OOPS

- Object Oriented Programming

- Classes & Objects:
  - objects are entities in the real world.
  - Class is like a blueprint of these entities.

Syntax

```
class name{
    // properties and methods;

};
```

Example:-

```
Class Teacher{
    // Properties / Attributes
    string name;
    string dept;
    string subject;
    double salary;
```

```
    // Methode or member function
    Void changedept [string newdept]{
        dept = newdept ;
    }
};
Int main() {
    // create object
    Teacher t1;
    // access or assign data
    t1. name = "Umais";
    t1. Subject = "C++";
    // access methode
    t1. changedept = "on";
}
```

- Access Modifiers
  1) Private :
     data & methods accessible inside class
  2) Public :
     data & methods accessible to every one.

3) Protected :
data & method accessible inside class & to its derived class

i. You can write them in your class by writting names and ' : '.
: We can write all of that in a single class

- Encapsulation
is wrapping up of data & member functions in a single unit called class.
helps in data hidding.

- Constructor
Special methode invoked automatically at time of object creation. Used for intialisation.
  - Same name as class
  - Constructor doesn't have a

return type.

- only called once (automatically), at object creation
- Memory allocation happens when constructor is called

Create constructor:-

```
class teacher{
Public:
// constructor ← Head ()
    cout << "Hi i am
        constructor \n";
}
```

∴ So basically do not contain any type.

Initialize value

```
Headdept ()
    dept = "computer";
}
```

∴ Constructor always declare public

Constructor

non-parameterized Parameterized Copy

1) Non-parameterized

```
name (){
    // statements;
}
```

2) Parameterized

```
Teacher(string n, string d, string s){
    name = n;
    dept = d;
    salary = s;
}
```

- this
  is a special pointer in C++ that points to the current object.

Syntax

```
this -> name = name;
```

∴ automatically created pointer

this -> prop is same as *(this).prop

3) Copy constructor
special constructor (default) used to copy properties of one object into another.

Example:

```
Teacher t2(t1);
// call the function
t2.getInfo();
```

Own Creation

```
Teacher(Teacher &orgObj){
    this->name = orgObj.name;
    //          //
    //          //
}
```

Copy
shallow ← Copy → Deep Copy

a) Shallow
A shallow copy of an

object copies all of the member values from one ~~subject~~ object to another.

b) deep

A deep copy, on the other hand, not only copies the member values but also makes copies of any dynamically allocated memory that the members point to.

∴ Memory allocation types
stack, heap

∴ shallow copy create problems when we allocate memory dynamically.

. New

New is used to assign new type

Syntax

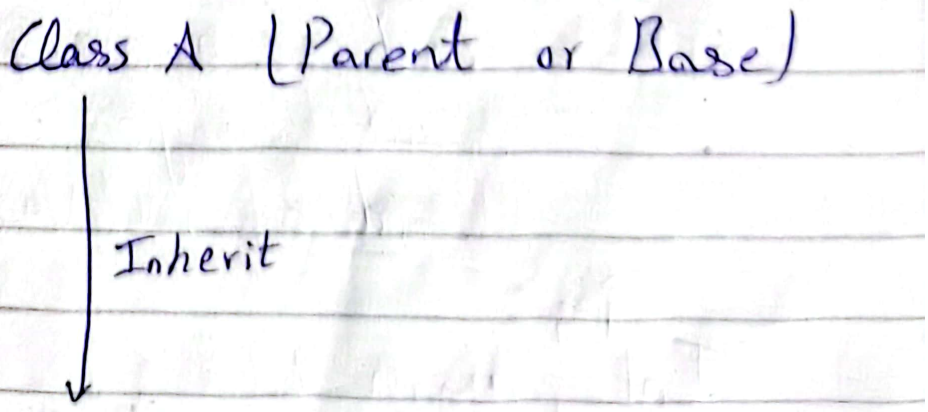
new float

- Destructor
  : destructor are importantly use to deallocate the new type.

Syntanx...

```
Same syntax
~Student(){  //own creation
    cout<<"delete everything\n";
    delete name;
}
```

- Inheritance
  when properties & member functions of base class are passed on to the derived class.



