OOP

Levels of Languages:..

1. Machine language
2. Assembly language
3. Higher level language.

Object oriented Procedural:-

Readability
Understanding
Maintenance → SMP

→change in software is called software maintenance.

Maintenance → Remove Bug.
→ update mean add new feature.
→ enhance perfomance | speed fast
→ convert windows to linux etc.

Readability and understanding is important to any change in the software is easy.

→ The programe of 5000 lines are divide into Chunks to Separate all the features.

Chunk means ke code ke har function ko separate kia jae. e.g:- CUI portal) attendance, faculty, courses, etc

Har kam assay aik procedure se hoga means ke procedural language.

Simply means ke function me divide kia gia ke take code ki readabity, understanding or maintenance easy ho sake.

Properties / Attributes Behaviour / Method

- Real world programming come into programming.
- Properties and behaviour are combine to make classes.

Properties and behaviour means ke har cheez ka ik behaviour hai or us ki kuch properties hai → e.g:- fixed chair and moveable chair.

Sailent Features of OOP:- 3MP.

→ Abstraction
→ Encapsulation
→ Inheritence
→ Polymorphism

→ These are the three main features.

Encapsulation: / Information hiding.

Means data members ko class me chupa dia jia hai, means ke koi dosri class access nhi kr sakta jab tak khud allow na kre.

Inheritence:-

Classes ko inherit krna. e.g:- ke already features kesi class me hai to us parent class ko child class ke sath inherit krna.

Class
Data Members Properties variables
Behaviour Function / Method

Abstraction:- Normally feature nhi hai, Lekin class itself ik abstraction

Polymorphism:-

Behaviour different hoga means ke us ka behaviour different hai.

Class Assessment:-

create function to add two numbers.

```
int add ( int a, int b) {
    a= 5;
    b= 6;
  a= a+b ;
  return  a;
  }
```

OOP

Q:- Write a function to add two numbers.

```
int add(int a, int b){        → Prototype (Signature
    int sum = a+b;
    return sum;
    }
int main(){
cout<<add(5,6);
    return 0;
    }
```

Q:- Write a function of square number.

```
int square (int a){
    int c = a*a;
     return c;
    }
int main(){
cout<<"The square root of "<< square(5);
    return 0;
    }
```

Q:- Pointers, Call by value, Call by reference, dereference.

```
void swap(int a, int b){
    int c = a;
    a = b;
    b = c;
}
  int main(){
int x=20;                    cout<<p;
int y=40;                    cout<<*p;
int *p = &x;                 cout<<&p; return 0; }
```

Q:- Write a programme to assign grade to the student.

```
#include <iostream>
using namespace std;
void grade (float a) {
    if (a > 70) {
    cout "The student have secured A grade";
    }
  else if (a > 50){
   cout" The student have secured B grade";
    }
  else
    {
     cout" Fail";
    }
  }
int main() {
  cout" Enter your marks to find grade \n";
    float x;
    cin >> x;
    grade (x);
  return 0;
  }
```

OOP

Two types of high level language:-

① Procedural language
② Object Oriented Programming language.

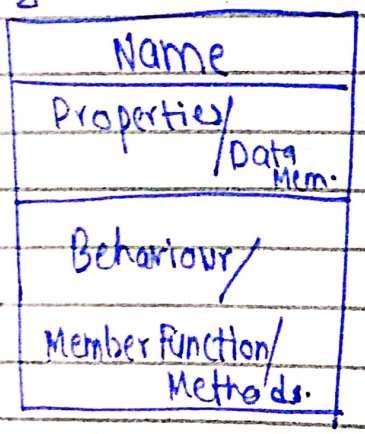
1. In procedural language programme divide into chunks and functions. Execution in variable form.

2. In object oriented Programme the variables are sum up into classes and objects.

OOP Features:

→ Encapsulation
→ ~~Ess~~ Polymorphism
→ Inheritence.

Classes → objects



: Properties
: Behaviour.

These are following have ability to form class objects.

① Person → students, faculty
② Place → campus information
③ Things → Result card, Challan form, etc
④ Concepts → Course registration is in field of concept.

Marks are not itself a class. it is the attribute of the class. (student, result) etc.

