```
package test;
public Class ABC
{
    public static void main (String [] args)
    {
        executable
        System.out.println ("Hello world");
    }
}
```

return type

name

parameter

Scanned with CamScanner

Data type.

Variable

Arrays.

Value vs Ref Type.

Data type
① Premitive Datatype
② Non-Premitive Datatype } Types

Premitive means very basic.
Premitive store only one number at a time.

Non-Premitive are called Data-Structure.
- objects
- Arrays
- structure
- String

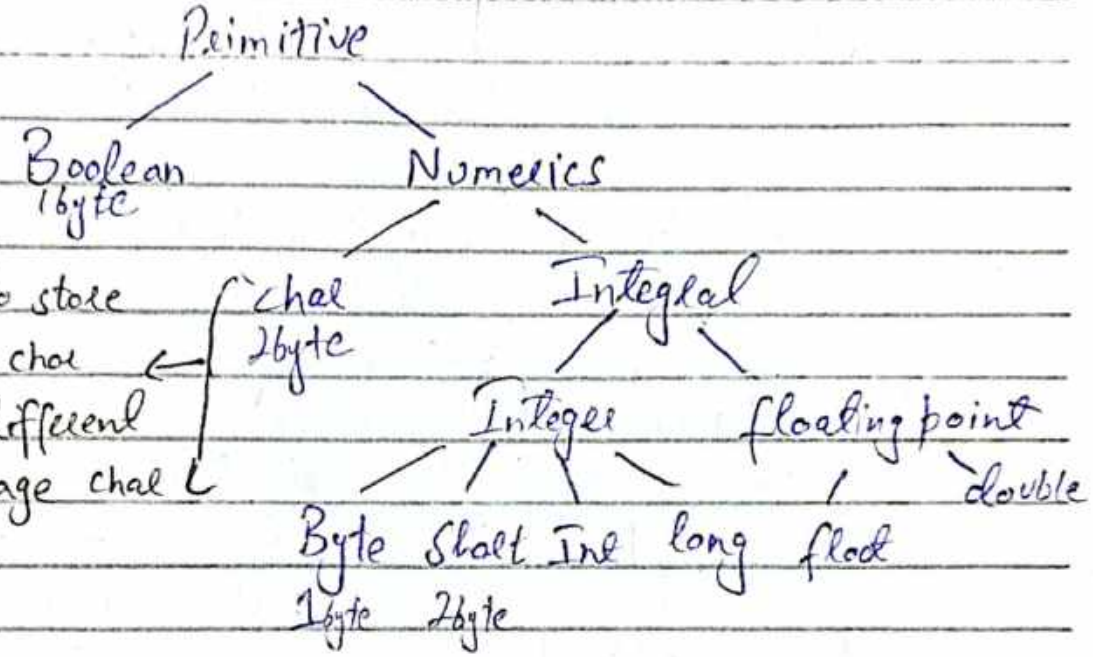
Non-Premitive store more than one number at a time.

Scanned with CamScanner

(Total 8 data type)

Primitive Type.

(1) Boolean } Types
(2) Numerics. }



→ There are 255 characters and every has its Specific ASCII value.
→ Some are visible char and some are invisible char. (Alt + Ascii)

Integral → Something that represent a number.

Scanned with CamScanner

To store data of 1 bits we allocate 1 byte for it.

To store data in one byte we have 8 bits, so one bits is for sign (+) or (-) and 7 is for store number

Formula for range of number

$$\left(-2^{n-1}, 2^{n-1}-1\right)$$

① Byte → 8 bit.

$$\left(-2^{8-1}, 2^{8-1}-1\right)$$

$$(-128, 127)$$

If we add 1 and store 128 in byte its change to -128 and 129 change to -127,

$\Rightarrow 127+256 = 127$

$\Rightarrow 127+257 = -128$

Scanned with CamScanner

(2) Short → 16 bit

$(-2^{15}, 2^{15}-1)$

$(-32768, 32767)$

8×8

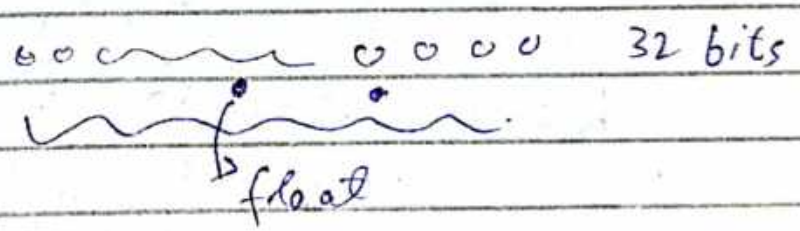
2^{10}
2^{6}
2^{16}

(3) Int → 32 bits.

(4) Long → 64 bits.

تيرنا

(5) float. → (.) float b/w the bits according to the requirement.



Scanned with CamScanner

Valiables :-

Scope of variable tell us in which field our variable use

C long Concept:

```
#
void abc();
int main()
{
    int a; -> local valiable

}
void abc ();
{
  printf ("%d", a);
}
```

can't use here.

```
#
void abc ();
int a; -> Global valiable.
int main()
{
  a = 10; -> use here
}
void abc()
{
  printf ("%d", a);
}
```

use here

Scanned with CamScanner

Java lang Concept

Local variable -> A variable that is declare inside a block is called local variable

instance variable -> A variable that is declare inside a class but outside method.
↳ Object are also called instance
Instance means, A thing that exist more than one.

Class variable, Static variable:-

A variable that is declare inside a block is called static variable.
with a keyword static

Scanned with CamScanner

new → need memory, memory assign

Arrays.

Declaration of array

① int [] arr
② int arr []
③ int [] arr

Memory allocation

arr = new int[5]

Other method of declaration

int [] arr = new int[5];

intialize :-

int [] arr = {10, 20, 30, 40, 50};

2D array - array of array

int [][] arr = new int [5][4];

It is called jagged array
↳ Actualy its not 2D array. only C# exit 2D array There is no 2D array in Java

Scanned with CamScanner

Memory allocation

```
arr[0] = new int[4]
arr[1] = new int[4]
   |
   |
   |
arr[4] = new int[4]
```

	0	1	2
0	00	01	02
1	10	11	12
2	20	21	22

```
arr[0][0] = 12
arr[0][1] = 24.
```

2	4	6	9	X
11	18	3	x	x
6	2	7	8	1

→ no element exit here error if we use this

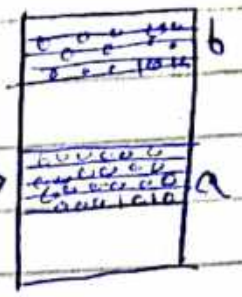
intilization of 2D array

```
int [][] arr = { {10, 20, 30},
                 {40, 50, 60},
                 {70, 80, 90} }
```

Scanned with CamScanner

Value vs Ref:-

```
int a;
a=10;
```



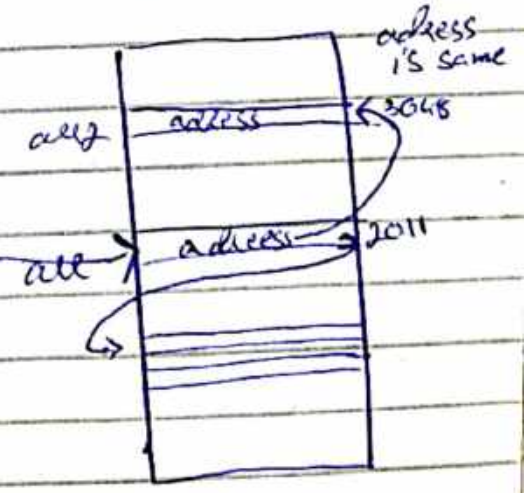
a store binary digit of 10 in ram.

$(10)_{10} = (1010)$

1010 represent in 32 bits is
00000000000 1010

start 0 dont effect the value of 1010.

```
int [] arr = {1, 2, 3};
```




```
int b=a;
```

$\boxed{\text{Value}}$ store in adress of a is store in b.

[redacted]

int [] arr2 = arr.

ref address of arr (2011) is stored in arr2.

If we change element of arr then the same index value of arr2 also change.

Primitive data type are value type
Non-primitive data are reference type

Structures in C are value type
Class in Java are ref type

int arr = new int [3]
int arr2 = new int [5]

arr2 = arr.

now index of arr2 is 3 and arr is also 3.

C#

Structure is value type
Class is reference type

Scanned with CamScanner

int a=10;
Static assigment

When user enter data and its assign to a then its dynamic assigment.

Scanned with CamScanner

Data type

Convert

Type Casting :-

Cast : One type change to another type.
↳ mould (ڈھالنا)

Type Casting Convert one ~~data type~~ data type variable
to another data type variable

byte — short — Int — long — float — double

Automatic conversion / Widening

Narrow

Manual conversion / Narrowing

wide

1 2 4 8 4 8

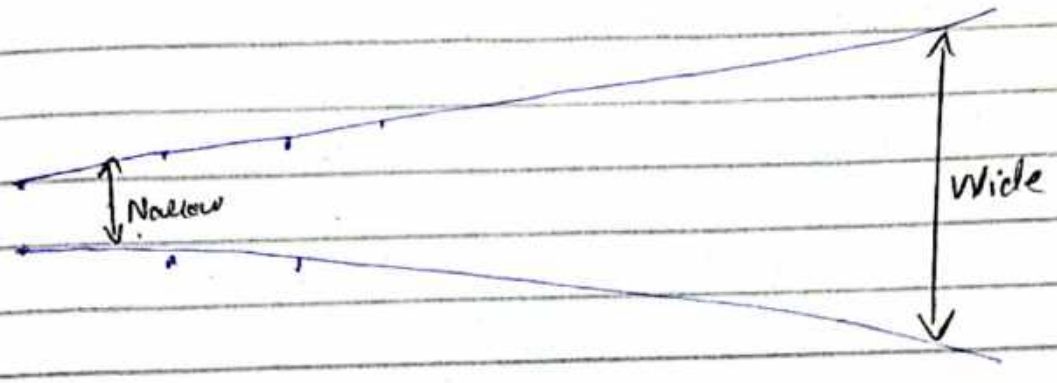
at

C

Scanned with CamScanner

byte – Short – Int – long – float – double

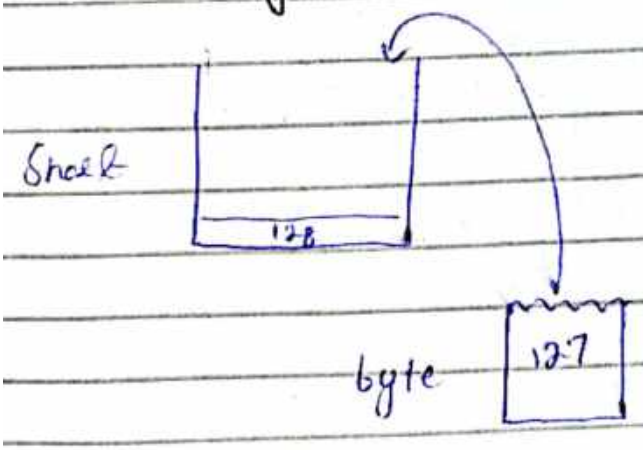
implicit / Automatic Conversion / Widening →



← explicit / Manual conversion / Narrowing.

1 2 4 8 4 8

Widening:-



Widening.
Automatic conversion
implicit.

11.3 — 11
float — long

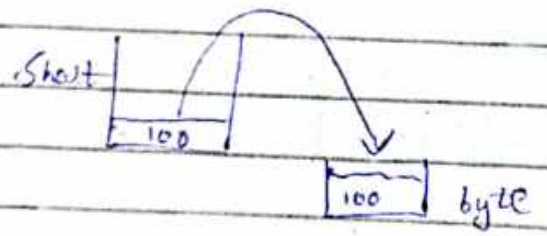
There is loss of 0.3 is conversion.

Program:-

```
byte b = 127
int i = b;      (implicitly)
long l = i;
double d = l;
```

Narrowing:-

There may be loss or may not be loss in Narrowing process



Scanned with CamScanner

Program.

int i = 10;

short s = i; Error. (Data may be loss so mention explicitly)

short s = (short) i;

int i = 74321;
short s = (short) i; data loss.
short range (-32768, 32767)

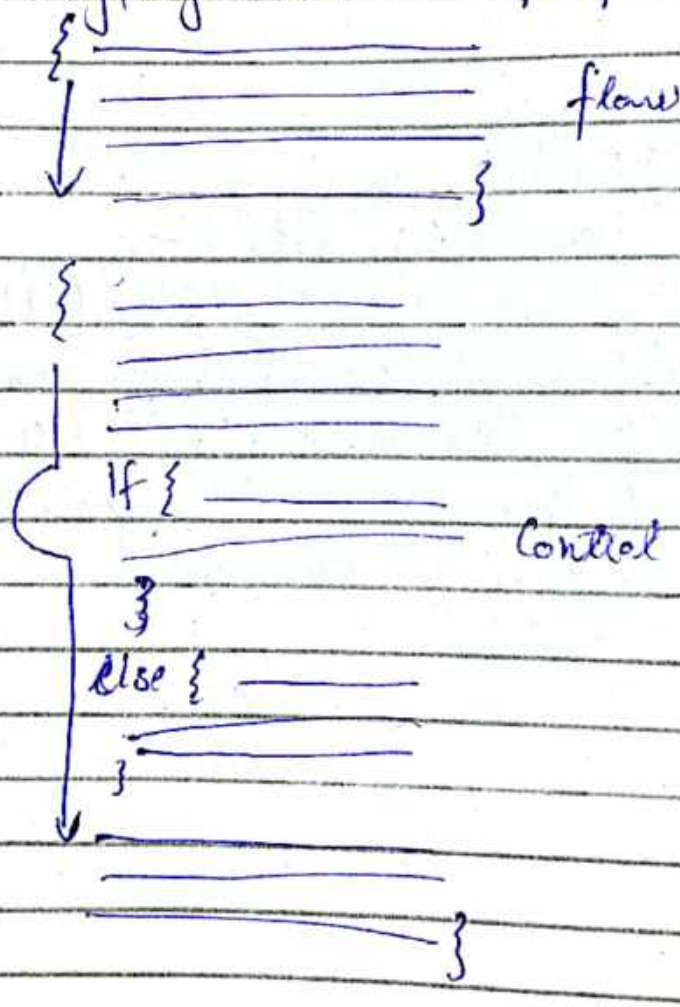
Data in i and s are not same.
after 32767, 32768 is act and 0 and
32769 is 1 so, clock run and number
reach at 74321 is store in s.

