double> obj2 = (x,y,z) =>
{
    C.W (x+y+z);
};
obj2.Invoke (100, 32.4f, 193.465
```

HS 256 → cryptographic Algorithum for creat...

Note

- Donot use String modifier or any modifier already mentioned in delegate

Syntax ( ) =>

Func, Action & Predicate Delegate

- Predefined generic delegate

```
public delegate double/void/bool Delegate1 (int x, float y, doub
class GenericDel
{
  public static double/void Addnum (int x, float y, double
  {
    C.W (x+y+2);
  }
  public static bool checklength (string str)
  {
    if (str.length > 5)
      return true;
      return false;
  }
```

Lambda expressions

- Lambda exp is short hand to write anomoyous method

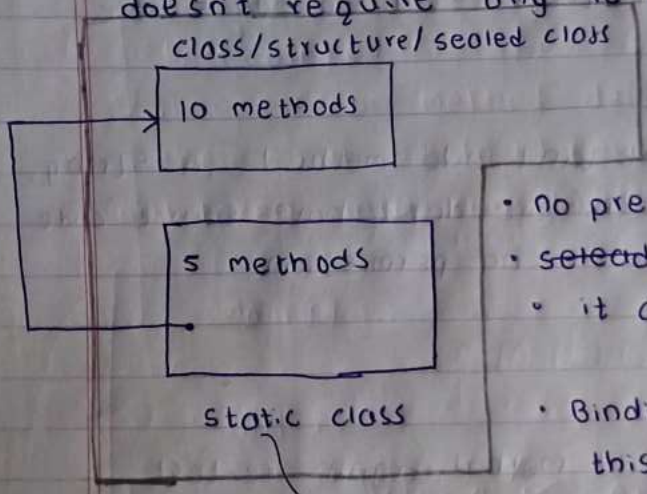
```
public delegate string Greetingdemo (string name);
class LambdaExp
{

public Static String Greetings (string name)
{
  return "Hello" + name + "a good morning!"
}

  main()
{
GreetingDelegate obj = new GreetingDelegate (Greetings);

string str = obj.Invoke("Raju);
  c.w (str);


}
```

replace

```
GreetingDelegate obj = (name) =>     [LAMBDA EXP]
GreetingDelegate obj = delegate (string name)
{
    return "Hello" + name "welcome";
};
string str = obj. Invove ("Raju");
    c.w (str);
}
```

HS 256 → cryptographic Algorithum for creation

- mechanism of adding new methods into existing class or struture also with out modifying the source code of original type. & this process no permission from original type & original type & doesn't require any re-compilation.



- no premission
- ~~select~~ sealed class
- it can be strut

} then also can add method via Extended methods concept.

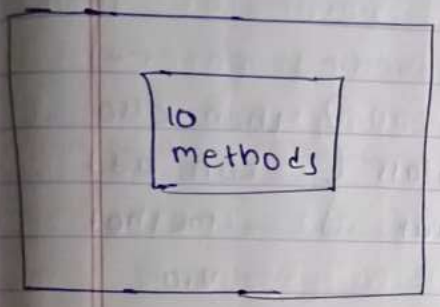
- Binding done by this keyword in static class ()

```
Class Programm
{
    public void Test1()
{
    C.w ("meth 1");
}

    Public void Test2()
    {
        C.w ("meth2");
```

```
main()
{
programm P= new Programm();
P. Test1();
P. Test2();
}
```

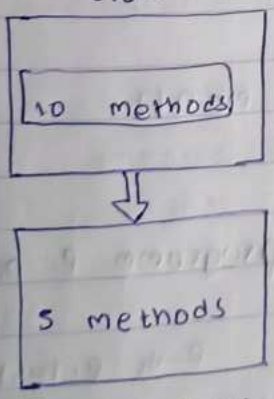
Extension methods

- New feature added in cshorp 3.0
- if a class has 10 methods wont to add 5 more methods in a class



- we can't add method in String, as we don't have source code or no permission.

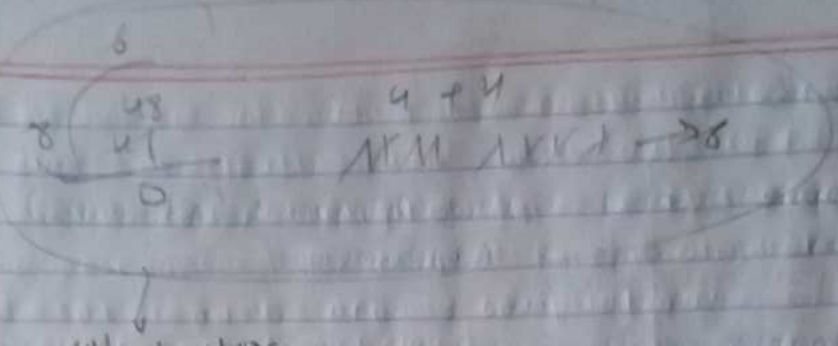
- A class send to test add method, again re-test
- one mechanism inheritance



- Inheritance is a mechanism using which we can extend functionality of class

- if class one is seloed class or is struct then we can't do inheritance

HS 256 → Cryptographic Algorithm for creating ... Signature
Hmac



Programming → to reduce line of code, we do short hands

```
int a = a>b ? a : b ;
```

Delegate

~~Anomou~~ Anonmoys lamda exp

Action, Pred, func [calculator]

1) Normal Delegate
2) multicast Delegate
3) Action
4) Func
5) Predicate

+ Anomoys method
+ lambda expression

- normal delegate is has abstraction of action, funct, pe
- anonymous method is abstraction for writing methods conventionaly
- Lambda exp is shorthand for writing Anonymous methods

3) we provided ~~method~~ parameter to method i.e extended but we haven't pass anything no need that parameter use is only for binding. [Binding parameter not taken into consideration, other all are considered]

4) An extension only one binding parameter & have at first place.

5) if extension method define with n parameter then there will be (n-1) parameter, ex Binding parameter not taken into consideration will calling.