Basic syntax of java:

keyword ↑ class name ↑

```
Class Book
{
    datatype attribute/Property name;
```

simple syntax to declare class member.

```
Private String title;
Public int price;

Public static void main (string a[]){
    Book b1 = new Book();
```

Book → classname; b1 → object/instance; new → keyword; Book() → constructor

```
Private
(X) b1.title = "Programming";
    b1.Price = 5000;
for out display
→ system.out.println = (b1.title);

}
}
```

"Note:- Encapsulation is the process for the access specifier. (Public and Private, Protected). Before class member data type. "

Private : Not access directly.
Public : Access directly

Class itself simply definition it cannot work until its objects are created.

→ Every programme must have one executable class (Main class)

```
Class Abdullah {
    Public static void main (String argx[]){
```

Public → for access; void → Return type; main → name; String type array for main

```
}
}
```

CSC 241

Constructors are the function of the class name.

Class Practice.

```
Class Practice {

    Public String Crtitle ;
    Public string Name ;
}
    Class Course {
Public static void main (string a[]) {

    Practice P1 = new Practice("Programming",
    "CSC241");

    System.out.Println(P1.Crtitle);
    system.out.Println(P1.Name);

Practice P2 = new Practice();
P2.Crtitle = "OOP";
P2.Name = "CSC241";

system.out.Println(P2.Crtitle);
system.out.Println(P2.Name);

    }

    }

    Practic P1 = new Practice();
    P1.Practice = ("OOP"; "C
```

Constructor can define for passing arguments

```
class Book {

    Public String title;
    public int Price;

    Public void display() {

        system.out.print ln( title);
        system.out.print ln (Price);

    }
}
```

```
Class Book {
Private String title;
Private int price;
    void display(){ → it is the function it directly the access the member of the class no need to write public/private.
    System.out.println(title);
    system.out.println(price);
    }
}
Class Abdullah {
Public static void main(string args[]){
    Book b1 = new Book();
    b1.title= "Programming";
    b1.prive= 500;
    b1.display();

}
}
```

Note:- Set or ~~Get~~ Get function:.

when the class member of the class are private you can access the private member of the class by creating the function. in the class.

We use set and get function.

↓ for i/p: ↓ for ~~read~~ output

```
void setTitle(String a)
{
    title = a;
}

String getTitle( )
    return title;
}

b1.settitle("Programming");

system.out.Println(b1.gettitle());

Void SetPrice(int b)
{   Price=b;
}
int getPrice( )
{ return price;
  }

b1.setPrice(500);
System.out.Println(b1.getPrice());

b1.display();
```

```
Class Course {

    Private String CrTitle;
    Private string CrName;

    Void SetCrTitle (String a){

        CrTitle = a;
    }
    Void Set CrName (String b) {

        CrName = b;
    }
    String getCrTitle ( ) {

        return CrTitle;
    }
    String getCrName( ) {

        return CrName;
    }
}

Class Abdullah {

Public static void main(String args[])
{
    Course C1 = new Course( );
    C1.setCrTitle("OOP Pro");
    C1.Set CrName("CSC 241");
```

```
System.out.Println(C1.getCrTitle());
System.out.println(C1.getCrName());
    }
}
```

output:
OOP Pro
CSC 242

When you not give the value it give the default value to the data members. It called the default constructor.

e.g:-

b1.display(); → Null → for title
0 → for price

Constructor:

- It purpose was the initialize the data members of an object.
- Constructor is a function having same name as class.
- It has no return type.
- No argument / Multiple Argument.

```
Class Book {

    String title;
    int price;
    // constructor same name as class.
    Book( → No argument/Parameter.
    {   title="ABC";
        price=100;
    }
}
```

→ It is constructor it give the value to the default. so, the value of null is replace to ABC and 0 to 100.

```
Book (String a , int b)
{
    title = a;
    Price = b;
}
}
```

↓ main class

```
Book b2 = new Book("OOP", 200);
b2.display();
```

OOP
500

Practice Session.

```
Class Course {

Private string crName ;
Private string Crcode ;

 void setCrName (String a) {
     CrName = a;
 }
Void setCrcode (String b){
     Crcode = b;
 }
String getCrName( ){
     return CrName;
 }
```

```
string getCrcode(){
    return crcode;
}

Book(string c, string d){
    crName=c;
    crcode=d;
}
void display()
{ system.out.println(crName);
    system.out.println(crcode);
}
}
Class Abdullah
{
public static void main(string args[])
{
    course c1 =new course();
    c1.setCrName("OOP");
    c1.setCrcode("CSC241");

system.out.println(c1.getCrName());
system.out.println(c1.getCrcode());

    course c2 = new course("ABC","csc241");
    course c3= new course();
    c2.display();
    c3.display();
}

}
```

OOP

Aarrays:-

```
int a[5];
int b[] = {20,21,30,45}
```

Array [index]

b[2] ⟶ 30

C++

Java.

```
int a[];
int [] a;
int   []a;
```

→ These are three posibilities.

```
int a[] = new int[5];   → to declare array.
int b[] = {2,4,6,8,9};
```

```
a[0]=40;
a[1]=60;
a[2]=50;
a[3]=100;
a[4]=200;
```

```
for(int i=0; i<5;i++){
cout<<a[i];
}
```

array.length → to find the size of array.

