Certificate

Name: Harshvardhan Purohit Class:

Roll No: Exam No:

Institution _____

This is certified to be the bonafide work of the student in the _____ *Laboratory during the academic year 20 / 20 .*

No. of practicals certified _____ *out of* _____ *in the subject of* _____

.....
Teacher In-charge

.....
Examiner's Signature

.....
Principal

Date: .....

Institution Rubber Stamp

(N.B: The candidate is expected to retain his/her journal till he/she passes in the subject.)

Index

S. No.	Name of the Experiment	Page No.	Date of Experiment	Date of Submission	Remarks
1	OOPS lab - 1				
2	OOPS lab - 2.				

Expt. No. _____ Date _____

Page No. _____

OOPS LAB 1

Theory

→ Basic concepts of Object oriented Programming.

Introduction

→ OOPS are introduced to overcome limitations in procedural programming language like reusability & maintainability.

Types of programming languages

- High level.
  - Object Oriented Programming
  - Procedural Programming
- Low level.
  - Assembly language
  - Machine language

High level languages (Only human can understand)

Low level languages (Only Machine can understand)

Teacher's Signature _____

Expt. No. _____ Page No. _____

Procedural languages (Eg: C, COBOL, FORTRAN)

- language deal with algorithm
- Program are divided into functions
- It used how Memory
- It is concept of top down Approach
- Overloading Not possible
- low secure

Object Oriented Programming (Eg: Java, C++, python)

- language deal with data
- Programs are divided into objects
- It used more than pop Memory
- It is concept of bottom up approach
- Overloading is possible
- Highly Secure

Objects → Self contained Component with

- Methods
- Properties
- Attributes

↓

It has Attribute Defining its Characteristics And Methods

Teacher's Signature _____

Expt. No. _____ Date _____

Page No. _____

Classes

- Used to refer to the categories or Types of Object.
- User-defined data-Type.
- It is an collection of Objects.
- It is just like an blue-print of object.
- This Data-Type has data members & Members function.

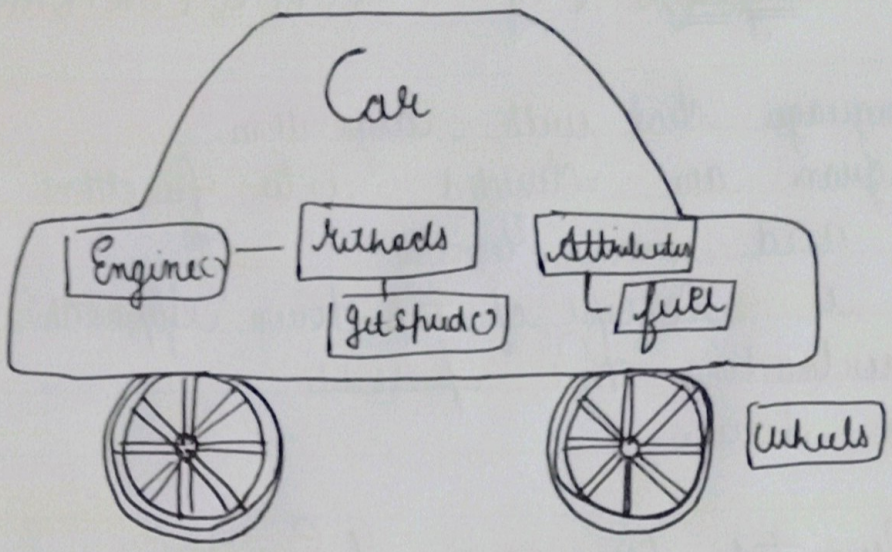
Eg

- In a class there are many Students
  - Students are objects
    - Students have common parameters
      - Name
      - Age
    - Only Values are different of different objects.
      - "Harsh" > Name
      - "Pranav" > Name
      - 10 > Age
      - 18 > Age

Note:

- Classes do not occupy any Memory location.
- Objects Occupies Memory location during Run Time.
- Objects are Necessary for accessing Classes.

Teacher's Signature _____


Car
Engine
Methods
Attributes
fuel
wheels

Expt. No. _____ Date _____

Page No. _____

4 Pillars of Object oriented Programming :->

- Data Abstraction
- Data Encapsulation
- Inheritance
- Polymorphism.

Data Abstraction

- Reduction of particular body of data to a simplified representation of whole.
- Removing characteristics (Unnecessary)
- Converting all the bulky characteristics into set of essential characteristics.
- For Example:
  - Suppose you are operating an Mobile phone, three features are :->
    - Memory Card.
    - SIM
    - Battery life
    - Design
    - Build & processor power.
  - You are only giving touch commands without knowing what's happening inside.