```
main ()
{
    int i = 5;
      i.factorial X
long r = i.factorial ✓
      C.W (r);
}
```

1) sealed class
2) no source code
3) Base class library

} can use Extension methods instead inheritance

3) we provided parameter to ~~method~~ method i.e extended but we haven't pass anything no need that parameter use is only for binding. [Binding parameter not taken into consideration, other all are considered]

4) An extension only one binding parameter & have at first place.

5) if extension method define with n parameter then there will be (n-1) parameter, ex Binding parameter not taken into consideration will calling.

```
main ()
{
    int i = 5;
      i. factorial X
long r = i. factorial ✓
      c.w (r);
}
```

1) sealed class
2) No source code
3) Base class library

co
us
Ex
me
in
inhe

Extended method in class

```
Static class Statclass
{
                                    (this Program P)
  Public Static void Test3(Program P)
  {                                  ↘ class
    C.W ("m3");                      ↘ String str => class
  }
  public Static long factorial (this Int32 x)
} {
  // logic      return x* Factorial (x-1);
}
```

```
class TestExt methods
{
  main() {
  program p = new program ();

  p.Test3();
    console.ReadLine();
}
```

Note:-

1) Extension methods are static but once bound with class or struct turn into nonstatic.
2) if an extension method define with same name & methods ~~with so~~ in class, ext° method not called first preference is original method is called. same name & signature not a expected

Exception Handling
[try - catch - finally]

- try-catch-finally
- throw
- throws

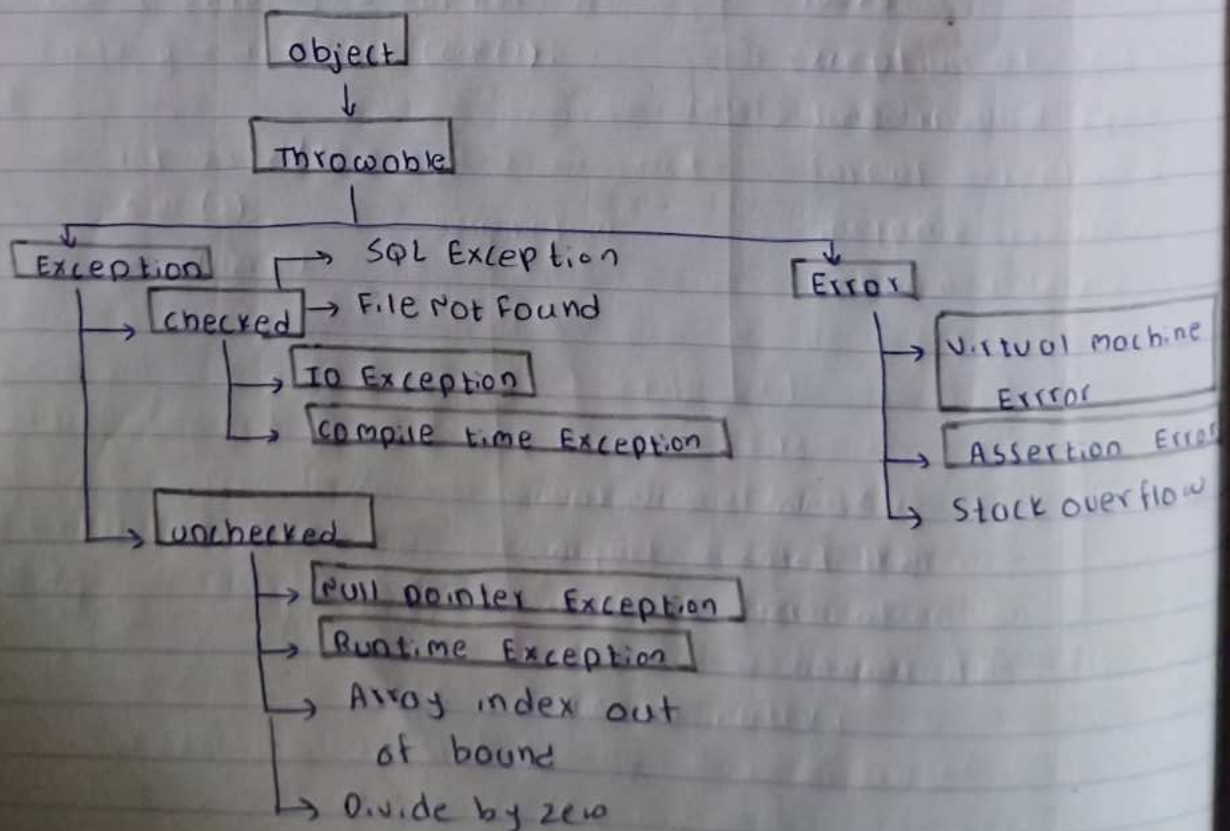
Exceptions & Exception Handling

Q What is a Exception?

→ 1) compile time error [syntax error, Array syntax wrong, ;x]

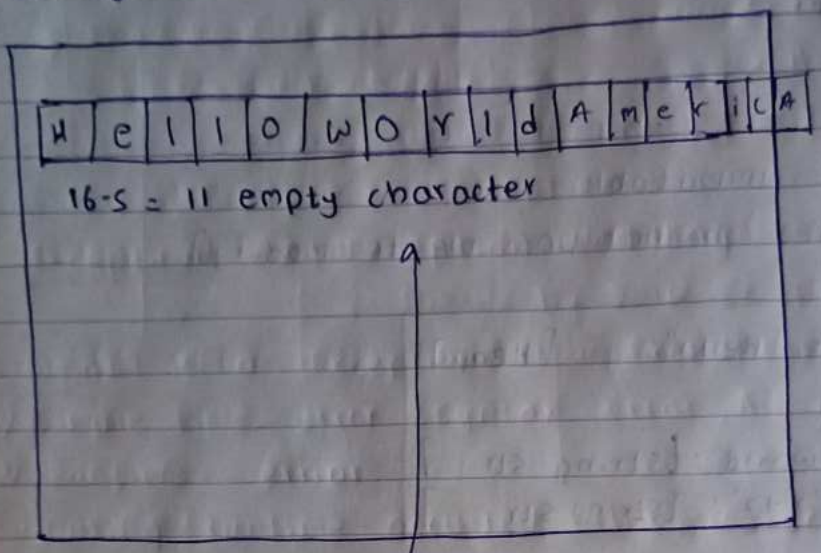
2) Runtime errror

- During runtime occurred error is called Runtime error
- wrong implementation of logic
- wrong Input Supplied
- Missing resources



Chat: Bothe B.S impact

StringBuilder sb = new StringBuilder ("Hello");



8 - 16 - 32 - 64 - 128

increase in size

sb.Append("world");
sb.Append("America");

} only one copy whenever frequent changes required.

use System.Diagnosis

→ stopwatch class

StringBuilder efficient
String not that efficient

StringBuilder sb = new StringBuilder (1000);

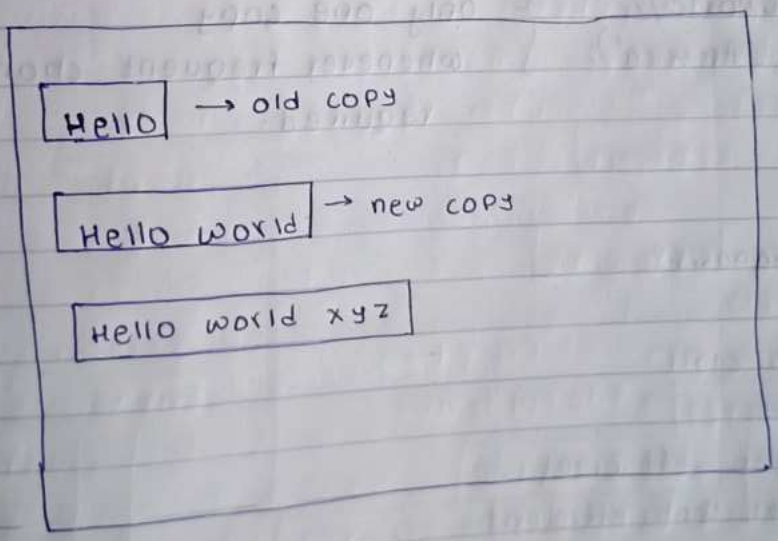
once 1000 finish double 2000.

Diff b/w Stringbuffer, String & Stringbuilder

Strings are immutable
[once declared, ~~be~~ not possible to modify]