Scanned with CamScanner

Control statement / structure:-

Control: Anything that is in ~~use~~ control: you can use them according to your use.

flow:- Anything move in specific direction



Scanned with CamScanner

Control Statement

① Selection / Decision
  ① Simple if
  ② if-else
  ③ (if, else if, else) ladder
  ④ Nested if
  ⑤ Switch

② Iterative / loop / repetive

  ① for
  ② while
  ③ do while

③ Jump statement

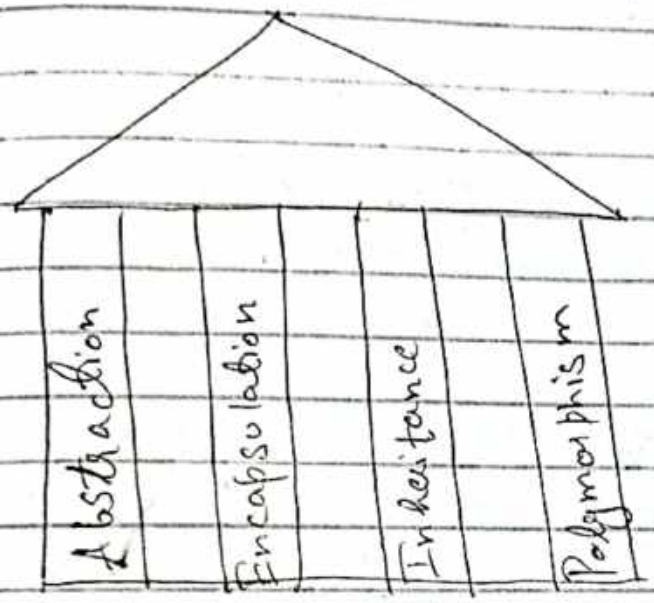
  ① break
  ② continue
  ③ return

goto is not usable in java. (avoid the use of goto)
slow out cpu.

Scanned with CamScanner

Basic Principle of OOP

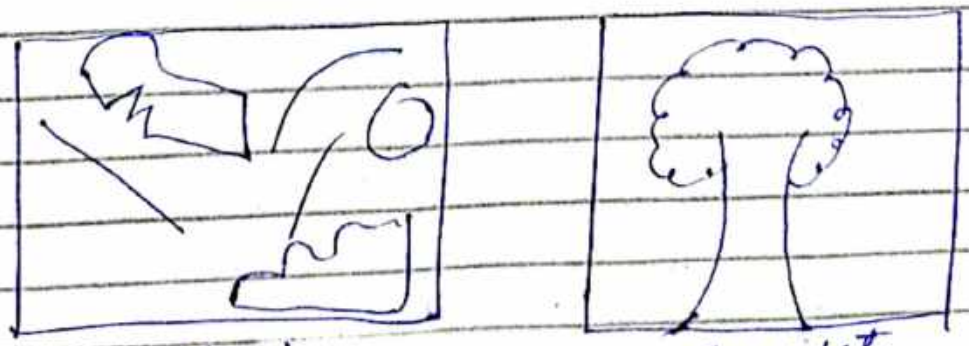
→ four basic principle of oop
↳ These are basic pilar of oop
↳ ستون



① Abstraction:-

Eng means :- The process of removing unnessary info in order to reduced the data and manage needed info

Abstract art , Concrete art



abstract Concrete

Scanned with CamScanner

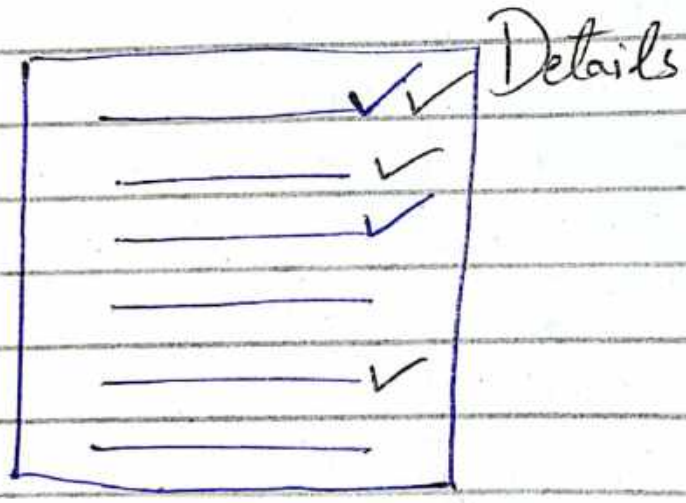
Teacher

Doc

Exam

detail of student

detail of student to every person teacher, doc, Exam is differ according to their need.



• neccassary for doc • neccasary for teache

~~neccasary for~~

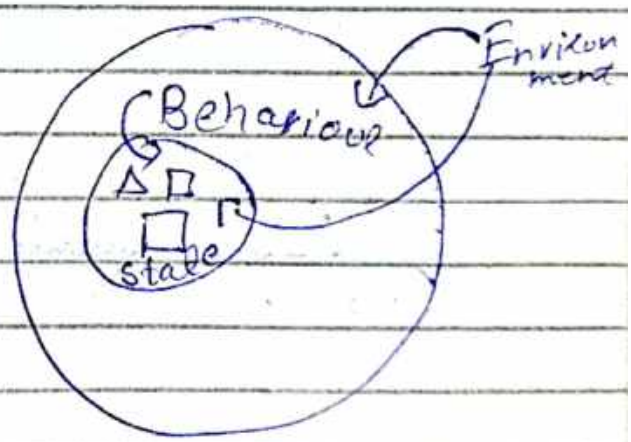
Scanned with CamScanner

(2) Encapsulation.

En capsula ation
In capsule

capsule : In order to protect medicine from environment and protect you (harmful fat) → from that medicine. we capsule (you) that medicine
Time capsule

Capsule: To protect any thing

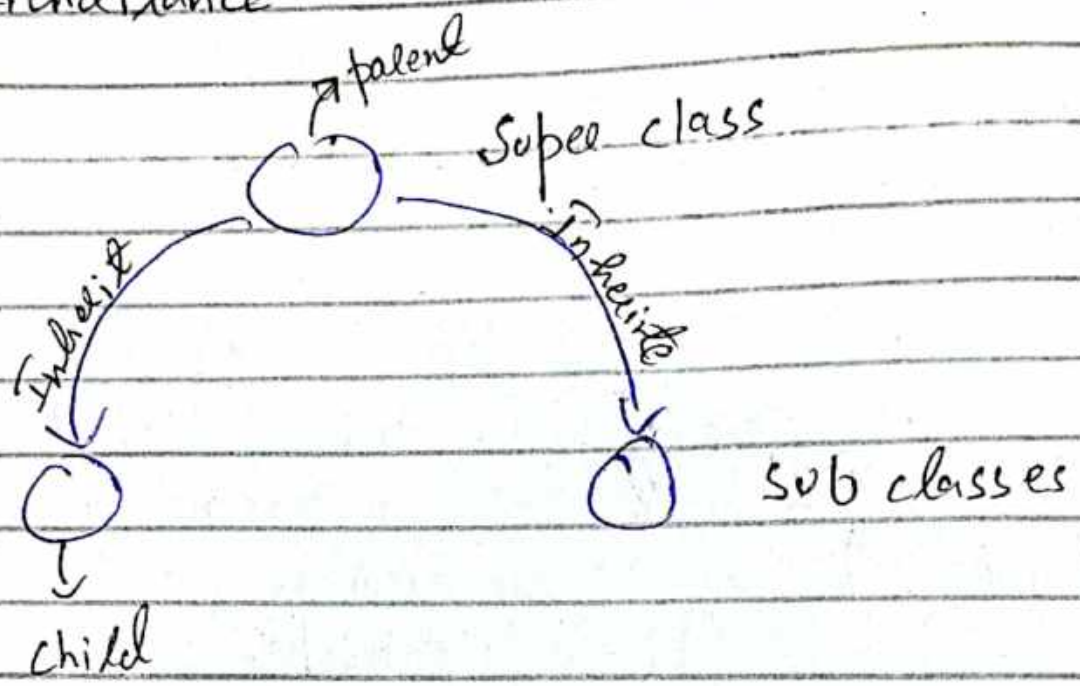


→ State are protected and control by Behaviour
→ If environment change the state then it is not protected

Eng mean: The process of data hiding is encapsulation

Scanned with CamScanner

(3) Inheritance



Eng mean:- The process of something to take the properties of something else

Scanned with CamScanner

(4) Poly morphism.

Poly morph ism

multi, many diff type,
more than on diff shape.

```
int add (int , int)
int add (int , int , int)
int add (int , int , int , int)
float add (float , float)
```

→ multiple form of add method.

Eng def. The properties of somthing to take multiple shape or multiple form

Scanned with CamScanner

Code terminate of single line by ;
Code terminate of multiple line by
{ ——————
——————
——————
——————
}

package test;
public Class ABC
{
public static void main (String [] args)
{
executable
System. out. println ("Hello world");
}
}

return type
name
parameter

Void $\xrightarrow{\text{Means}}$ ~~No~~ Nothing, empty, no return type

Englis $\Rightarrow$ space ~~void~~ b/w two galaxy is void

$\Rightarrow$ Void has return type but its return empty

Scanned with CamScanner

( ) palantesis

{ } braces

[ ] Bracket

=> There is no goto statement in java

Scanned with CamScanner

Encapsulation — Nested class
— Access Modifier

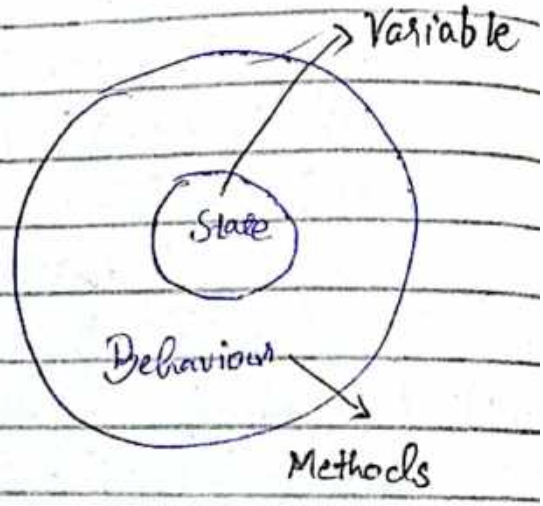
Polymorphism — overloading — constructor
— Methods
— overriding

Scanned with CamScanner

Intiduction to classes and object.

⇒ Variable & Method.
⇒ Declaring & creating Objects

Object are also called instance.



Structure in C:-

```
struct Person
{
    char name[50];
    char fname[50];
    int age.
}(p1, p2, p3);
```

if we dont declare this then put (;) here only.

```
Struct Person
{
};
```

Scanned with CamScanner

```
Struct Person
{
    char name[50];
    char fname[50];
    int age
    void getdata (char n[], char f[], int a)
    {
        name = n;
        fname = f;
        age = a;
    }
};
```

C can't do that
can't declare function
in structure

⇒ like method in
java.

Scanned with CamScanner

Class in Java:

```
class Person
{
    String name;
    String fname;
    int age;

    ~~void getData(~~
    void SetName (String S)
    {
        name = S;
    }
    void setFname (String S) {
        fname = S; }
    void setAge (int a) {
        age = a; }
    void setData(String n, String f, int x) {
        name = n;
        fname = f;
        age = x; }
    String getName () {
        return name; }
```

Scanned with CamScanner

Getter and Setter :-

Set → assign value in variable.
Get → return value.

```
String getFname () {
    return fname; }

int getAge () {
    return age; }

}
public class Test {
  public static void main (String[] args) {
    int a;
    Person p;

    p = new Person () ;
```

→ Constructor

assign memory

Make memory of the size of Person class

```
/* 1- declaring a ref          Person p;
   2- assigning memory         new Person ();
   3= Assigning to a ref.      p= new Person();
   */

Person p1= new Person () ;  → 3 in 1 line.
Personn p2 = new Person (); Second object.
```

Name, fname, age

Name, fname, age

Scanned with CamScanner

```
/*  p1                  |  p2
Name = Umar             |  Name = Ali
fName = Nawaz           |  fName = Khan
Age = 19                |  Age = 20   */
```

```
p1.setName("Ali");
p1.setFname("Khan");
p1.setAge(20);

System.out.print("FName of Person1 = " +
                 p1.getFname());

Person p3 = new Person();
p3.setData("Umer", "Nawaz", 10);

System.out.println("Data Stored in the object");
System.out.println("Name = " + p3.getName());
System.out.println("Fname = " + p3.getFname());
System.out.println("Age = " + p3.getAge());
```

↓ Class ↓ object ↓ method

```
    }
}
```

Scanned with CamScanner

Naming Convention.

1. Pascal
2. Camel
3. Snake.

Class always start with Capital letter.
Variable always stat with small letter.
method always with small letter + ()

Class

MyClass, ThisIsAClass.

Variable + methods

userID, myVariable,
userID(), myMethod()

Camel casing:-

Class — Person
Variable — person
method — person()

~~Two letters word and more than two left~~

~~Class~~

Scanned with CamScanner

Two or more than two words.

Class — MyClassName
Variable — myVariableName
method — myMethodName()

Scanned with CamScanner

lecture # 7

① Input using Scanner:-

② Access Specifier / Access Modifier.

Input using Scanner.

java.lang.* ⟶ implicitly import
↳ lang means language.

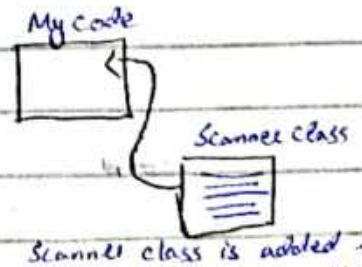
Class:

System.in.out.println( )
↳ class ↳ Object ↳ Method

scanf
↳ scanner is an input device
↳ for taking input.