Syntax

```
class student : Public Parent{
Public:
    int rollno;
}
```

Example

```
class Person {
Public:
    string name;
    int age;
};
class student : Public Person{
Public:
    int rollNo;
    void getInfo(){
        cout << name;
        cout << age;
        cout << rollNo;
    };
Int main(){
    student t1;
    t1.name = "Umais";
```

```
t1.age = 21;
t1.rollNo = 455;
getInfo();
```

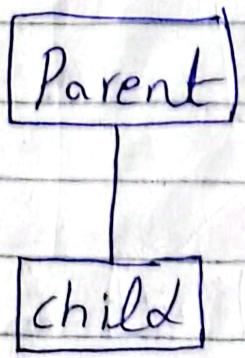
∴ First base constructor call then the drived constructor call atomatically

∴ Destructor can work opposite to constructor

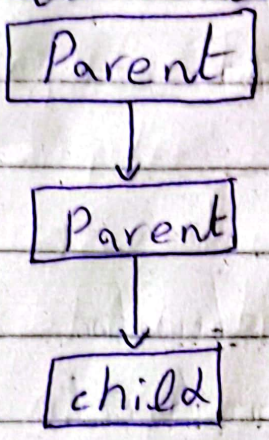
- Mode of inheritance
  See video image.

- Type of Inheritance

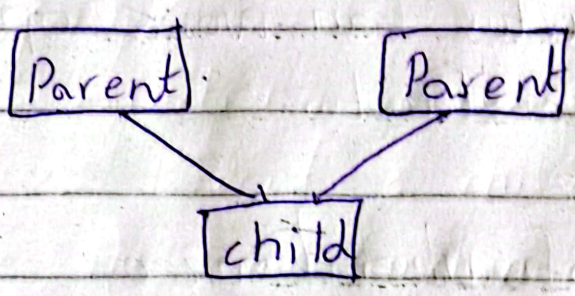
1) Single inheritance



2) Multi-level inheritance



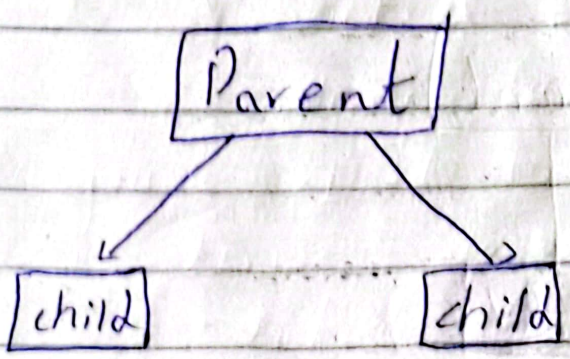
3) Multiple inheritance



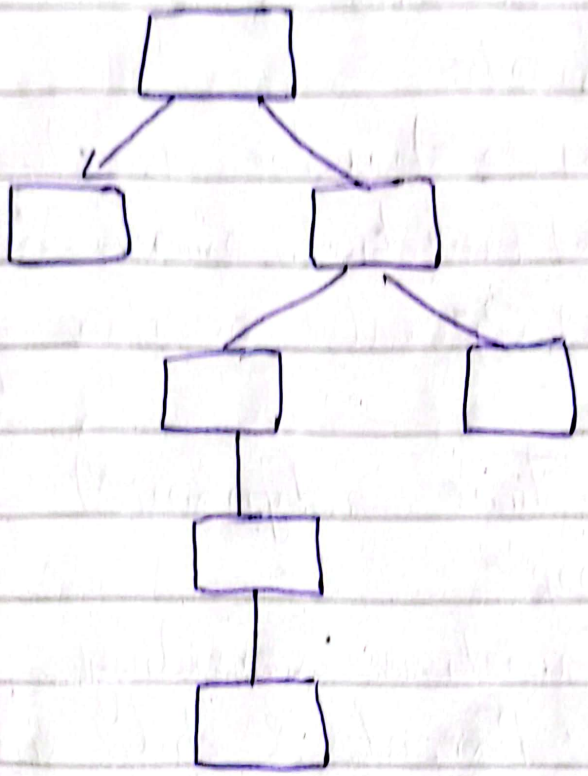
Syntax:-

Class name : Public Teacher, Public
student

4) Hierarchial inheritance



5) Hybrid inheritance



- Polymorphism
  is the ability of an objects to take on different forms or behave in different ways depending on the context in which they are used.

  Type
  - Compile Time
  - Run-Time

  Example
  constructor overloading

1) Compile Time

Example:

a) Constructor overloading

b) Function overloading

c) Operator overloading

2) Run-Time (dynamic)

Example

a) Function overriding

Parent & child both contain the same function with different implementation.