```
String str = "Hello";    [Heap]

str = str + "world"; [String str= ]
str = str + "xyz"; [String str= ]
```

whenever trying to concat



- N modification, N copy
- 100 changes + 1 original copy lots of memory wasted
- use stringbuider when lots of modification made to string.

1) user friendly messages can be displayed.

```
try
{
 -stmts
}
catch (<Exception class Name><variable>)
{
-stmts
}
```

1) stm which doesn't require ~~error~~ execution
2) stm which couse except

1) stm which should be executed only when there is error.

```
try
{
 main()
 {
 }
}
 catch (Divide by zero Exception e1)
{
}

 catch (Formatexception ex2)
 {
 }
  catch (Exception ex3)
{
}
```

```
if ()
{
}
else if ()
{
}
else { }
```

Part-2

Exception → Application Exception (non fatal) → Programmers will do.
→ System Exception (fatal) → done by CLR

- Index out of Bound Exception
- Format Exception
- Arithmeatic Exception
  - Divide by zero Exception
  - overflow Exception

```
main()
{
    C.w("Enter 1st number:");
    int x = int.Parse(con.R());
    C.w("Enter 2nd number:");
    int y = int.Parse(C.R());
}
    int z = x/y;
    C.w(z);
    C.w("End");
}
```

x → same

100/y → exception at t
line invalid
format excepti
instance creo
abnormal ter

terminotion

100/0 ~ envokes the class & object created then abnormal termination on that line & further line not executed.

System.Exception - Base class for all Exception
System.SystemException - related to runtime Environment
System.ApplicationException - explicitly defined by applications
System.IO.IOException
System.Format Excep
System.Invalidopt Excep
System.Argument Exced
} Checked Exception.

System.NullReference
System.Index out of Range
System.Divide by zero
System.Arithematic
} Unchecked Exception.

Note :- compiler never check logic only check format (syntax)

- Runtime error causes abnormal termination.
- Runtime error very dangerous
- Exception is classes
  - Responsible

1) IndexoutofBound Excep
2) Divideby zeroExcep
3) overflow Exception
4) Formot Exception
} Predefined classes Available for Runtime Except

Exception → logic for abnormal termination
- Readonly to display error message which is declared are virtual
- "message"

collections & generics

Types of collection

- Generic
- Non-Generic

collections
→ Dynamic Array

Array.Resize → Destroy old & create new

```
main()
{
  int [] arr = new int [10];
  Array.Resize(ref arr, 15);

}
```

- Resize [Increase in size]
- Insert into middle of array.
- never delete into middle of array.

- can Resize
- can insert in middle
- can delete from middle

while declaring exception declare

- Grand child
- child
- father
- Grand father
- fore father
- Ancestor

finally { c.w ("done"); }
↳ mandatory execution

```
try
{ }
catch()
{ }
finally
{ }
c.w ("");
```

```
try
{ }
catch()
{ }
finally
{ }
```

1) try - catch
2) try - catch - finally
3) try - finally
↳ for sure abnormal termination then also finally block occurred.

Signatur

HS 256 → cryptographic Algorithum for creating digital
HMAC

Part-3

- if all statement in try executed then last block of catch executed
- Abnormal termination stops & goes into require catch Block ↑ this stops.
- Last statement executes if any catch block is not execute