Array of class:-

```
Book a[] = new Book[5];
```

Book ↑ name of the class; a ↓ array; [5] ↓ 5 elements.

```
a[0]= new Book();
a[1]= new Book();
a[2]= new Book("Asad", Programming");
a[3]= new Book();
a[4]= new Book();
```

↑ Argument constructor.

we use new Book(); in array to create object and and assign object to a[], index. because we create object of class like this.

```
Book b1 = new Book();
for(int i=0 ; i<5 ; i++) {
    sout(a[i].display());
a[2].title = "Ahmedabad";
a[2].author = "Zain";
```

→ To change the value of the data for public.

Copy Constructor:-

```
Course(Course a){
code = a.code;
title = a.title;
}
```

The copy of one object to the other.

```
Course c1 = new Course();      → D.C
Course c2 = new Course("OOP", "Anjuad");  → P.C
Course c3 = new Course(c2);    → C.C
```

OOP

Object as Data Members
Static Data Members
Static Member Functions

```
Class page {
    String type;
    int weight;
}

Class Book {
    String name;
    String author;
    Page pages;  → Class as data member/ object as data member.
    Book ( ) {  → construct (no argument)
        name = "ABC";
        author = "XYZ";
        pages = new page( );  → Creating object.
        pages.type = "A4";   } → initilze the value to datamemberss as object.
        pages.weight = 5;
}

Book(String a, String b, String c, int d) {
        name = a;                  → Argument constructor.
        author = b;
        pages = new Page( );  → create object
        pages.type = c;
        pages.weight = d;
}
```

Note:- This concept ~~has a~~ is called a "HAS A RELATIONSHIP", in which the data is CONTAIN.

```
Public void display(){
    S.O.P(title);
    S.O.P(author);
    S.O.P(page.type);
    S.O.P(page.weight);
```

NOTE:- In inheritence or a child parent relation is called "IS A RELATIONSHIP".

```
Class xyz {
    P.S.V main( )
{   Page P= new Page();
    Book B1 = new Book();
        b1.display();
Book b2. new Book("OOP","Ali","A4", 8);
        b2.display();
```

Static Keyword:-

Static Data Member
Member Function

Non-Static Data Member
Member Function

• static Datamember is not copy to other object it has only one object and shared all the remaining object.

Non-static Data Members means ke har object me copy jae gi as a part of object.

```
Class Page {
    String type;
    int weight;
Static int count = 0;        → Static data Member.
Public static void fun()     → Static data Function.
{
    S.O.P("Helloooo");
}
```

OOP

Encapsulation (Security)
Inheritance (Reuseability)
Polymorphism

Generalization
↓
Specialization

Parent / Super / Base → Child / Sub / Derived

IS A RELATIONSHIP:- In this process the child object is treat as a parent object.

Inheritence :-

```
class person {
    String name;
    int age;
person() {  → Constructor

    }

Person(String a, int b) { → Argument Constructor.
    }
Public void display() {
```

Child class

This keyword is used for inheritence.

```
class Student extends Person → Parent class.
{
// All data members and function are part of child class
    String Program;     // 4 data members.
    String reg.no;
```

// Create constructor of child class in two ways.

```
Student () {
  string
  name = "Hli";
  age = 22;
  Programe = "BSCS";
  reg. NO = 30;
}
```

```
Student( ) {
    Super ();
  Program = "BSCS";
  reg.no = 30;
}
```

→ The super keyword is used to call the constructor of the parent class.

```
Student (string a, int b, string c,
  string d)
{
  super (a, b);
  Programe = c;
  reg.no = d;
}
```

→ caling the Argument constructor of the parent class.

```
display ( ) {
  super. display ();
System.out.Println (Programe);
System.out.println (reg.no);
}
```