⇒ Scanner class is not in the library of lang
They are in util class. java.util.*

Import of Scanner class



Scanner class is added to my code during compilation

import java.util.Scanner;

⇒ import the class after the package name.

↳ package mydemo;
import java.util.Scanner;

Scanned with CamScanner

```
package mydemo;
import java.util.Scanner;

public class MyDemo {
  public static void main (String [] args) {

    Scanner in= new Scanner (System.in);
    int x;
    x= in.nextInt();
    double y;
    y= in.nextDouble();
    char z;
    z= in.nextLine().charAt(0);
```

Input Mismatch Exceptio
↳ ~~Between~~
Data are of int type and you call other type of method like nextByte().

Scanned with CamScanner

Method of Scanner class.

1. next()
2. next line()
3. next Byte()
4. next Boolean() → true, false (small letters)
5. next Int()
6. next Short()
7. next long()
8. next Float()
9. next Double()

String.

next() : Use to scan string without space.
Use to scan only one word
they break scan on void space.

next line() : Scan the whole line. Multiple word.
they break on enter key.

Char

How to scan a char. ?
↳ using the ~~of~~ next(), ~~[illegible]~~ nextLine() method
use for String. and pick a char from
sum index.

• charAt() pick a char from specific index.
↳ method of String data type

Scanned with CamScanner

① next().charAt(0)
② nextLine().charAt(0).
↳ Use to scan character.

Scanned with CamScanner

Access Specifier / Access Modifier.

Specify: (بتانا) to clear, (واضح کرنا)

Modify: (تبدیل کرنا) to change.

Access: to reach, (حاصل کرنا)

Access Specify.

↳ Specify that who access this thing.

⇒ Who access our variable, method and class

Access Modify: (only in Java)

int x; change (خود بخود (default) کر دیتا ہے)

default ↳ public int x;

(change the specifier from default to public.)

Access level

① private
② default
③ protected
④ public.

Restriction level never reversable.

One modifier can change to next level modifier, But not upper modifier

Scanned with CamScanner

Class.

we use Three modifier with class

① private
② default
③ public.

[2 modifier and 1 in special case.
private (special case)]

We can't create a private class.

⇒ But we create nested class then that nested class is private. (special case)

↳ nested class use three access.
↳ non-nested class use two access

Methods / Variable.

① private
② default
③ protected
④ public.

⇒ Access modifier is also encapsulation. because we protect our code by different level

Scanned with CamScanner

Access level / Modifier	You	Your family	Your relative	Unknow person.
	Within Class	With in package	With in other package	Everywhere else
private	✓	✗	✗	✗
default (no keyword)	✓	✓	✗	✗
protected	✓	✓	✓*	✗
public	✓	✓	✓	✓

private:

private box → only for you

default:

Kitchen → you and your family not your relative and unknown person.

protected :-

Dining room → you and your family and (relative if you allow them) not for unknown person.

⇒ if the protected class is inherited then they access by other package

public :-

⇒ outside ~~package~~ project we use something of public by import them.

public park: Any one can use park allow to access the park.

* only if inherited

Scanned with CamScanner

Constructor:-

- Constructor with parameter.

— When we can't create constructor the java will create a default constructor

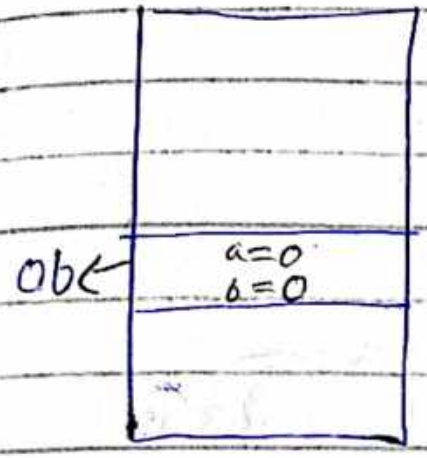
(1)
```
class Abc
{
    int a;
    int b;
    public Abc()
    { a=0;        default constructor
      b=0;
    }
}
```

Type of Constructor.

(1) Default Constructor | No argumented constructor.
(2) parameterize Constructor.
- passing object in constructor
- object as a parameter

(3) copy Constructor. (subset of (2))

Scanned with CamScanner



~~class name~~

Classname objname= new Constructor;

Abc ob= new Abc();

Abc ob;
ob = new Abc();
instanciation intialization.

② class Abc
{
int n;
int m;
public Abc()
{
n= 12; ← we also make a defaultconstructor then
m= 0; this become no argumented constructor
we use noargumented constructor.
for assign default value or
also assign our own value.
}
}

Scanned with CamScanner

Rule

① Constructor has the same name as class name.

② we differentiate the constructor ~~according~~ by changing their parameters.
- → type of parameter
- → no of parameter

③ Constructor is also a method with some ~~change~~ changing.

Scanned with CamScanner

(3) class Abc

```
{
  int h;
  int m;
  public Abc (int x, int y)
  {
      h = x;
      m = y; }
  public Abc (int x) {
      h = x;
      m = x; }
  public Abc () {
      h = 0;
      m = 0; }

  public Abc (Abj o) {
      h = o.h;
      m = o.m;
  }
```

Copy Constructor

no argument

If we don't make (no arg) constructor and make other type constructor then error at main. this cause error

```
Abc ob = new Abc ();
Abc ob = new Abc (10,20);
Abc obj = new Abc (12,0);
Abc ob2 = new Abc (1,0);
Abc ob3 = new Abc (10);
Abc ob4 = new Abc (0bi);
```

error need parameter

Lecture

- Constructor Overloading
- Packages.

Overloading is the concept of polymorphism.

Polymorphism.

- overloading
  - Constructor overloading
  - Method overloading
- Overriding.
  - Method overriding.

Static Polymorphism
- early binding
- compile time Polymorphism

Overload; work more than some one capacity

Overload by changing.
- no of parameter
- Type of parameter

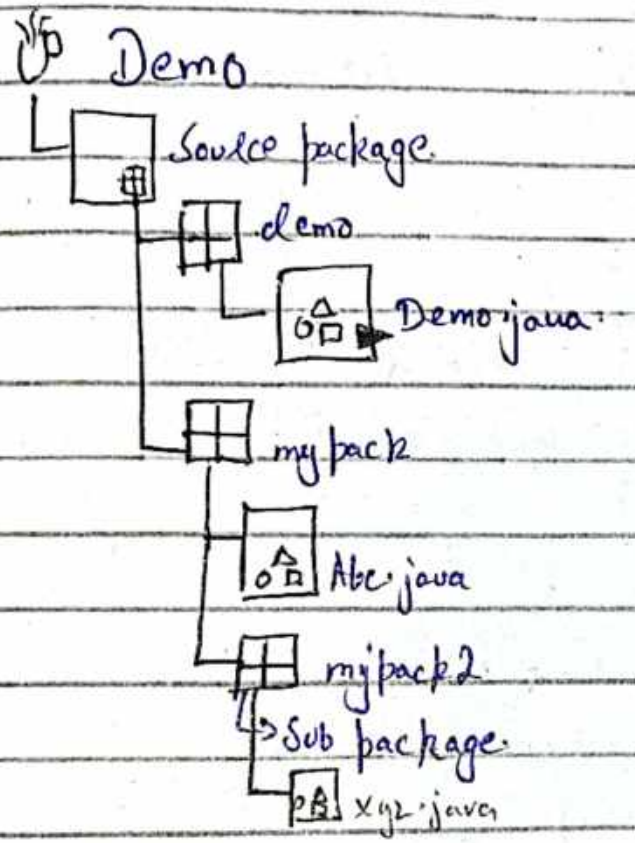
Purpose of constructor overloading:-

we can intialize the object in many different ways.

Scanned with CamScanner

Packages:-

- To Divide our code in different package.
  - ↳ To organize our code.



Demo.java

```
package demo;
public class Demo{
    public static void main(String[]a
    {
    }
}
```

Abc.java

```
package mypack
public class Abc {
    public static void main (String[]args)
    {
    }
}
```

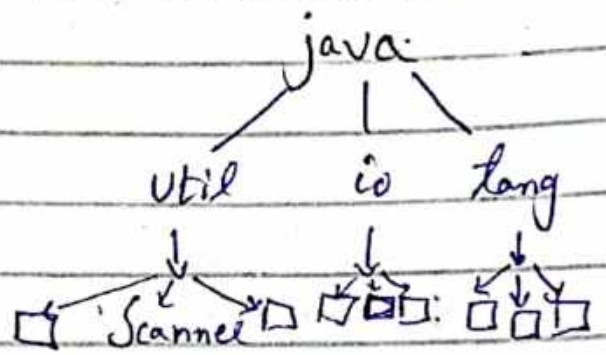
~~java → package~~

~~util → io lang~~

① no name clusion due to use of package.

Scanned with CamScanner

Java packages



package.

sub package.

classes, interface.

Access a package.

① Using import packageName.(*)

Access all the class in the pakages.

```
package demo;
import mypack.*;
import mypack.mypack2.*;
public class Demo{
    psvm ( ) {

Abc obi= new Abc();
Xyz ob2= new Xyz();
```

error.

because we can't access subpack
the subpack is treated as
other independent package.
so we also import them.

```
Xyz ob3= new Xyz();
```

Scanned with CamScanner

(2) Using import packageName.ClassName

(3) Using full name/path.

When class name is same and present in different package then we import Both class this cause error.
So use full path instead of import them

```
package demo;
import java.util.Date;
import java.sql.Date;
public class Demo {
    psvm () {

        Date ob = new Date();
                                  error
                                  can't recognise the class.
    }
}
```

```
package demo;

public class Demo {
    psvm () {

java.util.Date ob = new java.util.Date();
java.sql.Date obl = new java.sql.Date();
```

Scanned with CamScanner

Type of variable
Instance & Class Method
Method overloading.

Type of variable:-

Scope. where is demand?

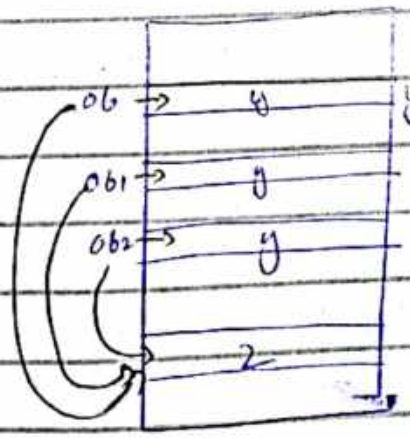
Lifetime, when variable created and destroy.

Lifetime of ~~static~~ instance variable.

↳ when u created the class.

Scanned with CamScanner

```
class Abc
{
    int y; //instance variable
    static int z; // class variable.

    public void hello(int x) //Local variable.
    {
Scope   int a;          // Local variable.
    Scope {
    }
    public void hello1()
    {
        Sout(z); }
}
```



Same reference of z for all object.

main

```
Abc ob = new Abc();
ob.hello;
```

a) lifetime of a & x is only when we remain in this line when we go to next line & & x destroy

```
Abc ob1 = new Abc();
```

```
Abc.z = 15;
ob.hello1(); 15
ob1.hello(); 15
```

main

```
System.out.print(Abc.z);
```

↳ access without create object.

Instance and Class Method

All instance scope things can't be access in static scope methods.

All the static scope things can be access in the instance scope things.

Scanned with CamScanner

```
public class Abc {

    int x;
    static int y;
    public int add(int a, int b) {      // instance method
        return (a+b);                   } x = ✓
    }                                   } y = ✓

    public static int add(int a, int b, int c)
    {                                   } x = ✗   //class method
        return (a+b+c);                 } y = ✓
    }
    public int sub(int a, int b) {
        return (a-b);
    }
```

```
Abc ob = new Abc();
ob.add(10,20);
```

```
Abc.add(10, 12, 30);
↳ direct call.
```

Scanned with CamScanner

Method Overloading.

Type promotion

add ((long), long)
add (long, double)
add (double, long)
add (double, double)

Overload:
- → # of parameter
- → Type of parameter

main
int a = 1235;
byte b = 10;
add (a, b);
↳ call add(long, long) method.

~~byte~~ → short,

Scanned with CamScanner

String & String operation:-

Value type ⟶ Built in datatype
Ref type ⟶ Arrays object.

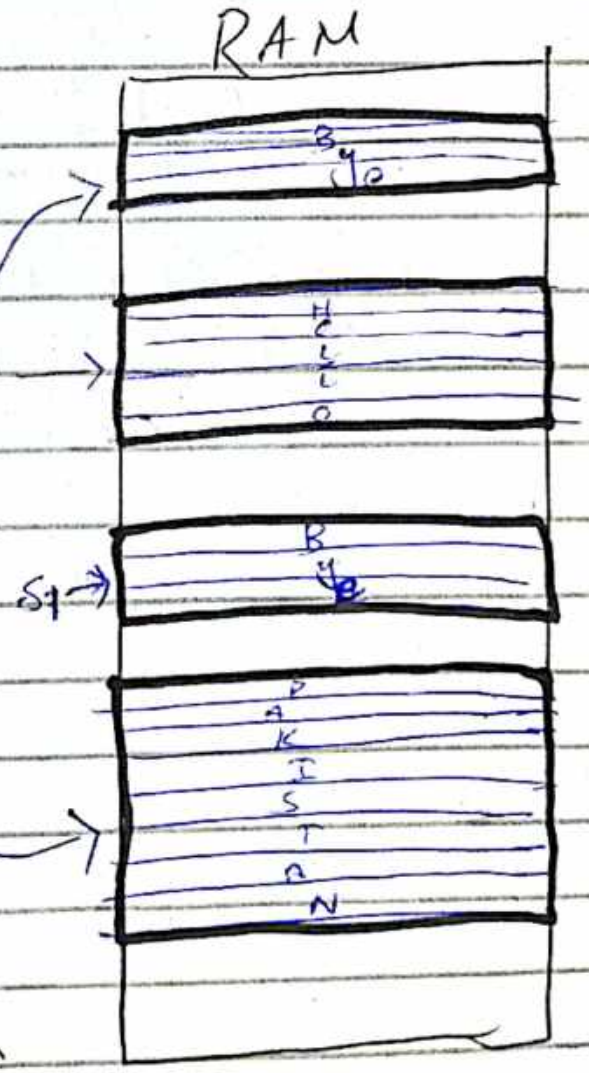
⇒ String is a class.