The parent class function is said to be overridden.

b) Virtual Functions

A virtual function is a member function that you expect to be redefined in derived classes.

- Virtual functions are dynamic in nature.

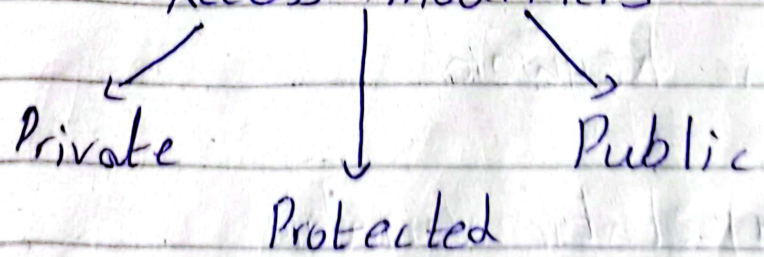
- defined by the keyword "virtual" inside a class base and are always declared with a base class and overridden in a child class.
- A virtual function is called during runtime.

- Abstraction

Hiding all unnecessary details & showing only the important parts.

One of that is

Access modifiers



One more way using abstract class:

- Abstract classes are used to provide a base class from which other classes can be drived.

objects

- They cannot be instantiated and are meant to be inherited.
- Abstract classes are typically used to define an interface for drived classes

Syntax

```
abstract class A {

};
```

∴ They do not create any class, they are the blue print of other classes.
It is use only for inherit.

- Static keyword

1) static varinble

Variables declared as static in a function are created & intialised once for the lifetime of the program.

Syntax

```
static int x=0;
```

Static variables in a class are created & intialised once. They are shared by all the objects of the class.

∴ If we change value, they will change in all

2) Static objects

∴ lifetime of program

Homework :-

Friend function and friend class.

Code with harry

- File handling

#include <fstream>

The useful classes for working with files in c++ are

1) fstreambase
2) ifstream
3) ofstream

Create a file with some name .txt

```
Int main(){
    ofstream myfile("Filename");
    myfile << "Hello";

// For read
    ifstream myfile("filenam")
    myfile >> str;

// get all data from file
    while(myfile.eof()==0){
        getline(myfile, str);
    myfile.close();
```

```
#include<iostream>
#include<ofstream>
int main(){
    string data;
    ofstream filename("C:\Http\desh
    \filename");
    File name.open
```

Again

Setter :- ← In class

```
Void getsalary (double s) {
    salary = s
}
```

Getter :-

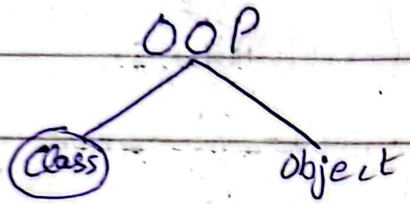
~~double Void getsalary (double) {~~
~~return salary;~~
~~}~~

```
double getsalary () {
    return salary;
}
```

Uni-Data
(OOP)

1) Class:-

A class is a block or templete or blueprint that defines a properties and behaviour of an object.



→ Blue print
↳ Templete
→ Properties → Variables
→ Behaviour

- Properties / Attributes / Data members / variab
- Behaviour / functions / member functions

Class → stack → non-physical
object → dynamic → physical

Programming

Procedural → C, C++

Object-oriented → C++, java

→ Algorithms
→ Divide into smaller called function
→ functions share global
→ Data move function to function

OOP :-
→ Emphasis data
→ divide int objects
→

Defination of
- Class
- Object
- OOPs

Difference

Java	C++
program.java	program.cpp
→ program.class	↓
→ program.exe	program.exe

website "oracle".

- Enviroments:-

JDK:

JRE:

JVM:

IDE:

API:

JAVAC:

Package: java.lang

Syntax:

Public class name {

```
Public static void main(String[]
                        args)
{
    System.out.print("First
                    java");
}
}
```

(2 hours)

"Java"

- Syntax

```
Public class name {
    Public static void main(strin
                    [] args) {

    }
}
```

- For output

```
System.out.println("Hello world!");
```

Shortcut

sout + tab

- Variables:

- Types

Primitive Non-Primitive/Reference

1) Primitive
   - byte -> 1 [-128 to 127]

- short → 2
- int → 4
- long → 8
- float → 4
- double → 8
- char → 2
- boolean → 1
- dauble..