Teacher's Signature _____

Expt. No. _____ Date _____

Page No. _____

Data Encapsulation

- Enables the programmer to combine both
  - Data & Function that Operate under a single unit.
- It is way to implement data abstraction.
- Example of Data Encapsulation:
  - Suppose there are 2 Department in your Company:
    - Finance
    - Sales.
  - In Order to accu Sales data you will require an person from finance Sales section.
  - This is an Example of Encapsulat -ion
  - Here Data (Sales Data) & Sales person are wrapped under single Name "Sales section".

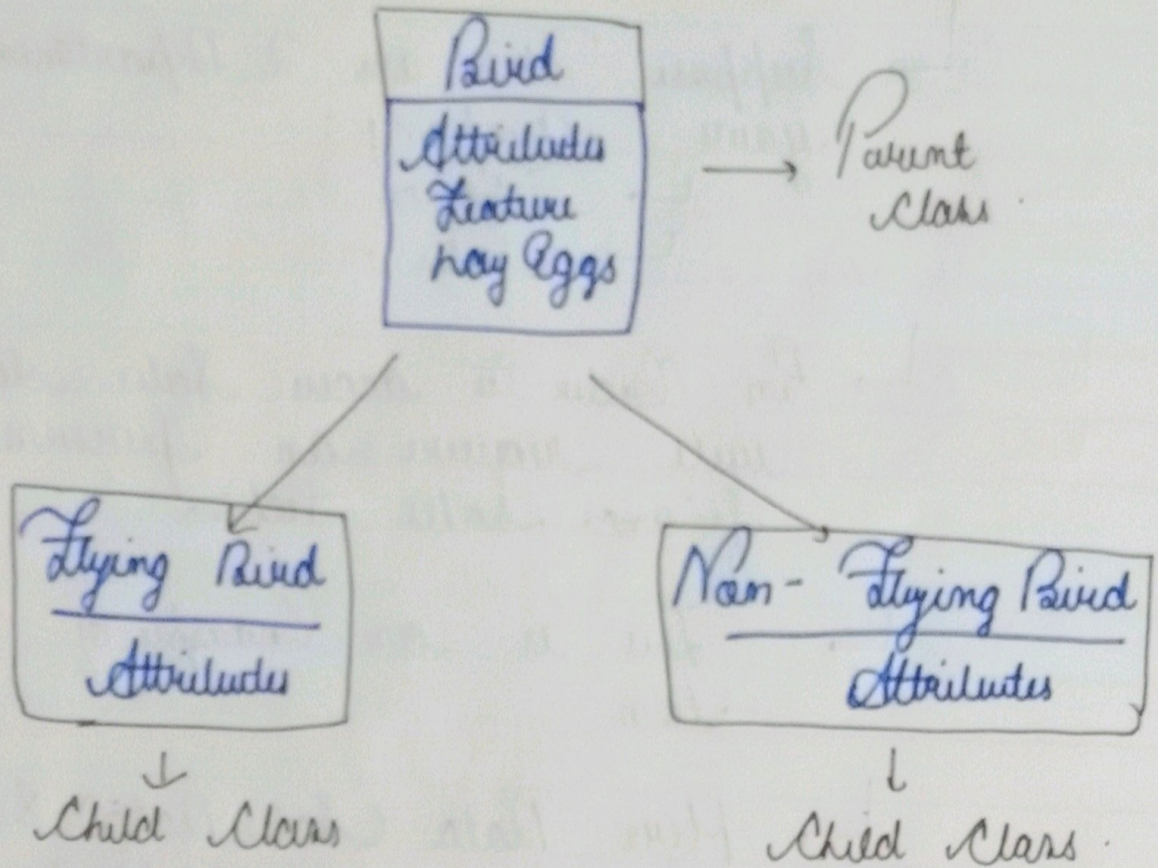
Teacher's Signature _____

↳ Encapsulation also leads to data abstraction or hiding.
↳ Using Encapsulation we can hide data.
↳ data of any of the section like:
  ↳ Sales ↳ finance ↳ Accounts
are hidden from any other section.

Inheritance

↳ Process by which objects of one class acquire the properties of objects of another class.
↳ It support concept of hierarchial classification.
↳ It supports idea of reusability.
↳ For Example:
  ↳ Bird is an class having some of the attributes.
  ↳ Suppose there are 2 more classes of flying bird & Non-flying bird.
  ↳ Now they both have (one different attributes) other attributes same as class bird.
  ↳ So bird is an parent class whose properties are inherited to flying bird & non flying bird classes.