```
catch (Exception ex)
{
}
```

Handles all exception but we cannot use only this catch block, because it causes overflow.

↳ if this is not there Abnormal termination occurs ..

finally { } → Always executed where catch block executed or not.

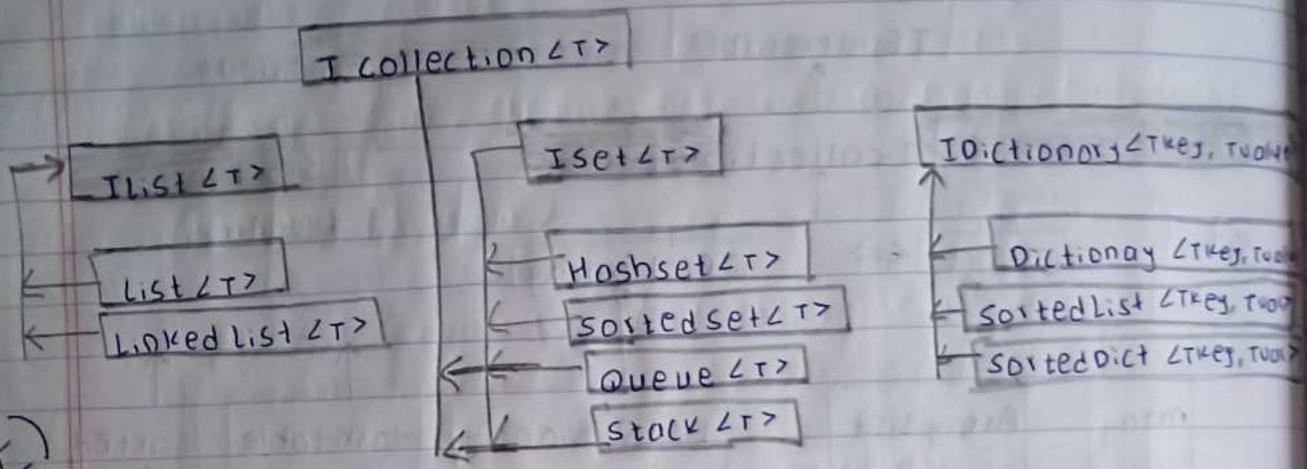
```
try
{ }
exte
catch (Exception ex)
{ C.w (ex.message); }
finally
{ C.w ("finally exec"); }
```

→ open a file HD
write into the file
close file x
→ close file x
→ close file

- control entering in try, finally 100% executes

Algorithm for creating digital

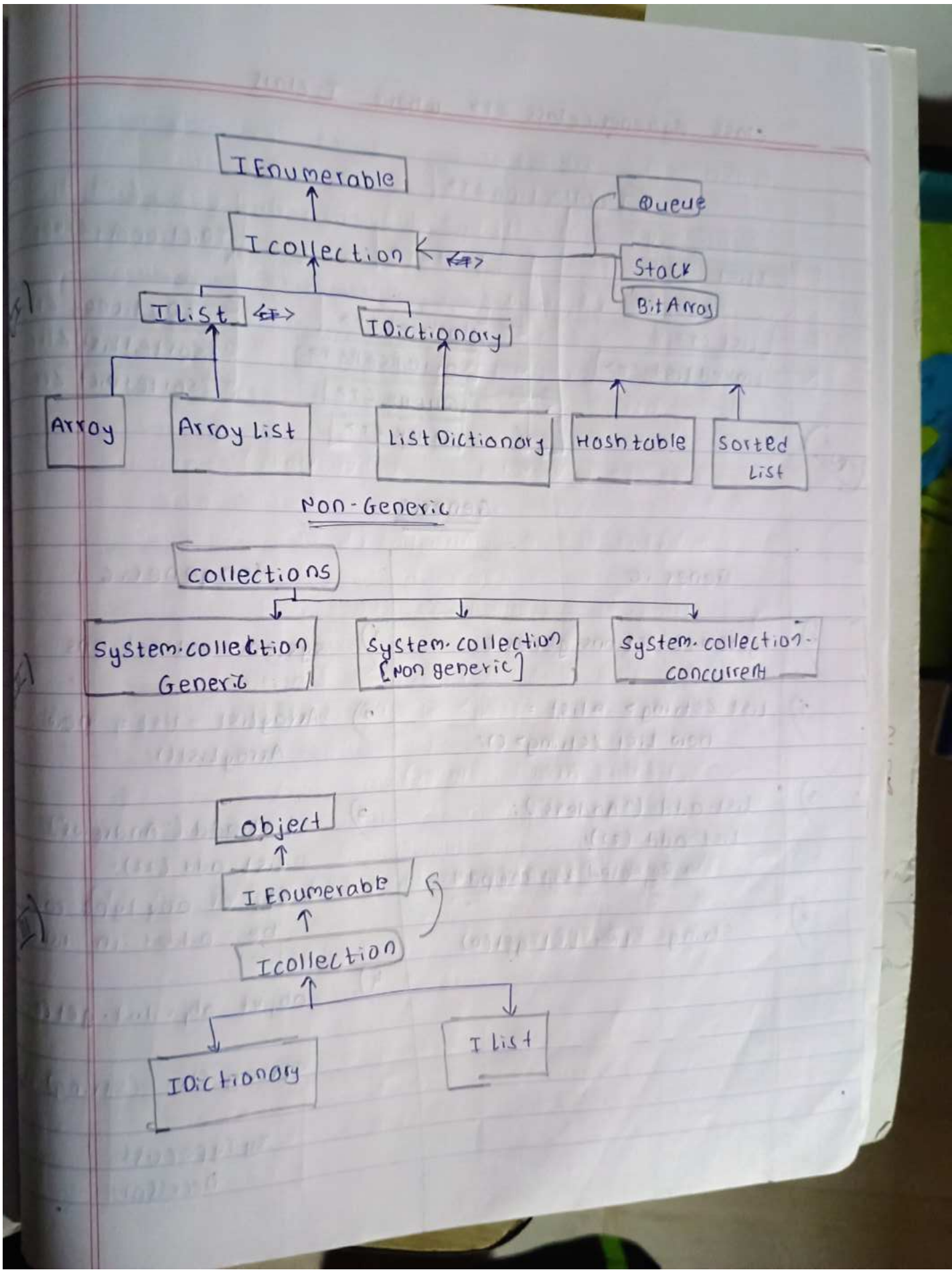
Class MyGenericclass <T> where T: class



IV)

Generic

	Generic	Nongeneric
1)	System.collections.generic	System.collections