- String is b/w ref and value its immutable type, (work like ref type act like value type)

object and variable

String (S) = "Hello";
String s1 = "Bye";
S = S1
S = "Pakistan";

Make another char array in memory than chang ref from "Hello" to new "Bye".



Make another char array in memory of "Pakistan then change ref from "Bye" to "Pakistan"

Scanned with CamScanner

Operation:-

Sting (S) = " "; → object

S. ~~~~ multiple methods (instance)
↓
operation.

(1) char At( ) → int (char)
↳ return the char at specific index

(2) compare TO( ), (int) → String
↳ compare two string and return int value to substraction the character. if zero they move to next char if its not zero then return that value.

S1.compareTO(S2)

(3) compare TO Ignore Case( ) → String, (Int)
↳ compare string but ignore the case i.e 'I' and 'i' are equal.

(4) concat( ) → string (String)

S1 = S2.concat(S1)
Add two string work like $\boxed{S1 + S2}$

Scanned with CamScanner

(5) equals ( ) (Bolean)
↳ compare two string but return true or false.

(6) equalsIgnoreCase ( ) (Boolean)
↳ compare two string and return true or false but ignore case.

(7) endsWith( ) (True / false) (Boolean)
↳ check that the string ends with specific provided string.

S_1 = "Pakistan"
S_1.endsWith ("tan") ⟶ true

(8) indexof (char) (int)
↳ return the index of specific character.
If char is not in string return (-1).

S_1 = "Pakistan";
S_1.indexof (i) ⟶ 3

(9) indexof (string) (int)
↳ return the index of first char of string.

S_1 = "This is of other type"
S_1.indexof ("other") ⟶ 11

Scanned with CamScanner

(10) lastIndexOf(char) / lastIndexOf(String) (int)

Return the index of specific char and first char of string. but check from the last.

S2 = "India bad"
S2.indexOf(a) ⟶ 4
S2.lastIndexOf(a) ⟶ 6

(11) length() (int)

return the ~~the~~ length of a string.

S1 = "Hello";
S1.length() ⟶ 5

(12) matches(String) (Boolean)

Same as equals method.

Scanned with CamScanner

(13) regionMatches (intoffset, String, int, length) (Boolean)

s1 = "This is of other type".
s2 = "Islamabad

s1.(12, s2, 3, 5)

(14) replace (char, char) (String)

s1 = "Pakistan";
s2 = s1.replace('k', 't');
s2 = Patistan

(15) replaceAll (char/String, char/String) (String)

s1 = "Pakistan";
s2 = s1.replaceAll("ak", "he");
s2 = Pheistan

(16) replaceFirst (char/String, char/String) (String)

replace only first element of a string.

s1 = "Pakistan";
s2 = s1.replace('a', 'h');
s3 = s1.replaceFirst('a', 'h');

s2 = Phhkisthn
s3 = Phkistan

Scanned with CamScanner

(17) Starts With() (Boolean)

Same as endswith() but it check from start of string.

String start with specific string.

(18) to char Array. (Array of char)

convert the string to a char[].

```
s1 = "Pakistan";
char[] a = s1.toCharArray.
```

(19) trim()

remove the void space from first and last of a string.

```
s = "----Hello---";
s1 = s.trim()
```

$\boxed{s_1 = \text{"Hello"};}$

(20) toLowerCase() → convert all upper case to lower
toUpperCase() → convert all lowercase to upper.

Scanned with CamScanner

Convert char array to String

```
char[] all = {'h','e','l','l','o'};
String s = new String(all);
```

→ Substring
→ split.

Check a gmail adress.

1. @
2. .com Atend
3. @ ____ .com
4. no space

Scanned with CamScanner

Garbage Collection:-

Garbage: Some thing that you don't need and not for your interest (no more use).

Garbage Collection:- collect the garbage and remove them from the ram.

-> We can make things garbage by unreferencing them.

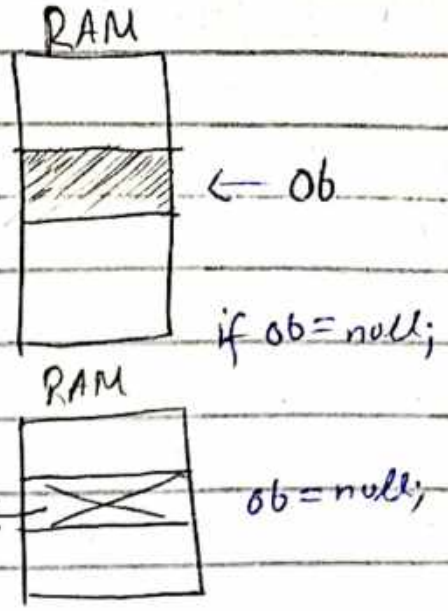
Unreferenced:-

1. By Nullifing the reference.
2. By Assigning reference To other.
3. Using Anonymous Object.
   - ↳ new ABC();
   - ↳ We use anonymouse object when we only us an object only for one time.

Scanned with CamScanner

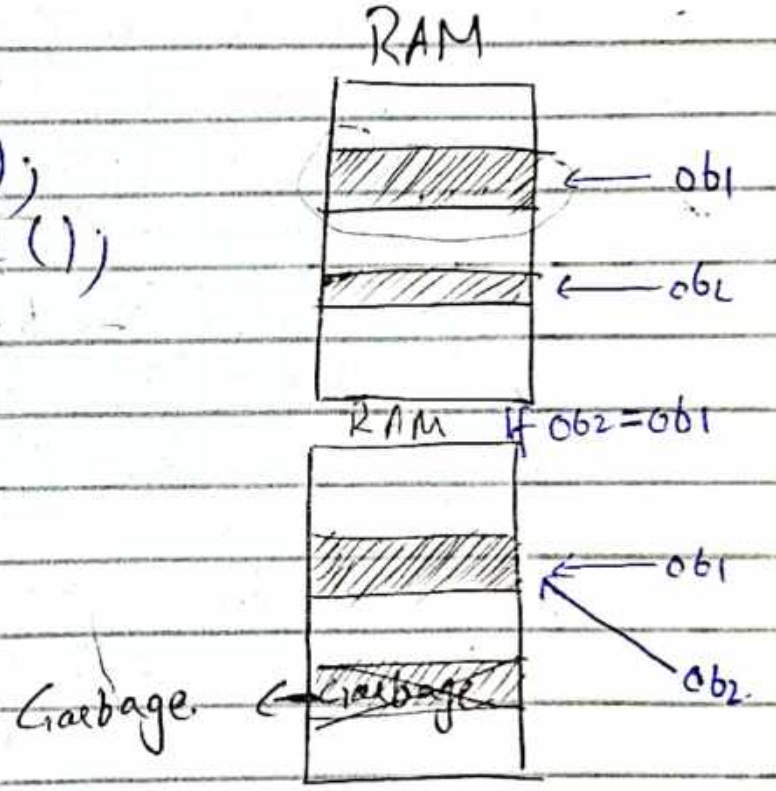
(1)

```
Abc ob = new Abc();
  // do whatever with ob.

ob = null; (1)
```



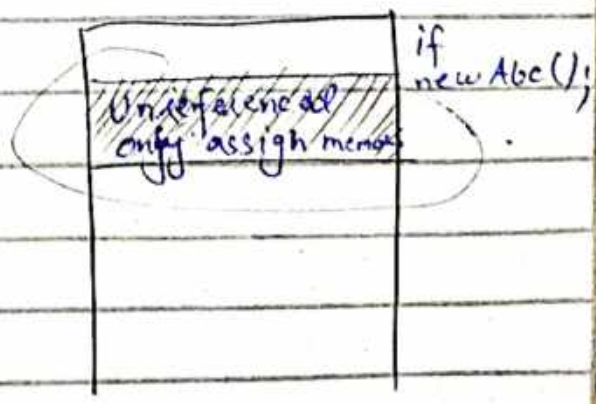
(2)

```
Abc ob1 = new Abc();
Abc ob2 = new Abc();
  : // code

ob2 = ob1;
```



(3)

```
class Abc {
    // code
    add(...)
      {....}.
}
```



main:-

new Abc().add(...); → object not created.

Scanned with CamScanner

Garbage Collector.

System.gc();

fin

There was possibility that some time object can't destroy.

Then chance of destroy of object is 70%.

Scanned with CamScanner

```
class Abc
{
    String name;
  public Abc (String n) { name = n; }.


  protected void finalize () {
      Sout (name + "destroyed");
    }
}
```

main:-

```
Abc ob1 = new Abc ("Obj 1");
Abc ob2 = new Abc ("Obj 2");
Abc ob3 = new Abc ("Obj 3");

ob1 = null    //nullifing the reference.
ob2 = ob3     // Assining reference to other.

new Abc ("Anyo");// Anonymous objects

System.gc();
```

Scanned with CamScanner

finalize :-

=> method that is hide in any class.

protected void finalize().

=> finalize method is empty nothing in them
↳ Default finilize like Default constructor

```
protected void finalize ()
{
    Sout ("Object destroyed");
}
```

finalize method is used to realese resources.

Scanned with CamScanner

Association:-

Association: relation b/w two things (تعلق)

=> Association b/w two object.
- Same class
- Different class

Association
- -> Composition (Strong Association)
- -> Aggregation (weak Association)

```
class Abc {
    int x;
    xyz ob= new Xyz();
}
```

main:-

```
Abc obj= new Abc();
```

obj.ob. -> Association obj.x;

Strong Association.

If we destroy Abc class obj the xyz ob object also destroy then its Strong Association

Weak Association

If xyz ob object can't destroy its weak Association.

Scanned with CamScanner

Weak:-

No [new] Keyword, object pass as a parameter

Strong:-

Object create as a new keyword.

Scanned with CamScanner

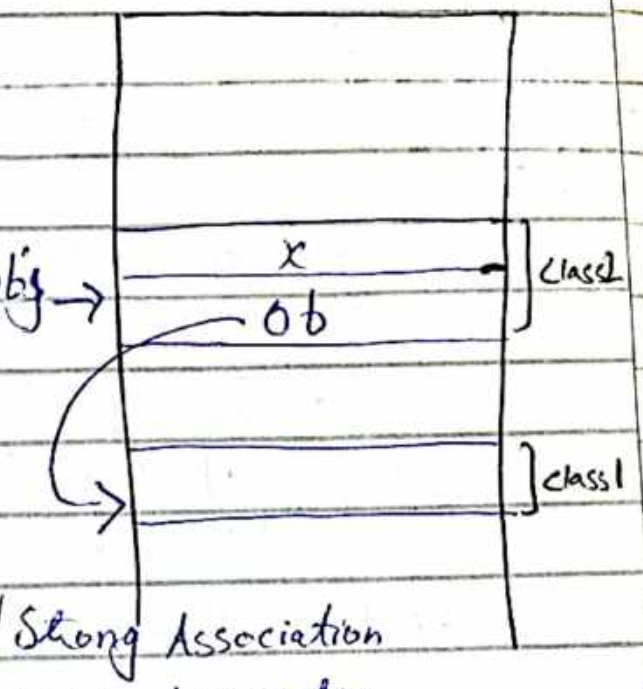
Strong Association:-

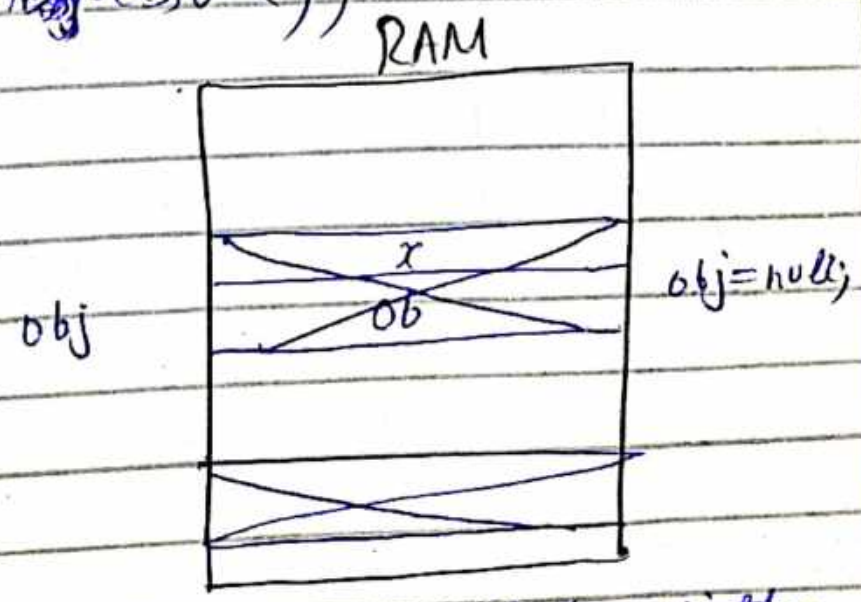
```
class Class1 {

// Code
}

Class Class2 {

int x;
Class1 ob = new Class1(); // Strong Association
xyz obz;
//Code
}
```

no association

main:-

```
Class2 obj = new Class2();

obj = null;
System.gc();
```





If we destroy obj then ob also destroy.

Nested classes

Nest:- گھونسلہ

Nested:- Any thing inside the same thing is nested

→ We need nested class for data security and to secure of class (inner)

→ To manage our code

Type of Nested class

1. Static nested class.
2. Inner class (non-static class)
3. Local class
4. Anonymous class. /Abstract class

Scanned with CamScanner

```
class outer
{
    int x;

    class inner
    {
        int a;
        int b;

    }
}
```

Scanned with CamScanner

1) Static nested Class

Static keyword make thing independent, use without instance

A class start with static keyword.