- How to intialize:-

```
byte age= 30;
int phone = 123456789;
long phone2 = 12345678900L;
float pi = 3.14F;
char letter = '@';
boolean isadiult = false;
```

2) Non-Primitive

- string

In string to check length

* system.out.println(name.length());

- New

String name - New string ("value");

-> Strings

- check value at index

System.out.println("name.charAt(0));

Use charAt(0)

- Replace

string name2 = name.replace('a', 'b

System.out.println(name2);

∴ not change original string

- Substring

string name = "Aman and Ak

System.out.println(name.substring(2, 5);

∴ First include, last not include

∴ strings are immutible in java

-> Arrays

Syntax

int[] name = new int[3];

Intialized values

marks[0] = 97;

marks[1] = 98;

marks[2] = 99;

∴ When we do not intialize value in java then the variable or array is autoratially intialized with zero or null value that the difference in c++ and java. In case of c++, they contain garbage value.

- Print length of array
  system.out.println(name.length);

- Sorting
  Arrays sorting
  First
  Import a package
  import java.util.Arrays;
  Then
  Arrays.sort(name);

- Direct intialized:-
  int[] name = {97, 98, 95};

→ 2D Arrays

Syntax

```
int [][] name = {{97, 98, 99},
                 {66, 75, 92}};
System.out.println(name[0][0]);
```

→ Casting

Implicit Explicit

Example of explicit

```
int p = 100;
int p2 = p + 18.0   ← error
int p2 = p + (int)18.0;
system.out.println(P2);
```

→ Constants

Syntax

```
final float PI = 3.14F;
```

→ Operators

- Arithmetic
- Assignment

- logical
- Comparison

1) Arithmetic

+ , - , * , / , %

2) Assignment

= , += , -= , *= , /= , %=

3) Increment & decrement

num++ , ++num

num-- , --num

→ Maths

- Max & Min

```
System.out.println(Math.max(5, 6));
System.out.println(Math.min(5, 6));
```

- Random

```
System.out.println(Math.random());
    // Print in form of long
// automatically provide random values
System.out.println((int)(Math.random()*100));
```

→ Input in java

Syntax

Scanner name = new scanner(system.

First import package

import java.util.scanner;

then

Scanner name = new ~~string~~scanner(syste-

- store in variable

system.out.println("Input age");

int age = sc.nextInt();

system.out.println(age);

- If store in other type

~~int~~ float age = sc.nextFloat();

- To store in a string

string name = sc.next();

∴ It just store a single word

For full line:

string name = sc.nextLine();

4) Comparison Operator
==, !=, <, >, <=, >=

→ Conditional statements:
- if ( ) ... : syntax same
- if else ...
- if - else if
- switch

5) logical operators:-
&&, ||, !

→ loops:-
- for loop
  for(int i=0; i<100; i++) ... i=i+1
  for(int i=100; i>0; i--) ... i=i-1
- while
  int i=100;
  while(i>=1){
  System.out.println(i);
  i=i-1
  }
- do-while loop
  int k=100;

```
do {
    sy
    k = k - 1;
} while (i >= 1);
```

→ Break & continue

Example of break

```
int i = 0
while (true) {        ← Infinite
    system.out.println(i);
    i = i + 1
    if (i > 5) {
        break;
    }
```

Example of continue;

```
int i = 0;
while (true) {
    if (i == 3) {
        i = i + 1;
        continue;
    }
    system.out.println(i);
```

```
        i=i+1;
        if(i>5){
            break;
        }
    }
```

→ Expection handling

∴ Expection can be catch it does not create disturbance to run other code

∴ In case of error the code will not run but in expection the other blocks or properties will work.

∴ we can handle expection by keywords try or catch

Syntax:

```
try {
    // those statement which are
    expection exception or error
} catch (Exception exception){
    // this is class
```

```
}
  statement
    " "  ;
```

Example:

```
int[]marks = {97, 98, 95};
try {
   system.out.println(marks[5]
} catch (Exception exception){
    // do something after
}
System.out.println("Umais");
```

→ Methods / Functions

Syntax

```
Public class Main {
   Public static Fname () {

   }
   Public static void main(
                String[] args){
      Fname();
   }
}
```

→ Function of Parameter

```
Public static void Printname (String name) {
    system.out.print (name);
}
public static void main (String [] args){
    Printname ("umais");
}
```

"Series"