2)	List <string> mlist = new List <string> ();	ArrayList list = new ArrayList();
3)	List.add ("Anders"); List.add (22); → Rejected by compiler	List.add ("Andrews"); List.add (22); obj of any type can be added in list
4)	String str = List.get(0)	object obj = list.get(0); String str = (string) obj; By Typecast necessary



Non-Generic collections

- Stack
- Queue
- Linkedlist
- Sortedlist
- Arraylist
- Hashtable

} collection classes [Non generic]

using System.Collections;

	Array	Arraylist [collection]
1)	Fixed length	1) variable length
2)	Not possible to insert	2) possible insert any position
3)	Not possible delete	3) Delete from middle

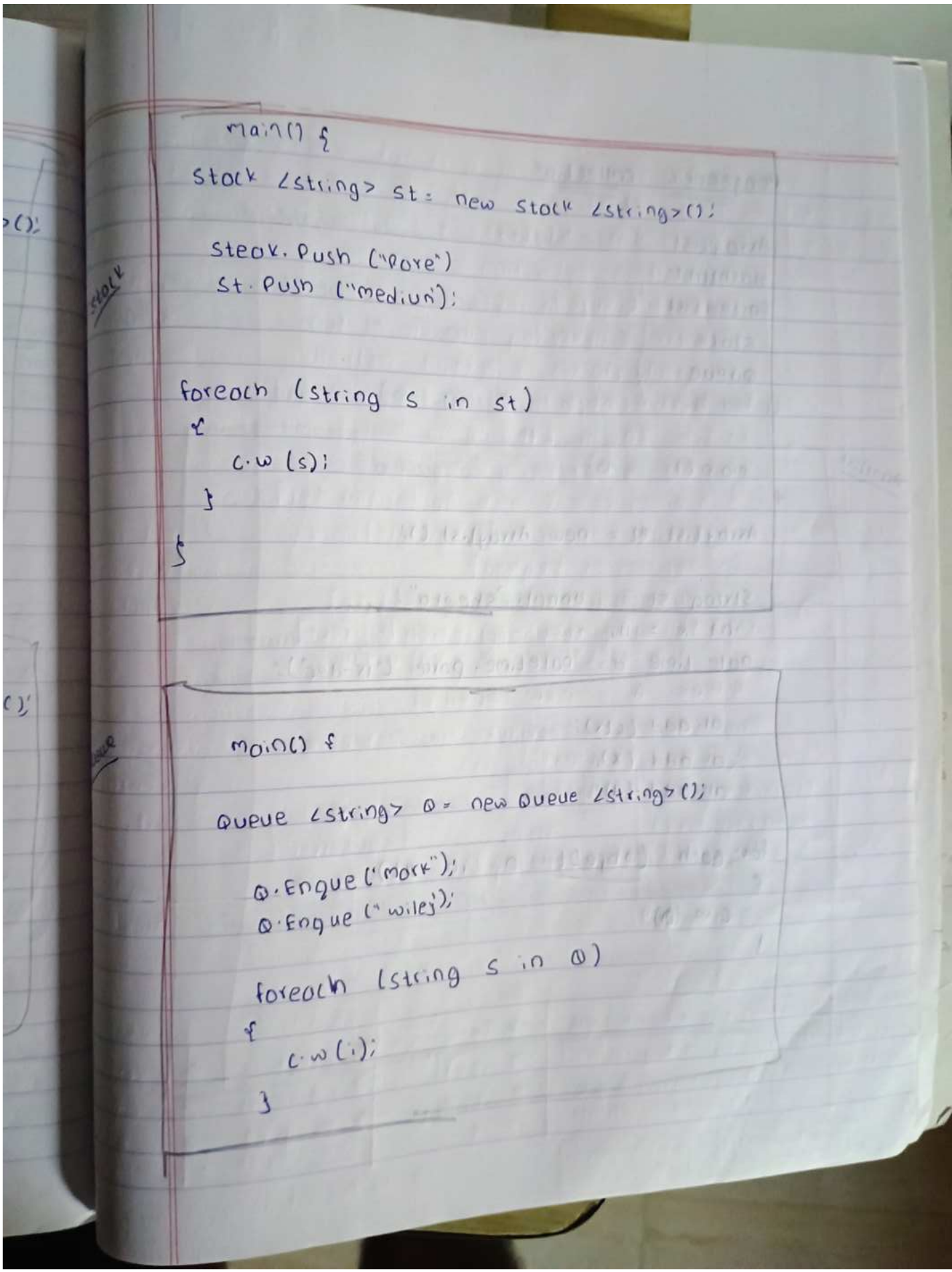
al.insert(3, 250);
(value, object)