2) Inner class

class with no static keyword.

(3) Local class:-

A class that is created inside a method, object of this class is created after the blue print of class

```
public void meth()
{
    class Abc
    {
        //code
    }
    Abc ob=new Abc();
}
```

```
public void meth()
{
error Abc ob=new Abc();
    class Abc
    {
        //code
    }
}
```

Scanned with CamScanner

Inner class.

```
class outer
{
    class Abc
    {    //inner class.

    }
}

main:-

Outer ob = new outer();
Outer.Abc ob1 = ob.new Abc();
```

Static Class

```
class Outer
{
    static class Abc   //Static nested cl
    {

    }
}

main:-

Outer.Abc ob = new Outer.Abc
```

Local class

```
Class Outer
{
    public void method()
    {
        class Abc   //Local class.
        {

        }
    }
}
```

Scanned with CamScanner

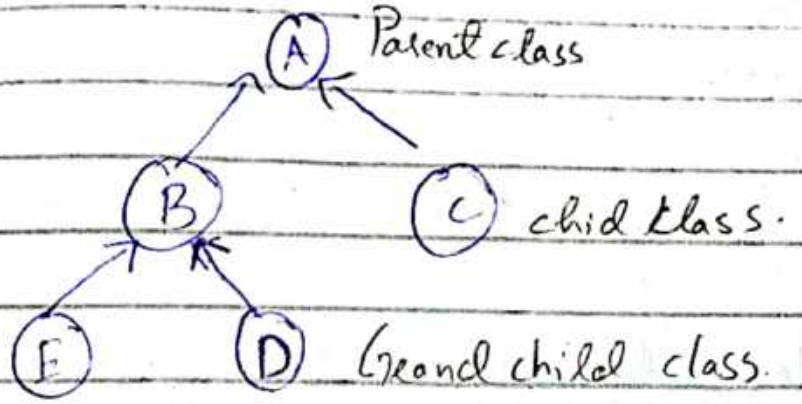
Anonymous classes.

```
abstract class Abc
{
    public abstract void method();

}

Abc ob = new Abc()
{
    pubic void method()
    {
        Soot("hello");
    }
};

Abc ob2 = new Abc()
{
    public void method()
    {
        Sout("Bye");
    }
};
```

Scanned with CamScanner

Inheritance:-

Inherit: وراثت



Keyword for inheritance:-

extends

functionality of inherit:-

1. Reusability.
2. Overide methods (method perform diff in child class)
3. extends the ~~com~~ already completed work.

Parent class ——— Child class
Super class ——— Sub class.
Base class ——— Derived class.

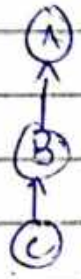
→ private things in parent class can't be access in the child class.

Scanned with CamScanner

Type of Inheritance:-

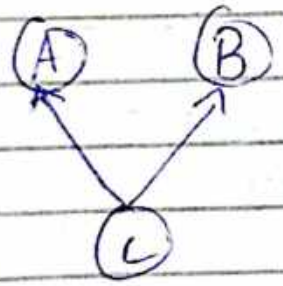
1. Simple inheritance
2. Multilevel inheritance
3. Hierarchical inheritance
4. Multiple inheritance.
5. Hybrid inheritance

Multilevel:-



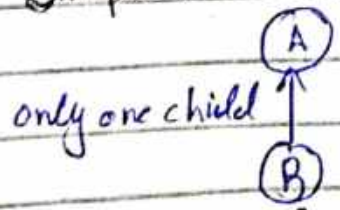

```
class A
{ //code }
class B extends A
{ //code }
class C extends B
{ //code }
```

Multiple:-



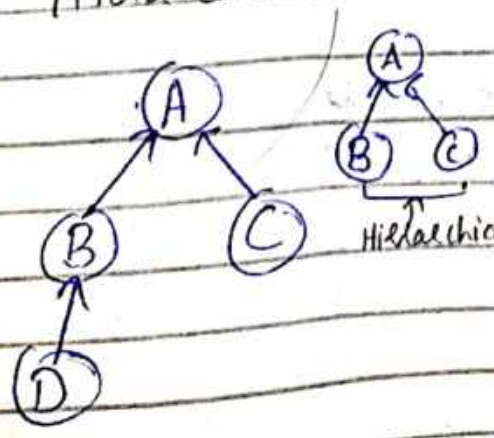
→ class C extend A, B (muliple)
Multiple inheritance is not allowed only for classes in java
work of multiple inherit can also use by using Multilevel inherit.

Simple:-




```
class A
{//code }
class B extends A
{ //code }
```

Hierarchical:-

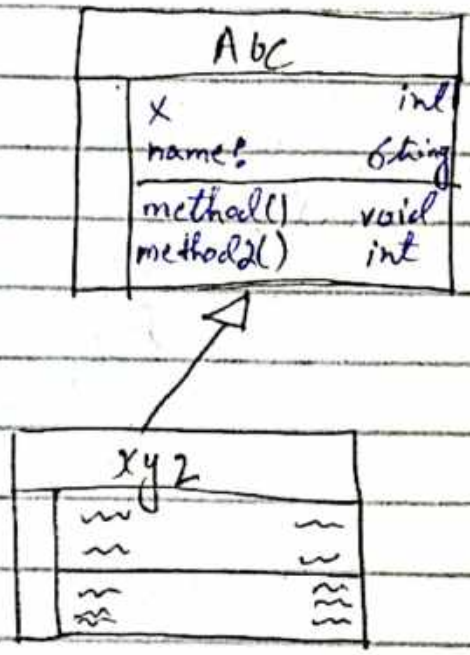



```
class A
{//code}
class B extends A.
{ //code }
class C extends A
{//code }
class D extends B
{ //code }
```

Scanned with CamScanner

Code:-

```
class Abc
{
    int x;
    String name;
    public void method(int x, int y)
    {
        // code
    }
    public void method2()
    {
        // Code
    }
}
```



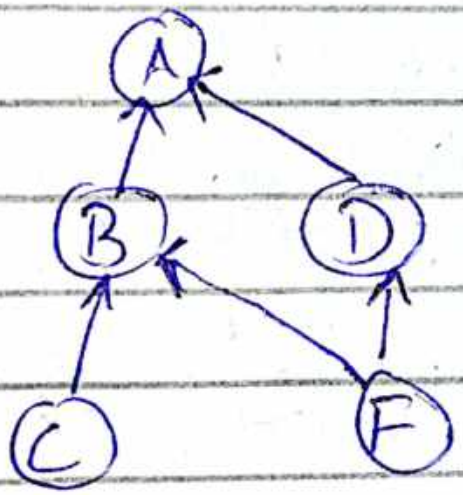

```
Class Xyz extend Abc
{
    int y;

    public void method3()
    {
        //code
    }
    @overcide
    public void method2()
    {
        //new code to change functionality
    }
```

Child → Xyz; Parent → Abc

Scanned with CamScanner

Hybrid.

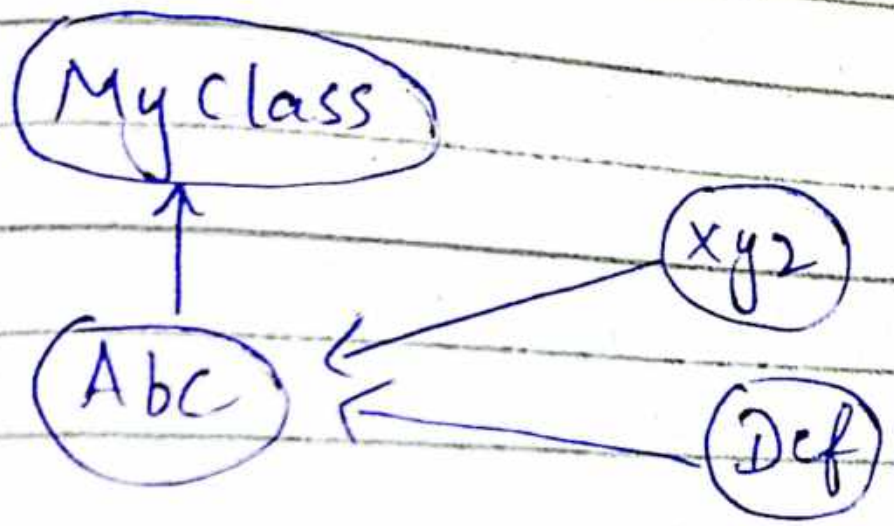
Multiple inherit + any other inherit.



not allowed in java.

Scanned with CamScanner

Example




```
class Myclass. { // code }
class Abc extends Myclass { //code }
class Xyz extends Abc. { //code }
class Def extend Abc { //code }.
```

Scanned with CamScanner

Constructor Chaining and Method overriding:-

Method overriding:-

Polymorphism — Overloading — Constructor, Method.
Polymorphism — Overriding

* Overriding is the second name of Overwriting

Overriding ≃ Overwriting

Override. Modify the inherited method

* For overriding of methods signature must be same i.e (return type, method-name and parameters) are same

Compiler Annotation :-
@

@override check ~~whether~~ weather ove method are override of not if signature can't match then they cause error.

this → represent same class
super → represent parent class

static method can't override it can be re-declared.

Scanned with CamScanner

=> Constructor can't be override.

```
class Abc
{

    public void method()
    {
        Sout(" Functionality of Abc);
    }

}

Class Xyz extends Abc.
{

}
```

main:-

```
Xyz obj = new Xyz();
obj.method();
```

output:-
Functionality of Abc

```
Class Def extends Abc
{
    @Override
    public void method()
    {
        Sout("Functionality of Def");
    }
}
class Check extends Abc
{
    @Override
    public void Method() error wrong name.
    {
        Sout(" ~~~~~~ ");
    }
}
```

main:-

```
Def ob = new Def();
ob.method();
```

output:-
Functionality of Def

Scanned with CamScanner

Constructor Chaining.

Chainging :- when things are connected together

~~create~~ Create a straight chain not make loop

Scanned with CamScanner

```
class A
{
  ~~public~~ int x;
        int y;
  public A()
  {      -> this(10) then infinite loop
    x = 0;
  }

  public A(int a)
  {   this();
      y = a;
  }
```

Scanned with CamScanner

Association Has-a relationship

Inheritance Is-a relationship

init block.

{

}

Scanned with CamScanner

Final Keyword:- (Keyword final)

Final Decision → can't be changed.
Final Result → Restriction

Use of Final keyword.

1. Class
2. Method
3. Variable

1,2,3 We restrict these thing by use final keyword.

Class:-

Final class can't be inherit.

Method:-

final method in super class can't be override in sub class

→ final method is inherited and use but can't be change, override.

→ no use of final keyword with constructor. useless.

Scanned with CamScanner

```
final class Abc
{     int a;
    public void me() {  //code
        }

}
```

```
class Xyz extends Abc
{
}
```

(X) compile time error

```
class Abc
{ int a;
final public void me()
    {
      //code
    }
```

```
class Xyz extends Abc
{
    @Override
    public void me()
    {
      Sout ("Hello");
    }
}
```

(X) error

Scanned with CamScanner

Valiable

Value can't be changed of that valiable.
↳ work as constant

Static variable can also be final.

~~Check~~ value of blank final variable (instance) can be intialize. (only in contructal)
↳ Can't be intialize in setter methods or other method

→ assign value for first time is intialization
↳ assign value second time is assignment

Static blank final variable can intialize in static contents (no in constructor)

Parameter of method also be final they can't be change but we assign value to them by calling method

Scanned with CamScanner

instance variable

```
class Abc
{
final double PI = 3.1415;
    public void me()
    {
    (X) PI = 3.2; error
    }
}
```

Static variable

```
class Abc
{
  final static int x=10;
}
```

```
main
(X) Abc.x = 20 ---> error
```

```
class Abc
{
    final double pi;
    final static int x;
    public Abc()
    {
      pi = 3.14;
    }
    public Abc(double a)
    {
      pi = a;
```

Blank final variable

Uninitialize final variable

Blank static final variable

Scanned with CamScanner

Static ~~Sepos~~ Scope

```
static
{
    x = 15;
}

public void method (final int a)
{
    a = a + 10;        error
    final int c = a;
    c = c + 2          error
```

```
main:
Abc ob = new Abc();
ob.method (10);
```

Scanned with CamScanner

Abstraction :-

Abstraction

Abstract class (0% — 100%) Not Support Multiple inheritance	Interface (100%) Support Multiple Inheritance

⇒ force the ~~inherited~~ sub class to implement abstract method.

⇒ if there is no method abstract in abstract class, then we can't create object of them but we can implement any method in ~~extend~~ sub class.

Keyword

abstract

Scanned with CamScanner

```
abstract class Abc
{

    public void hello()
    {
        System.out.print("hello world");
    }

    public abstract void method();
    public abstract int calculate(int a, int b);
}

class Myclass extends Abc
{
    @Override //Compiler Annotation
    public void method()
    {
        //code. (implement your code)
    }
    @Override
    public int calculate (int x, int y)
    {
        return (x+y)
    }
}
```

(if final) ↓ can't override

if type promotion can't do type promotion signature same.

main

```
Abc ob = new Abc();
```
error

```
Abc ob = new Abc()
{ //implement void method();
};
```

concrete, non-abstract

abstract

Scanned with CamScanner

Interface (100% abstract)

Keyword:

interface.

→ interface also have variables.

If we have a method in interface and same signature abstract method in abstract class.
then if we extend abstract class and implement interface then we only implement one of two signature method.

class & class	extends
class & interface	implements
interface & interface	extends. (Do muliple inheritance)
interface & class.	not allowed.

Scanned with CamScanner

→ interface Abc

```
{
default public void me()
{
    //Code
}
public void print();
public int calculate (int a, int b);
}
```

error (only abstract things)
(Use default keyword if do this)

→ interface Xyz {
public void hello(); → access modifier check, Yes
lower level to upper level
~~upper to lower not allow~~
}

class MyClasses implements Abc, Xyz

```
{
@ override
public void print()
{
    //Code
}
@ Override
public int calculate (int a, int b)
{
    return (a-b)
}
@ Override
public void hello()
{
    //Code
}
}
```

① interface Abc ② interface Xyz, ③ abstract class Abs

class extends Abs implement Abc, Xyz. Multiple inheritance

Scanned with CamScanner

```
interface IntA
{
    public void method1();
}

interface IntB extends IntA
{
    public void method2();
}
```

```
abstract class Abc
{
    public abstract void hello();
}
interface Abs
{
    public void hello();
}
class MyClass extends Abc implements Abs.
{
    @Override
    public void hello() → only implement one hello.
    {
        //code
    }
```

Scanned with CamScanner

Exception handling:-

Exception:- disturb the normal execution of program.