Java

```
Class Person {
        String name;
        String address;
        int age;

    Person() {
        name = "ABC";
        address: "Farid Town";
        age = 100;
    }
    Person (String name, String address, int age){
        this.name=name;
        this.address= address;
        this. age = age;
    }
    Public void display () {
        systen.out.println (name);
        System.out.println(address);
        system.out.println( age);
    }
}

Class Faculty extends Person {
        string designation;
        String Department;

    Faculty () {
        Super ();
        designation = "Lahore";
        Department= "BSCS";
    }
```

```
Faculty(String a, String b, int c, String d, String e){
    super(a,b,c);
    designation = d;
    Department = e;
}
Public void display(){
    super.display();
    system.out.println(designation);
    System.out.println(Department);
}
}
Public class ab{
Public static void main(String[] args){
    Faculty F1 = new Faculty();
    F1.display();
    Faculty F2 = new Faculty("XYZ", "AB", 12, "X", "2"
    F2.display();
}
}
```

Q:- This or super.

Q:

Overloading Function. It is the function in which same name but different argument.	Overriding. It is the function in which same name and same argument in the parent or child class.

~~Person~~

Polymorphism:- Same name but different output
Same item with different behaviour.

e.g:-

```
Person P = new Person();
   P.display();  → in this output 3 members.
Student st = new Student();
   st.display();
   P=st;
P.display(); → In this output 5 members.
                bcz object assign.
```

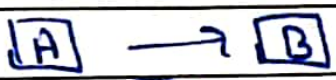
This is the concept of polymorphism.

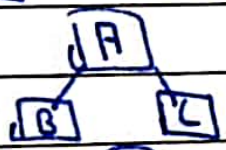
```
Class P{
   P.V.display()
   { S.O.P("Hello");
   }
   }
   Class C extends P {
     P.V. display(){
     S.O.P("Good Bye");
   }
   }
P.S.V main(){
   P p=new P();
   P.display();
  C c = new C();
   p=c;
  p.display();
  }
  }
```

Q:- Benefit of polymorphism.

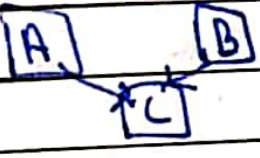
M T W T F S **OOP** DATE:

Protected Access
Types of inheritence
Abstract Class
Interface
4 type of Access specifier.
private → Same class (directly)
Public → Every where
Protected → child classes + Same package
Default → Same package

Types of inheritence:-
Single 

Hierarchical 

Multi level 

Multiple 

→ Java doesnot support multiple inheritence.

M T W T F S **OOP** DATE:

Abstraction:

Hide Details. | How
Share necessary | essential information | What

→ Abstract class is partial Abstraction, but interfaces is complete Abstraction.

Abstract class:-

- It cannot be instantiated | objects cannot be created.
- It purpose only use in inheritence, only subclasses can be create.
- Can have abstract methods.
  - methods without implementation.

Example:-

```
abstract class shape {
    int dimensions;
public abstract void area();
    Public void show() {
      S.o.p("No of dimensions" + dimensions);
    }
}
```

→ This is abstract method it cannot. Implemented.

↓ This is normal method.

→ The object can't be create of abstract class.

```
Shape s1 = new shape(); → // Error.
Class rectangle extends shape {
    int length;
    int width;
  rectangle() {
    length = 4;
    width = 2;
    dimension: 2;
  }
```

M T W T F S DATE:

```
Public void area(){
    Int c= length * width;
    S.O.P("Area is" +C);
  }
  }
class circle extends shape{
    int radius;
circle (){
  };
Public void area() {
    int c= 3.14 *radius * radius;
  S.O.P=("Area is" +C);
}
}
```

// The child class implement the parent class abstract function and parent class only declare the abstract method.

```
abstract class Bank {
        int Interest;
  public abstract void profit();
    }
class Allied. extends Bank {
        int amount;
    public void profit(){
```

It is contract between classes. they all follow the same contract (common item share same).

M T W T F S OOP. DATE:

```
interface engine {
    public void speed ();
    public void throttle ();
}
class honda implements engine {
    public void speed()
    { S.O.P ("speed changed by 10");
    }
    Public void throttle () {
        S.O.P ("Fael injection change by 5");
    }
}
Class Toyota implements engine {
    public void speed ()
    { S.O.P("speed changed by 15");
    }
Public void throttle () {
    S.O.P("Fuel changed by 6");
    }
}
```

Multiple Inheritence:-

```
class Corolla extends Toyota implements Engine.
```

Child class. Parent class Interface