```
main()
{
    Arraylist al = new Arraylist(); // (10);
    al.Add(100);
    Console.Readline();
    c.w(al.Capacity);
}
```

100 → 200 → 400 → 800
10 → 20 → 40 → 80
0 → 4 → 8 → 16

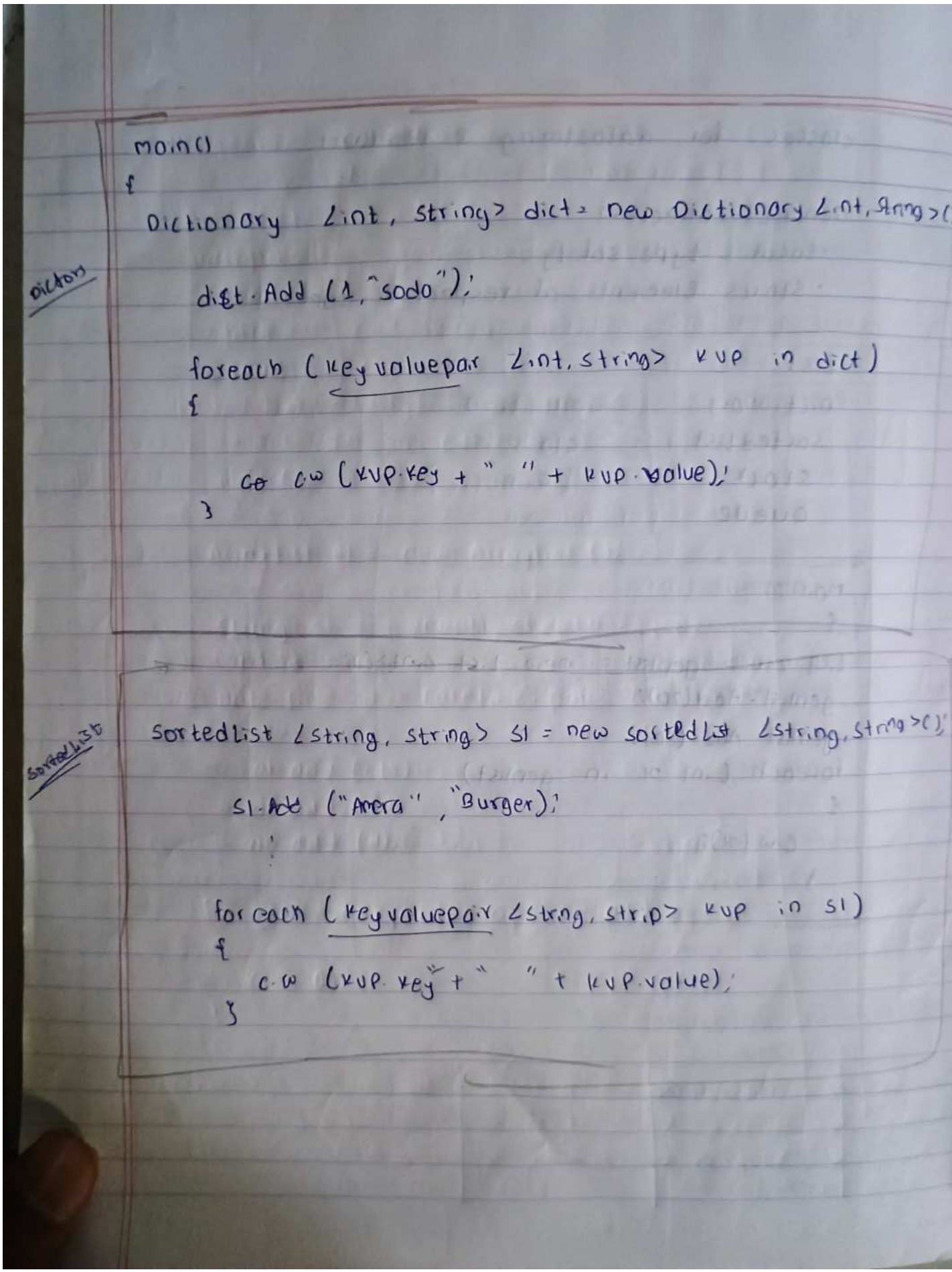
4 → 8 → 16 → 32 → 64

al.Remove(200);
al.RemoveAt(1);