Teacher's Signature

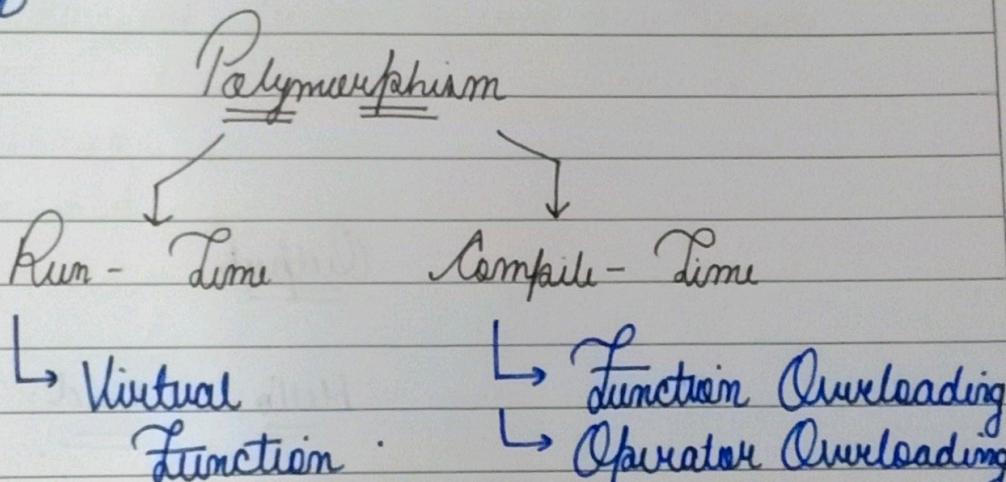

Bird
Attributes
Feature
Lay Eggs
Parent Class
Flying Bird
Attributes
Child Class
Non-Flying Bird
Attributes
Child Class

Expt. No. _____ Date _____

Page No. _____

Polymorphism

- Meaning in greek "Ability to take more than one form".
- For Example:
  - An Operation may exhibit different behaviours in different instances.
    - For 2 Numbers, + operator will add Numbers.
    - For 2 strings, + operator will concatinate strings.
- Ability to use same thing differently at different situation.



Teacher's Signature _____

Expt. No. _____ Date _____

Page No. _____

Function Overloading

↳ Using same function name but with different attributes.

Operator Overloading

↳ Process of making an operator to exhibit different behaviours in different instances is known as Operator Overloading.

Virtual Function

↳ Member function in base class that you redefine in derived class.

C++ Program without using Object & Classes

```
#include <iostream>
using namespace std;

int main()

{
    cout << "Hello world" << endl;
    return 0;
}
```

Teacher's Signature _____

Output

Hello world.

Date _____

Expt. No. _____ Page No. _____

OOPS LAB 2

Problem : Write a program to implement the concept of Class :->

```
#include <iostream>
using namespace std;

class person
{
  public:
    string name;
    int age;

    void display()
    {
      cout << "name is : " << name << endl;
      cout << "age is : " << age << endl;
    }
};

int main()
{
  person p1;
  cout << "Please enter your name : ";
  cin >> p1.name;
  cout << "Please enter your age : ";
  cin >> p1.age;
  cout << "\n";
```

Teacher's Signature _____

Output:

```
Please enter your name : harsh
Please enter your age : 89

name is : harsh
age is : 89

Thanks for using the name printer.
```

Expt. No. _____ Date _____

Page No. _____

```
cout << " name is : " << p1.name << endl;
cout << " age is : " << p1.age << endl;
cout << "\n";
p1.display();
cout << " Thanks for using name printer " << endl;
return 0;
}
```

Problem 2 : Write a Program using concept of class ?

```
#include <iostream>
using namespace std;
// This is version 1 of the name printer

class person
{
  public:
  string name;
  int age;

  void display()
  {
    int press;
    cout << " Please enter 1 to print name " << endl <<
         " Please enter 2 to exit " << endl;
    cin >> press;
    if (press == 1)
```

Teacher's Signature _____

Output

```
Please enter your name : harsh
Please enter your age : 89
Please enter 1 to print the name
Please enter 2 to exit
1
name is : harsh
age is : 89
Thanks for using the name printer.
```

Expt. No. _____ Date _____

Page No. _____

```
        {
            cout << " name is : " << name << endl;
            cout << " age is : " << age << endl;
        }
        else
        {
            return ;
        }
    }
};

int main()
{
    person p1;
    cout << "Please enter your name : ";
    cin >> p1.name;
    cout << " Please enter your age : ";
    cin >> p1.age;
    p1.display();

    cout << " Thanks for using name pointer" << endl;

    return 0;

}
```

Teacher's Signature _____

Expt. No. _____ Date _____

Page No. _____

Problem 3: Write a program to understand concept of public private acus specifier:

```
# include <iostream>
using namespace std;

class aging
{
    private:
      int age;

    public:
     void aging (int a)
     {
       age = a;
       cout << " The age of person is : " << age << endl;
     }

};pl;

int main()
{
    pl. aging (12);
     return 0;

}
```

**

Teacher's Signature _____

Output

The age of the person is : 12