=> Error and Exception are completely different.

Error → Logical, Syntax, run-time error

Handle a situation before for that situation is Exception handling.

Type of Exception:-

1. Checked Exception/Compile time Exception
2. Unchecked Exception / Run time exception
3. Error

1. → IO Exception (compiler check them)

   Example:-

   [File f = "D:/Abc.txt"] IOException.

2. (compliers can't predict them)

   → Arithmetic 5/0

   → Array out of bound.

3. → JVM, JRE, insufficient memory

   System issues.

   System error.

=> Its different from coding error

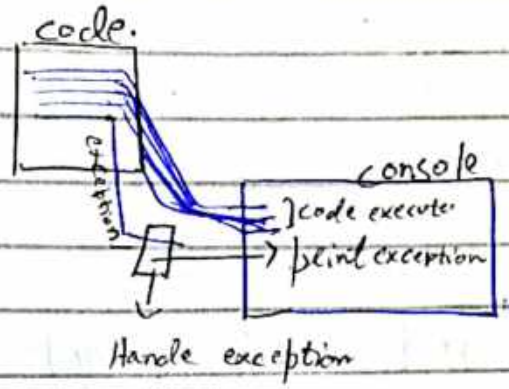
catch the exception and don't disturb the execution of program.

Scanned with CamScanner

Try & Catch

try to run this code if exception occur then catch that exception.

→ when their comes exception then catch excute otherwise try executes



try-catch code

```
try
{
    // code that can possibaly generate Exception
}
catch (Exception ex)
{
    // Your response
}
```

```
public static void main (Strings [] args) (Arthemetic exception)
{
    Scanner in = new Scanner(System.in);
    int a, b;
    a = in.nextInt();
    b = in.nextInt();
    try { System.out.print(a/b); }
    catch (Exception ex)
    {
        System.out.print("You can't divide by zero");
    }
}
```

Scanned with CamScanner

Multiple catch block.

```
public static void main(Strings [] args)
{
  Scannee in = new Scannee (System.in);
   int a,b;
  try
  {
    a=in.nextInt();
    b= in.nextInt();
     System.out.print (a/b);
  }
 catch (InputMismatchException ex)
  {
    System.out.print ("Wrong Input");
  }
catch (ArithemeticException ex)
  {
   System.out.print ("Divide by zero");
  }
 catch (Exception ex)
  {
   System.out.print ("Exception");
} }
```

Scanned with CamScanner

Multiple catch block code

```
try
{
    //code
}
catch (Exception type ex)
{
,
}
catch( Exception type ex )
{

}
------ another catch block. if need
catch (Exception ex)
{
  System.out.(ex.message);
}
```

if no any catch found then execute this catch just like else-statement in else-if ladder. (It must be last)

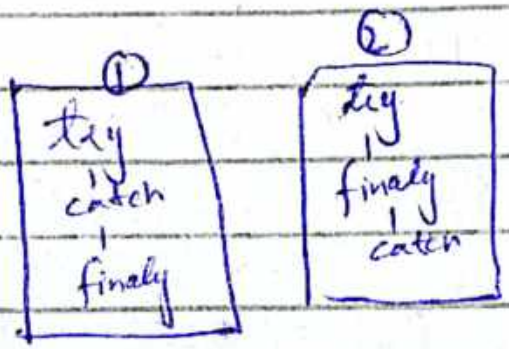
Scanned with CamScanner

• throw, throws, Finally Block, Chained exception
Custom Exception.

throw
↳ Keyword.

throws

Finally Block.




```
//code
try
{ ————————
  ————————}
catch ( . . . )
{ ————————
  ————————}
finally
{
    // Always Executes
}
```

statement that is always excuted, very imp then put them in finaly block.

finaly is not alone always with try.

Scanned with CamScanner

```
class Abc
{

    public void method() throws Exception
    {
        Scanner in = new Scanner (System.in);
        int a = in.nextInt();
    }
}

main:
        Abc ob = new Abc();
        try
        {
            ob.method();
        }
        catch (Exception e)
        {
            //code
        }

try
{
    File f. . . (------)
    //code
    file.close(f);
}
catch ( . . . )
{

}
finally
{
    file.close(f);
}.
```

exception catch

Nested try :-

try-catch in one ~~try-ca~~ try-catch block.

Chained Exception (rarely used)

```
try
{
    try
    {
        try
        {
            ①
        }
        catch ( ... )
        {
            ①
            throw e;
        }
        ②
        catch( ... )
        {
            ②
            throw e;
        }
    }
    ③
    catch ( ... )
    {
        ③
    }
}
```

Chain

Chain

Scanned with CamScanner

nested try:-

```
int a=10, b;
Scanner in=new Scanner(System.in);
try
{
    try
    {
        b=in.nextInt();
    }
    catch(InputMisMatchException e)
    {   in=new Scanner(System.in);//wrong input, input again.
        b=in.nextInt();
    }
    try
    {
        System.out.print(a/b);
    }
    catch(ArithmeticException e)
    {
        b=1
        Sout("Divide by zero change to 1="+a/b);
    }
    catch(———)
    {  //header
    }
}
```

Scanned with CamScanner

Generics

Generic :- Something that is very general.
Generic are also called parameterized class.
Code :-

```
class Abc (class with two int variable)
{
 int a=0;
 int b=0;

 public void setData(int x, int y)
 {
    a=x;
    b=y;
 }

 public void printData()
 {
 System.out.print("a is"+a+"b is"+b);
 }
main_ Abc ob = new Abc();
      ob.setData(10, 20);
      ob.print();
Xyz  ob1 = new Xyz();
     ob1.setData("hello", "World");
       ob1.print Data
```

Scanned with CamScanner

```
class Xyz {                    (class with two String variable)
    String a, b;
    public void SetData (String a, String b)
    {
        this.a = a;
        this.b = b;
    }
    public void printData ()
    {
        System.out.println(a + "  " + b);
    }
}
```

~~Generic~~ Generic class :-

```
class Abs <DT>                 (more than one as parameter)
                 → It should be all capital (naming convention)

    DT a;
    DT b;
    public void setData (DT a, DT b)
        { this.a = a;
          this.b = b;
        }
    public void printData ()
    {
        System.out.println(a + "  " + b);
```

main :-

```
① Abs <Integer> ob = new Abs <> ();
    ob.setData (10, 20);

② Abs <String> ob = new Abs <> ();
    ob.setData ("hello", "World");

③ Abs <Double> ob = new Abs <> ();
    ob.setData (17.3, 8.2);
```

Wrapper classes for Built in Data Type

style

Scanned with CamScanner

Wrapper :- Insistance for build in Data Type variable act as object.

int a = 10; (variable)
Integer a = 10; (object)

Wrapper class object also has different operations.

Data type ~~variable~~	wrapper classes
boolean	Boolean
char	Character
int	Integer
byte	Byte
short	Short
long	Long
float	Float
double	Double

```
Class Abc <DT, V>
{
  DT i;
  V 2;
public void SetData (DT i, V 2)
{
    this.i = i;
    this.2 = 2;
}
public void printData ()
{
  System.out.print(i + " " + 2);
```

Scanned with CamScanner

8

$^8P_2 = \underline{8 \times 7}$

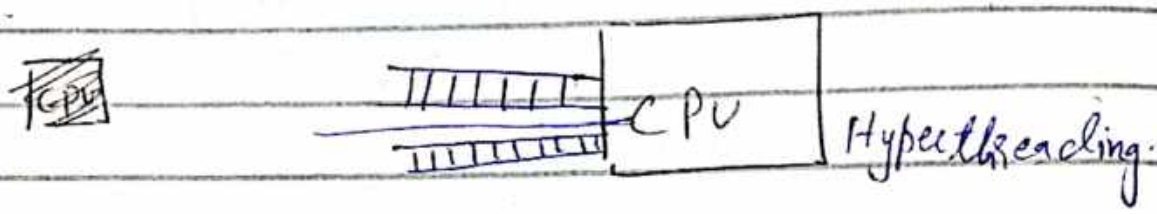
main

```
Abs<Integer, Double> ob1=new Abs<>();
ob1.setData (10, 15.3.

Abs<String, Character> ob2=new Abs<>();
ob2.setData ("hello", 'F');

Abs<Boolean, Double> ob3 = new Abs<>();
ob3.setData (true, 8.2);
```

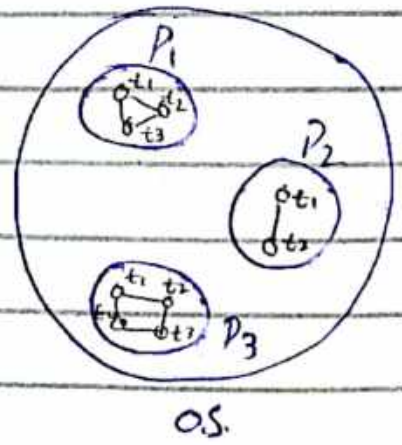
Scanned with CamScanner

Multithreading:-

Multi threading : Hyperthreading
Multiprocessing :
Multi tasking :

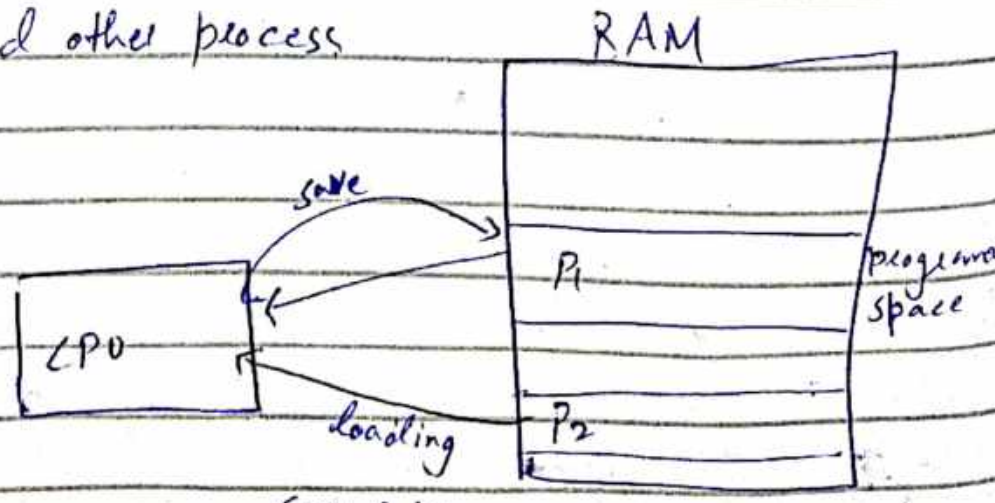


* Divide whole programe into small parts threads.
* In a process there are multiple threads
* threads is light weight task

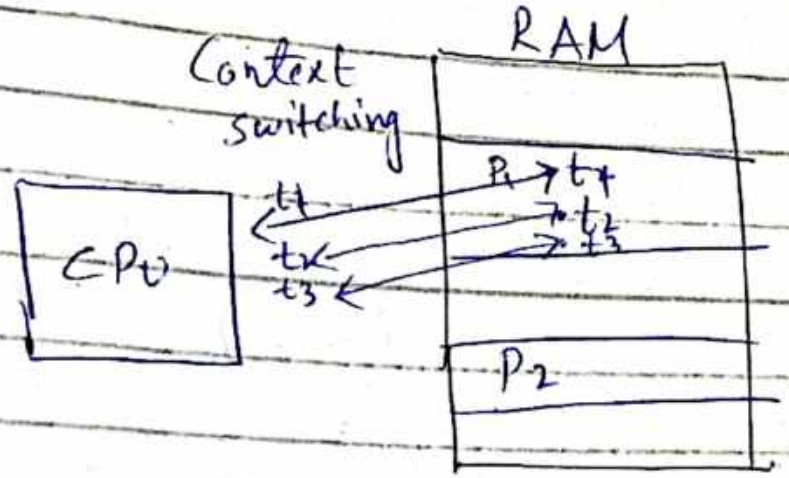


Why thread is important?

first save then load other process



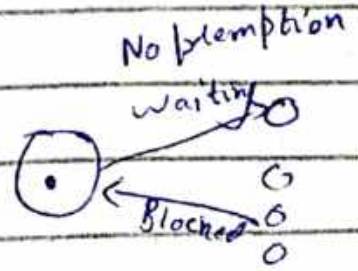
Scanned with CamScanner



Better to make threads instead of process.

State of threads

1 - New
2 - Runnable (loading)
3 - Running (Actual excution
4 - Blocked (Someone stop working)
5 - Waiting
6 - time waited
7 - Terminated



Way to create threads.

2 ways.