```
main() {
Stack <string> st = new Stack <string>();

Steak.Push ("Pore")
St.Push ("mediun");


foreach (string s in st)
{
   c.w (s);
}
}
```

```
main() {

Queue <string> Q = new Queue <string>();

Q.Enque ("mork");
Q.Enque ("wiley");

foreach (string s in Q)
{
   c.w (i);
}
```



Dictionary

```
main()
{
 Dictionary <int, string> dict = new Dictionary <int, string>()

    dict.Add (1, "sodo");

    foreach (keyvaluepair <int, string> kvp in dict)
    {

       cw (kvp.key + " " + kvp.value);
    }
```

SortedList

```
sortedList <string, string> sl = new sortedList <string, string>();

    sl.Add ("Amera", "Burger");

    for each (keyvaluepair <string, str.p> kvp in sl)
    {
       c.w (kvp.key + " " + kvp.value);
    }
```

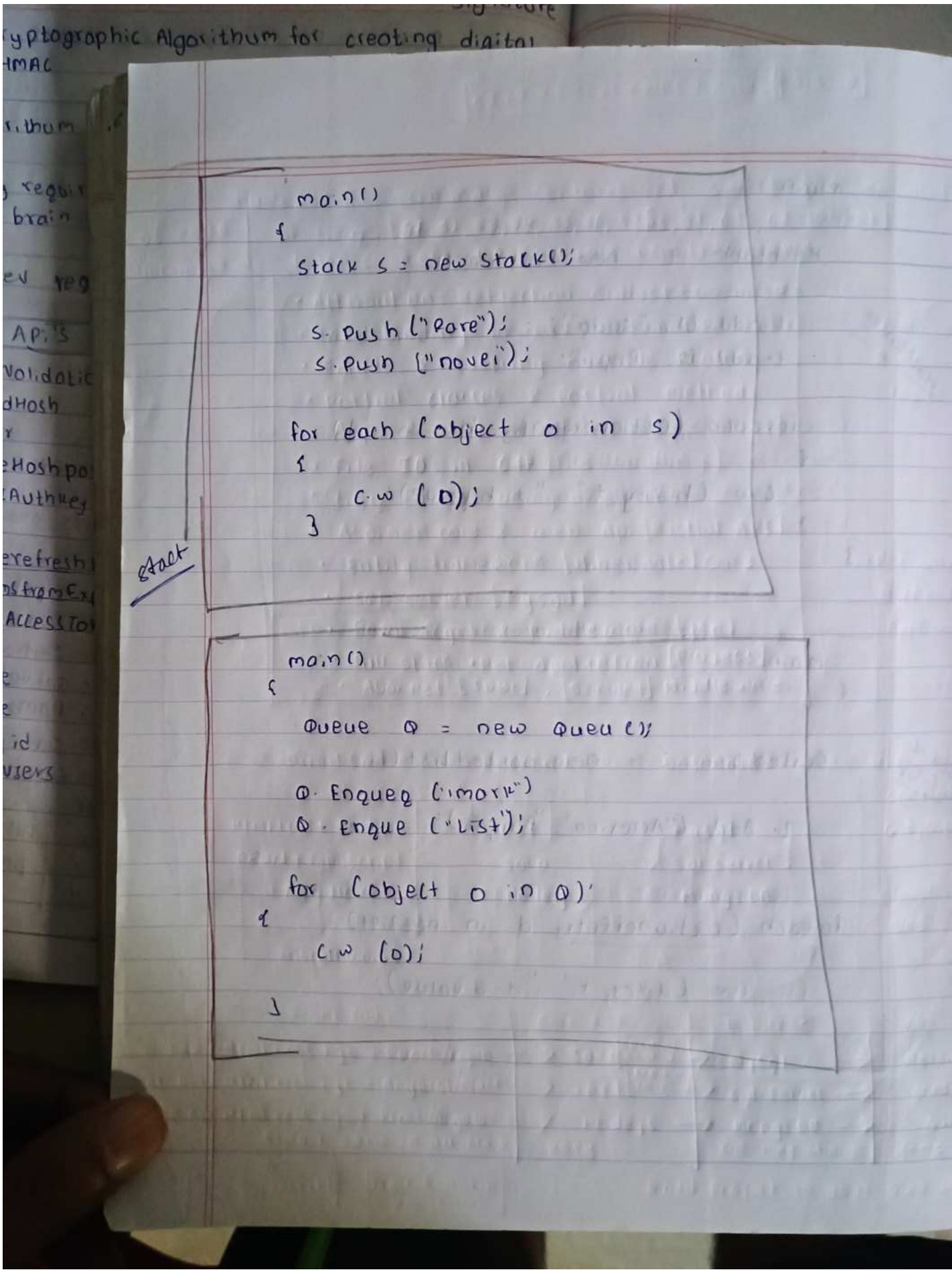
classes for datastorage & retrieval

Generic collection
- Ensures type safety
- Stores Elements of respective type

List
Dictionary
Sorted List
Stack
Queue

} collection classes [Generic]

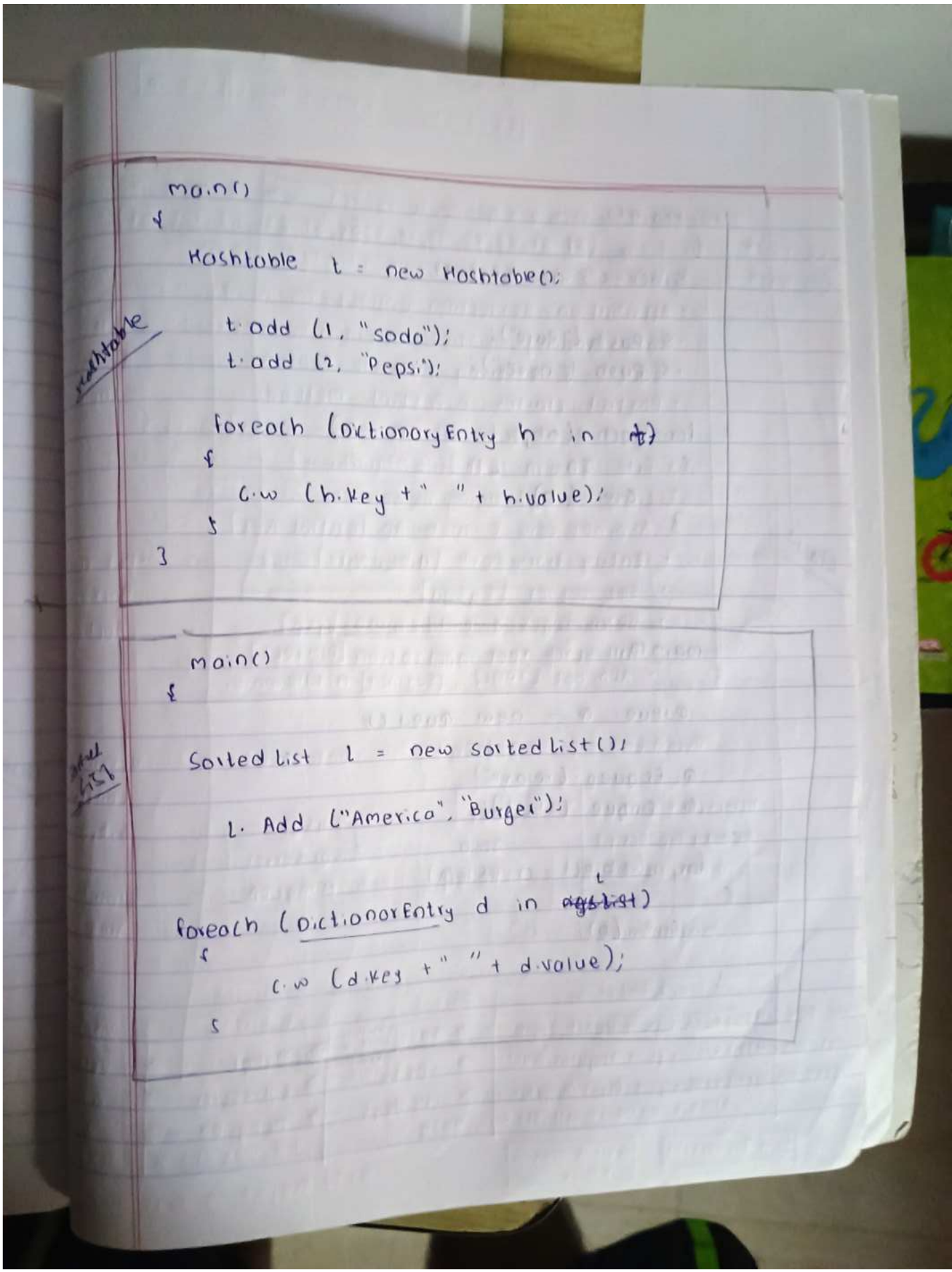
```
main()
{
List <int> genlist = new List <int>();
genlist.Add(20);
genlist.Add(30);
foreach (int x in genlist)
{
   c.w(x);
}
```



```
main()
{
  Stack S = new Stack();

  S.Push("Pare");
  S.Push("novei");

  for each (object o in S)
  {
     C.w (o);
  }
```

Stack

```
main()
{
  Queue Q = new Queu();

  Q.EnqueQ("imorK")
  Q.Enque("List");

  for (object o in Q)
{
   C.w (o);