Java

- Java Installation
  1) JDk
     Java development kit
     ↳ contain tools
  2) Code Editor / IDE
     - ↳ VS code
     - ↳ Intellij
     - ↳ Eclipse

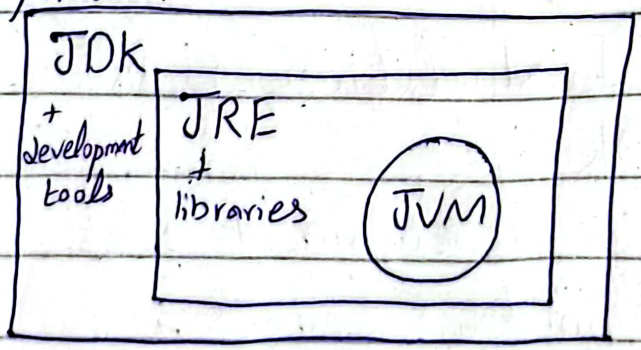
- First Code In VS-code

```
Class FirstClass {
    Public static void main(String[]
        system.out.println("Hello world");
    }
}
```

i. Class name is same as file name that saved in computer.

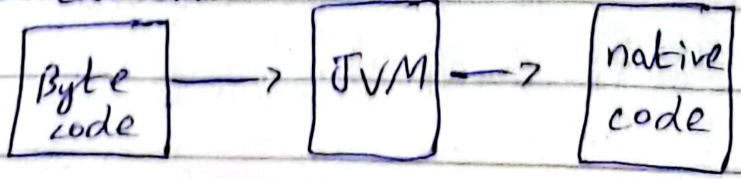
- How is code running?

a) Compilation



.java .class

b) Execution



- Output

Syntax

System.out.print( );

For next line

use \n

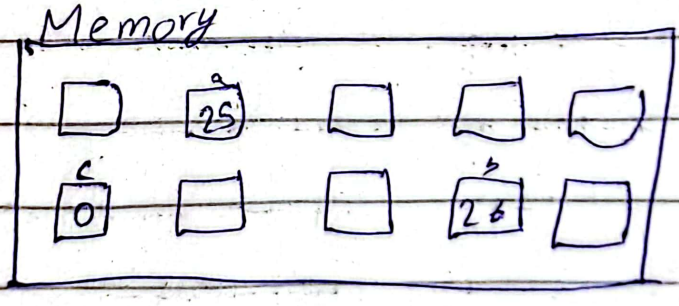
system.out.println( );

or

use \n in data

System.out.print("Hello world \n");

- Variables



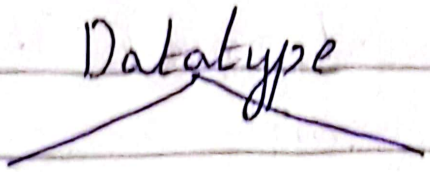
Declare:

datatype name;

Intialize

datatype name = value;

- Datatypes



Primitive	Non-Primitive
byte	string
short	Array
char	class
boolean	Object
int	Interface
long	
float	
double	

- Input

First

```
import java.util.*;
```

Now use input

```
scanner name =new scanner
                (system.in);
string name = name.next();
system.out.println(name);
```

In string for input complete line use nextLine();

∴ More type of data input

- nextInt();
- nextFloat();
- etc

OOPs in Java

- Object

```
Student s1 = new Student();
```

`new` → allocate memory for object

`Student()` → constructor to construct object

- Getter or Setter

```
Public class student {
Private: string name;
         string fname;
         int rollNo;
Public:
    Public void setname (string name){
        this → name = name;
```

```
Public string getname(){
        return name;
    }
// f name
Public void setfName(string fnam
        this -> fname = fname;
    }
Public string getfName(){
        return fname;
    }
```

Inheritance

```
Class A extends B {

    }
```

Input In Java

```
public static void main(string [] args)
Scanner input = new scanner (system.
boolean a = input.nextboolean();
System.out.println(a);
}
```

→ Abstract methode

```
Public abstract void NewDisplay();
Public void display(){ → non-Abstract
    sout "Hello world";
}
}
```

∴ Abstract method do not have ~~object~~ body and the abstract class do not contain object. we are not able to create object of abstract class.

∴ Can a abstract methode ~~diffi~~ define multitime in one class?

Accounting Fundamental

Rules of Dr. Cr.

	Dr.	Cr.
Assets	+	-
Exp	+	
Dividends	+	