1. To implement Runnable → interface
2. To Extends Thread → class

Scanned with CamScanner

```
class Abc
{
  public void run()
  {
    for (int i=1 ; i<11 ; i++)
    {
      Sout(" Iteration # " + i);
      Thread.sleep(500)
      → milli second
    }
  }
  public void start()
  {
    run();
  }
}
```

```
class xyz
{
  public void run()
  {
    for (int i=1; i<11 ; i++)
    {
      Sout("Iteration # "+i);
    }
  }
  public void start()
  {
    run();
  }
}
```

main :—

```
Abc ob1 = new Abc();
xyz ob2 = new xyz();

ob1.start();
ob2.start();
```

1 — 10 1 2 3 4 5 6 7 8 9 10

1 — 10

```
class Xyz extends Thread
{
  @Override
  public void run()
  {
  for(int i=1; i<=10; i++)
  {
    Sout("Iteration # "+i);
  }
}
```

```
class Abc implement Runnable
{
  @Override
  public void run()
  {
  for(int i=1; i<=10; i++)
  {
    Sout("Iteration # "+i)
  }
}
```

RAW
WAR
WAW

main:-

```
Xyz obl = new Xyz();
Abc ob2 = new Abc();
Thread t1 = new Thread(ob2)

obl.start();
t1.start();
Sout("Hellor");
```

Execute Abc without implement Runnable

* if statement is independent then use threading
if statement is Dependent the don't use it

int a=10; int b=20 — Independent
int c=a+b; Dependent on a and b

* check Dependency

GUI:-

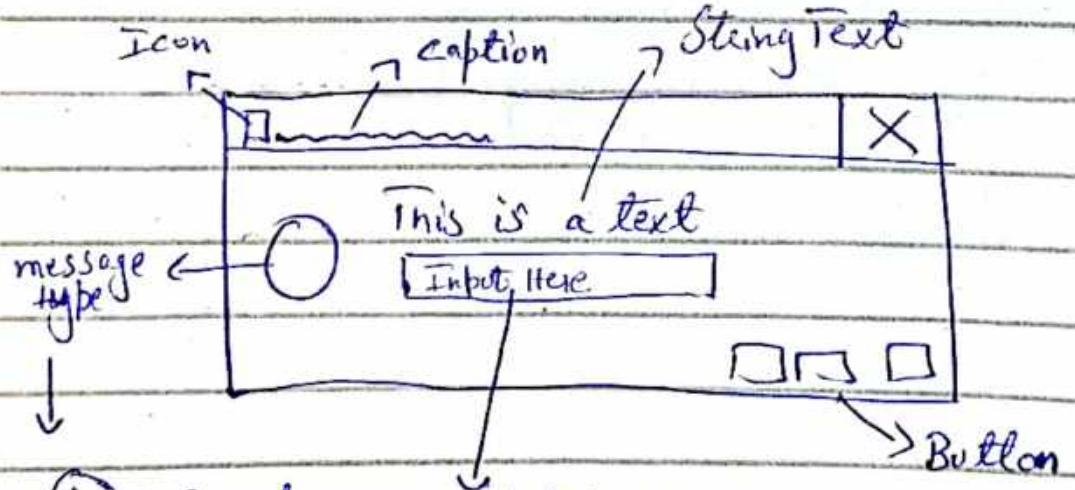
Swing component:-

Javax.swing.Jxxx...

It can be ~~can~~ made from:
1. By code
2. Drag & Drop

Dialoge Box:- (Static method)

messageDialoge → Delivering message
input Dialoge
Confirm Dialoge → for confirmation (Yes/No)



1. information
2. warning
3. Error
4. Question

Scanned with CamScanner

Message Dialoge (3 overload)

```
impact javax.swing.JOptionPane;

public static void main(String[] args)
{
```

① JOptionPane.showMessageDialoge(null, "window Display");

(show parent) (message)

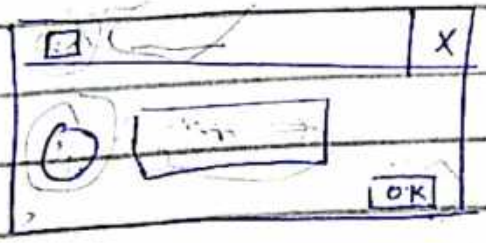
② JOptionPane.showMessageDialoge(null, "My window", "window Display", JOptionPane.WarningMessage());

also use number instead of this.

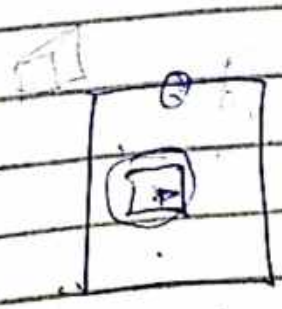
0
1
2
3

③ Icon ico = new Icon("C:\> i.ico");

JOptionPane.showMessageDialoge(null, "Mywindow", "windowDistaged", JoptionPane.ErrorMessage(), ico);



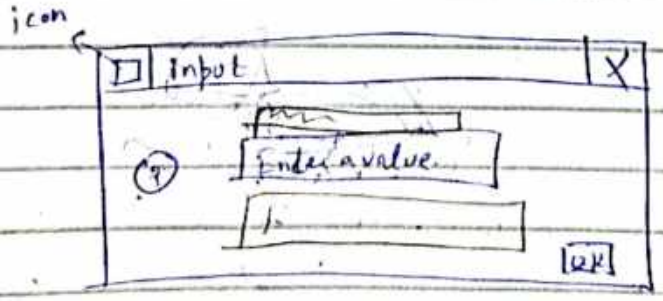
Message Dailoge Box



Scanned with CamScanner

Input Dailoge Box:- (6 over load)

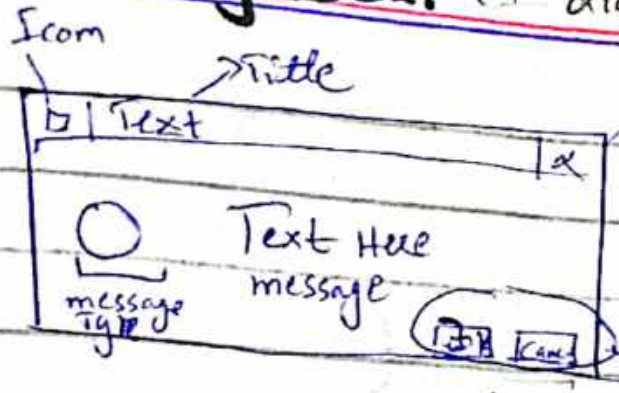
```
impact javax.swing.JOptionPane;

public static void main(String[] args)
{
String s;
① s= JOptionPane.showInputDialog("Enter a value");
                                  object

② s = JOptionPane.showInputDialog(null, "Enter a value");
                                  component   object

③ s=JOptionPane.showInputDialog(null, "Enter a value", "Number only
                                cmpt        o            o1
                                );

④ s=JOptionPane.showInputDialog(null, "Enter a value", "Number only".
                                cmpt        o           o1
                                JOptionPane.informationMessage);
                                int

⑤ s=JOptionPane.showInputDialog(null, "Enter a value", "Number only"
                                cmpt        o           o1
                                JOptionPane.informationMessage, ico);
                                int                         pic

⑥
```



① (message)
② (component, message)
③ (message, initial value)
④ (component, message, initial value)
⑤ (component, message, title, message Type)
⑥ (component, message, title, message Type, icon, selectionValues, initialSelection);

Scanned with CamScanner

ConfirmDialog Box. (4 dialogue box)



→ int
0. → Approve opt
1 → Disapprove
2 → cancel

→ option type

Yes
OK
Yes No.
Yes No Cancel.

Approve, DisApprove, 3rd option/cancel

{

only

① int ch = JOptionPane.showConfirmDialog(null, "Are you sure?");

Text

② int ch = JOptionPane.showConfirmDialog(null, "Are you sure",
"Select", JOptionPane.YES-No-Cancle-Option);

Title

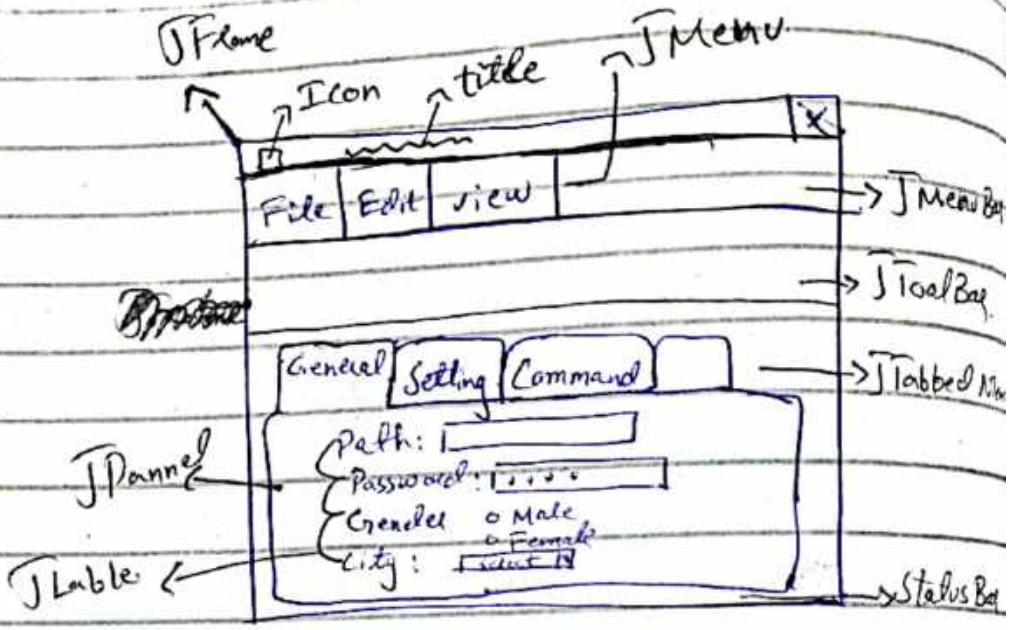
ly"

③ int ch = JOptionPane.showConfirmDialog(null, "Are you sure",
"Select", 2, 0

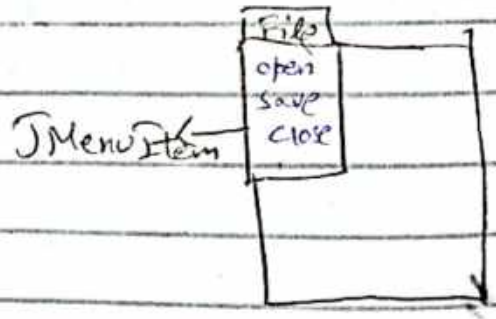
Scanned with CamScanner

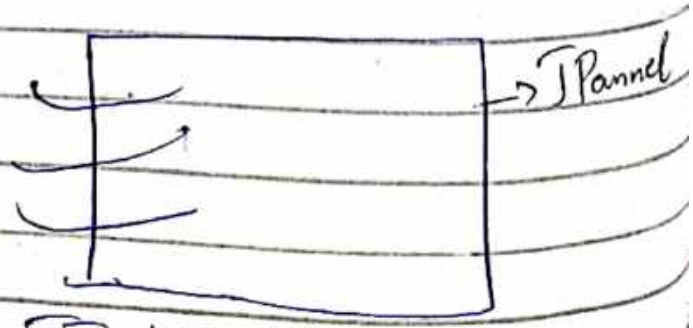
Swing Component

JFrame
- → icon
- → title
- → Size




```
JMenuBar jm = new JMenuBar();
JMenu j = new JMenu("File");
jm.add(j)
```





Path: [ ] — JTextField

Passwrd: [...] — J Passwrd Field

Gender { Male, Female } — JRadio Button

Scanned with CamScanner

JButtonGroup bg1 = new JButtonGroup();

Interest

☐ Sport
☐ programming
☐ internet

JCheckBox

City : select City

Mansehra
Lahore
Peshawar

JComboBox.

log:

JTextArea.

Save Cancel

JButton

City

Mansehra
Lahore
Peshawar
Abbotabad

JList

10

JSpinner.

Scanned with CamScanner

```
import javax.swing.*

class MyFrame extends JFrame
{   JTextField txt.
    JPanel pnl;
    JLabel lbl1;
    JLablel lbl2;
    JComboBox cbox;
    JButton btn;
    public MyFrame()
    {  txt = new JTextField();
       pnl = new JPanel();
       lbl1 = new JLable("This is a lable");
       lbl2 = new JLable();
       cbox = new JComboBox();
       btn = new JButton();

       lbl2.setText("This is other label");
       cbox.addItem("This is option 1");
       cbox.addItem("This is option 2");
       cbox.addItem("Option 3");

       btn.setText("Save");

       this.setPrefferedSize(new Dimension(200,200));
       this.setResizable(false);
       this.setTitle("MyWindow");
       txt.setPrefferedSize(new Dimension(150,15));
       lbl1.setForgeound(Color.RED);
       pnl.setlayout(new FlowLayout());

       pnl.add(lbl1);
       pnl.add(txt);
```

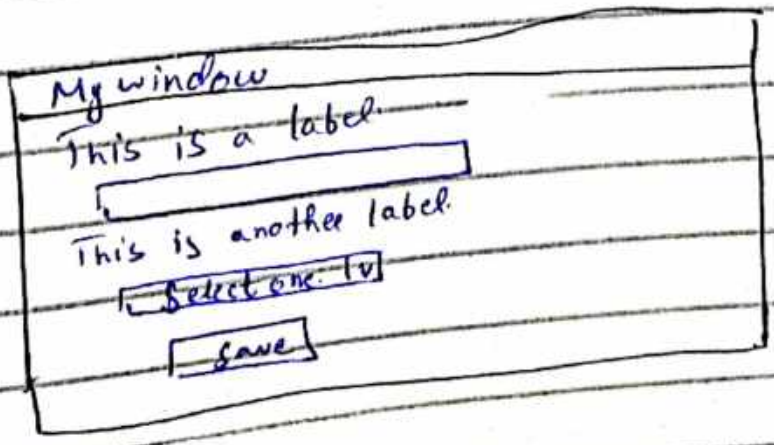
cbox.addItem("Select o

Scanned with CamScanner

```
pnl.add(lbl2);
pnl.add(cbox);
pnl.add(btn);

this.add(pnl);
}
```

main:-

```
MyFrame f = new MyFrame();
f.setVisible(true);
```



Scanned with CamScanner

GUI using Drag and Drop

Event Handling using code

Event Handling using Drag & Drop

→ Backend functionality

Full stack Developer.

→ Front End.

→ Back End.