}
```



```
main()
{
  Hashtable t = new Hashtable();
    t.add (1, "sodo");
    t.add (2, "Pepsi");

  foreach (DictionoryEntry h in t)
  {
    C.w (h.key + " " + h.value);
  }
}
```

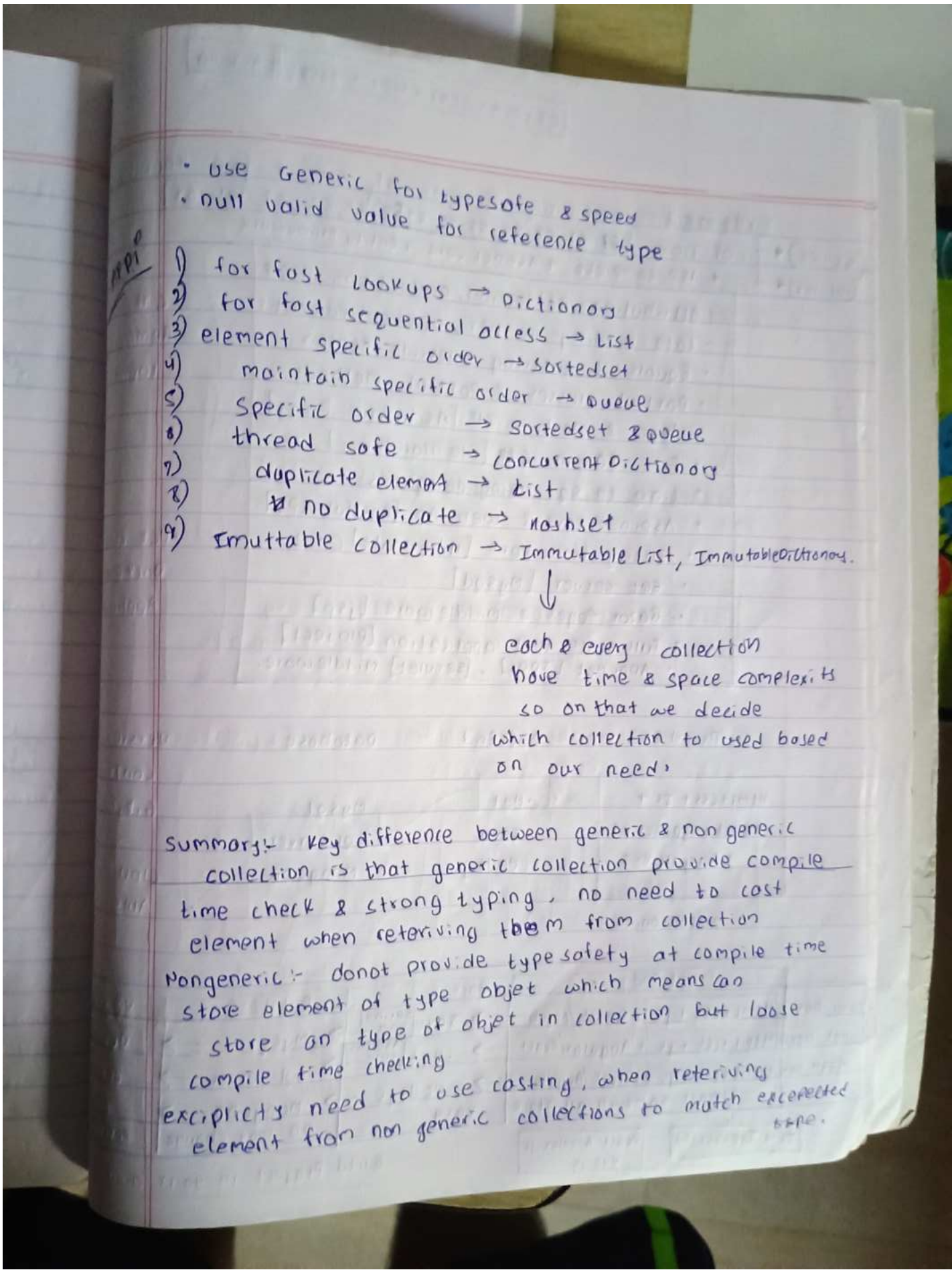
```
main()
{
  Sorted list l = new sorted list();
    l. Add ("America", "Burger");

foreach (DictionorEntry d in l)
  {
      C.w (d.key + " " + d.value);
  }
```

Nongeneric collection

- ArrayList
- Hashtable
- Sorted List
- Stack
- Queue
- Linked List

ArrayList

```
main()
{
  ArrayList al = new ArrayList();

  String st = "unnati, shweta";
   int x = 11;
  Date time d = Datetime.parse("18-dec");

    al.add(str);
    al.add(x);
    al.add(d);

  for each (object o in al)
  {
    C.w(o);
  }
```



- use Generic for typesafe & speed
- null valid value for reference type

Topic

1) for fast lookups → Dictionary
2) for fast sequential access → List
3) element specific order → Sortedset
4) maintain specific order → Queue
5) Specific order → Sortedset & Queue
6) thread safe → Concurrent Dictionary
7) duplicate element → List
8) no duplicate → Hashset
9) Imuttable collection → Immutable List, ImmutableDictionary.

↓

each & every collection have time & space complexity so on that we decide which collection to used based on our need.

Summary:- key difference between generic & non generic collection is that generic collection provide compile time check & strong typing, no need to cast element when retriving them from collection

Nongeneric:- donot provide type safety at compile time store element of type objet which means can store on type of objet in collection but loose compile time checking

explicity need to use casting, when retriving element from non generic collections to match excepected type.

Kudvenkat remaining lectures of C# & goin know
of Linq [51, 52, 53, 54, 55, 56, 57, 58, 59, 61, 62, 63 101, 102]
+ ASP complete + Asp.net core + Entity Framework

- Attribution, Reflection, collections in C#
- Late binding in .NET, optional Parameter
- Equals & tostring override Reason
- Partial classes & partial methods
- Asyn & wait in C# [Aschronys Progg]
- File IO in C# [File Handling]
- Linq to sql using delegates

- Asp.net core & web Api [Restful API]
- Entity fromework [Hibernate] core
- SQL server [Mysql]
- Razor Pages & middleware [jsp]
- Full stack .Net application [Project]
- ADO.Net [JDBC], [Servlet] middleware.