Event:

Listener — Mouse — Enter, Realease, Click, Exist

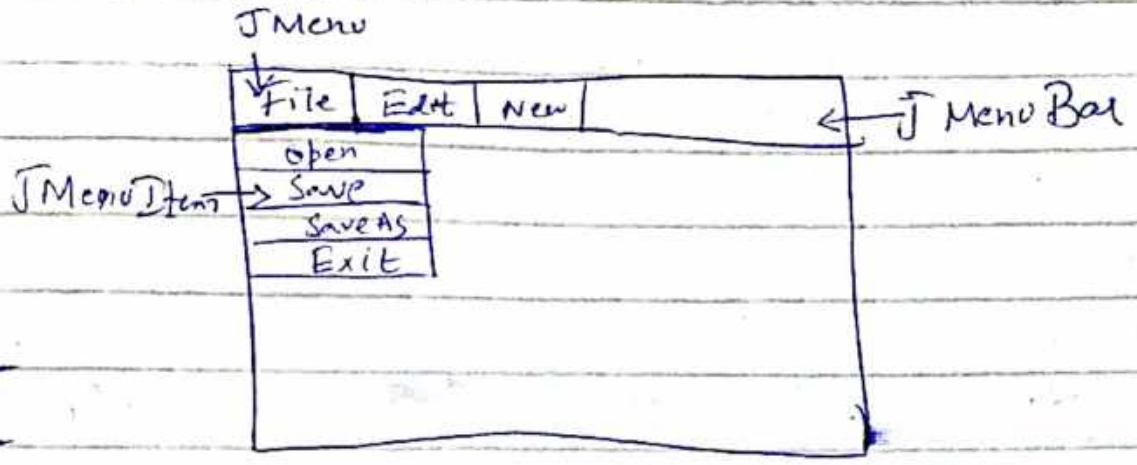
Listener — Action

Listener — Key

Scanned with CamScanner

GUI:-

Menu Bar.



Event Handling:-

Action Perfaimed.
↳ Write code here

Label LbL

JLabel.

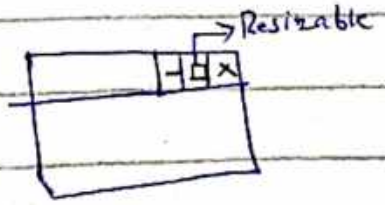
Properties:-

① font change Text size, bold, style
②

Also Paste logo at label.
cant add multiple text at label.

Resizable:-

① True or False
↳ in properties



Scanned with CamScanner

Button :- btn

Event Handling :-
Action Performed.
→ Code Here

Event Handling :-

Key
- Pressed
- Released
- Typed.

Mouse
- Clicked
- Enter
- Exit
- Pressed
- Released

listener
- Key
- Mouse
- Action

Scanned with CamScanner

(12) Check Box
(16) Date.
(15) Panel

Text Field. txt

(1) getText() (2) isEmpty() (3) grabFocus()

Text area:- ta

Properties → line wrap (no cursor)
(1) setText();

ComboBox; cmb

obj · addItem("—"); Code

Properties → model → addItem there.
· getSelected ~~Value~~ Item ()
· getSelectedIndex();

Password Field txt

Scanned with CamScanner

File Chosser

JFileChosser f = new JFile Chosser.
f. showOpen Dialoge (this);
File ff = f. getSelected File();

Check Box: chk

- isSelected();
- getText();

Tabbed Pane

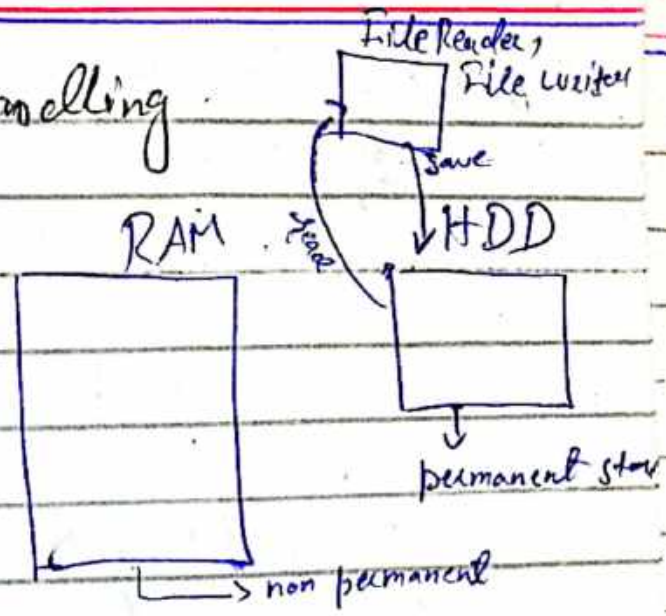
→ Use to add Multiple panel.

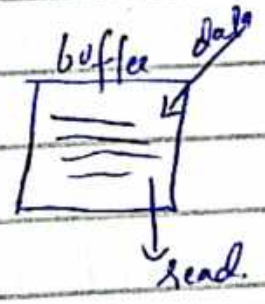
AddFrom Palette → SwingContainer → Panel.

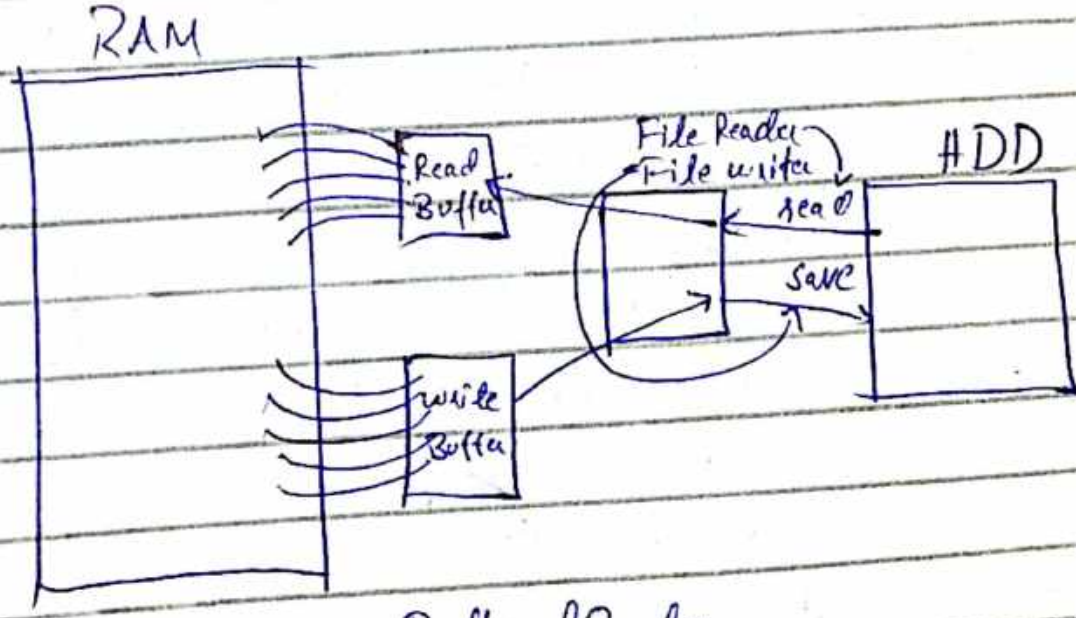
Scanned with CamScanner

Filing / File Handling

① Read (File Reader)
② Write. (File writer)







File → FileReader → BufferedReader
File writer → Buffered writer

Scanned with CamScanner

Reddar:-

```
File f = new File("E:\\Test.txt");
FileReader fr = new FileReader(f);
BufferedReader br = new BufferedReader(fr);
String line = br.readLine();

while (line != null)
{   sout(line);
    line = br.readLine();
}

br.close();
fr.close();


Scanner in = new Scanner(System.in);
File f = new File("E:\\Abc.txt");
FileWriter fw = new FileWriter(f);
BufferedWriter bw = new BufferedWriter(fw);
sout("Enter Text to save in the file: ");
String line = in.nextLine();
bw.write(line);

bw.close();
fw.close();
```

Scanned with CamScanner

Socket Programming.

Socket provide Data Flow.

Port no:- 2^{16} : 0 — 65535

Address , port

```
import java.net.Socket;
import java.net.ServeSocket;
```

hello

send [192.168.10.55 , 3487]

listen loop [Server Socket 3487
accept.
Socket s = ←] Thread

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
```

Scanned with CamScanner

~~Break~~ Send

```
try{ Socket s = new Socket(txtIP.getText ; 4567);
Data Output Stream out = new DataOutputStream(s.getOutputStream);
    out.writeUTF(txtmessage.getText());
    out.close();
    s.close();
    txtmessage.setText(" ");
}catch(Exception){}
```

Recieve.

```
ServerSocket SS;
Thread Listen;
```

Inner Class

↳ class Listen extends Thread.

```
{
    public void run()
    {
        while (true)
        {
        try
        {
            SS=new ServerSocket(4567);
        Socket s= SS.accept();
        DataInputStream in = new DataInputStream(s.getInputStream);
StringmSg=in.readUTF();
String sender= s.getInetAdress().toString();

txtmain.setText(txtmain.getText() + "\n"+ sender+" "+ msg);
    in.close();
    s.close();
    SS.close();
}catch() {}
```

Scanned with CamScanner

```
public form()
{
    listen = new Listen();
    listen.Start();

}
```

IP 127.0.0.0.1

Scanned with CamScanner

Data Base Connectivity

JDBC → Derby

netbeam have integrated DataBase

- → select
- → insert
- → delete

Goto Services → Databases → JavaDB

Create Data Base

Excel

ID	NAM	Last	Salay	AGE
1	SHA	eeq	1000	20
2				

JavaDB → right click (start server) then (Create Data Base)
- → Data Base Name → Comsat
- → User Name → ABCD
- → Password →

JavaDB Now have Data Base.

Right click on created DB (Comsats.) and (connet)

ABCD
- Tables → Right click
- View
- Procedure

Scanned with CamScanner

Create Table
Table name: Emp
Add column → Click

Add Column
Name : ~~Type~~ ID
TYPE : Integer
Size : Scale :
Default :
☐ Primary key ☐ Unique

Primary key → for unique adress.

1 → ID → Integer
2 → FName → Varchar 40
3 → Salary → Double.
3 Column.

Varchar
Variable
character
As String

ABCD
- Table
  - Emp
- View
- Procedure

Scanned with CamScanner

SQL is not case sensitive

#	ID	FName	Salary
1	1001	ALI Khan	6500.50
2	1044	Ahsan Khan	6504.75

insertion

(1) insert into ABCD.emp values(1001, 'ALI Khan', 6500.50);
insert into ABCD.emp values(1044, 'Ahasan Khan', 6504.7)

Selection

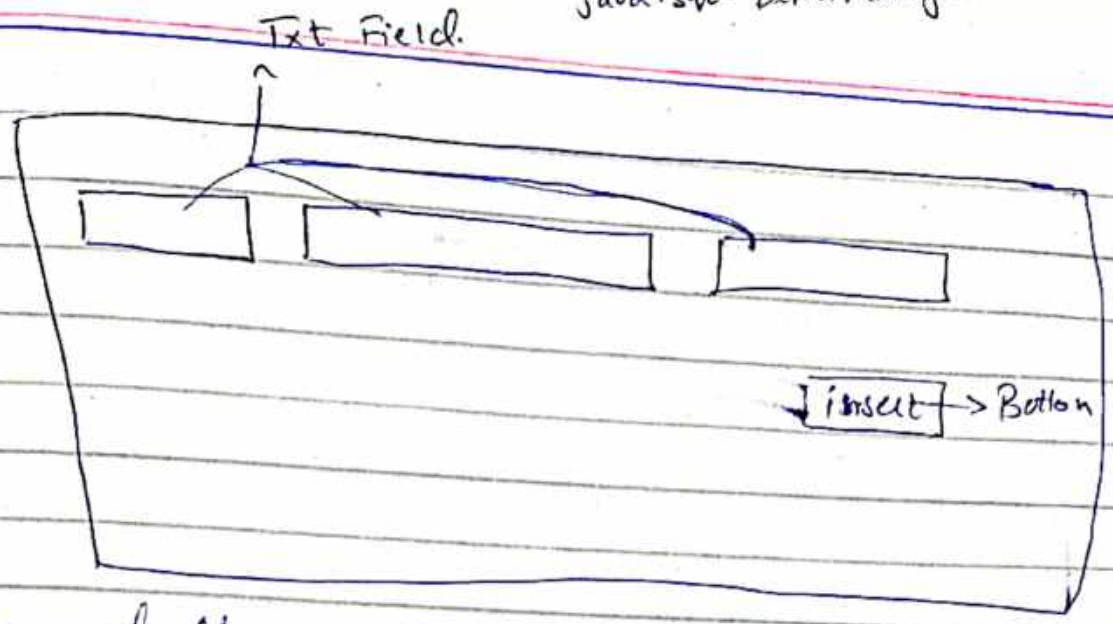
Select * From ABCD.EMP where ID = 1001
Select * From ABCD.EMP
↳ select all Data
select * From ABCD.EMP where FName = 'Ahsan Khan'

Deletion

Delete From ABCD.EMP where ID = 1045

Scanned with CamScanner

Java.sql.Connection
Java.sql.DriverManager



Connect String → Display name.

```
Action Performed
{
int x = Integer.parseInt(txtID.getText());
String y = txtName.getText();
Double z = Double.parseDouble(txtSalary.getText());

~~Connect~~
String cs = "~~~~~~~~~~~~~~~~~~~~";
String user = "abcd";
String pass = "1234";
try {
Connection con = DriverManager.getConnection(cs, user,
Statement st = con.createStatement();  pass);
String sql = "insert into ABCD.EMP values
                ("+x+","+y+","+z+")";
st.execute(sql);
con.close();
}
catch (Exception e) { }
```

Scanned with CamScanner

Result Set

Result Set rs =

⟹ Methods

① rs.next();
② rs.previous();
③ rs.first();
④ rs.last();

Result set rs;

- Search btn

String cs = "~~~";
String u.s = "~~~";
String pass = "~~~";

Connection con = Dri~~
Statement st = ~~~;

String sql = "~~slct~~ select * from abcd.emp";

rs=st.executeQuery(sql);
~~con.close();~~ con.setAutoComit(false);
rs.next();
int id = rs.getInt("ID");
String name = rs.getString("FName");
double salary = rs.getDouble("Salary");
txtID1.setText(id.toString());
txtName1.setText(name);
txtSalary1.setText(sal.toString());

Copy into next btn

Scanned with CamScanner