Introduction to Object Oriented Programming:

x) Object - Oriented Programming:

OPP is a programming paradigm based upon objects having data and methods that aims to incorporate advantages of modularity and reuseability.

It is the programming pattern in which programs are structured around objects rather than functions or logic.

x) Concepts of OOP

The concepts of object-oriented programming are as follows:

(i): C++ class
(ii): C++ object
(iii) Encapsulation
(iv): Abstraction
(v): Inheritance
(vi): Polymorphism

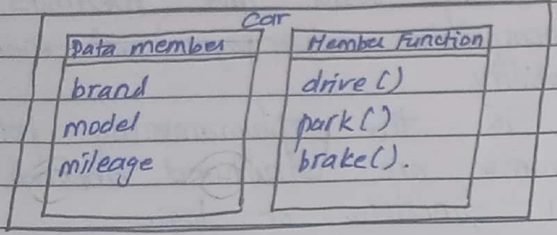
Date. No.

i): C++ class:

→ A class is the blueprint of object.

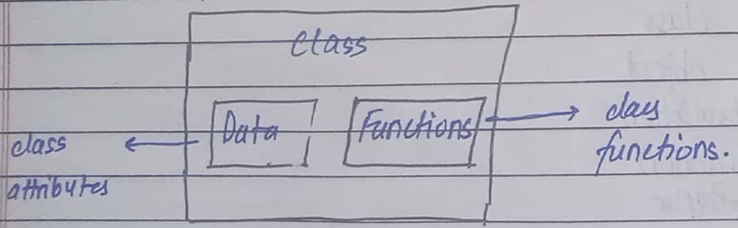
→ A class consists of data members and member functions.

Let us consider a class car.



Here, details of car ie, datamembers are the data and the functions in car ie, member functions are the functions in that class.

So,



Date. No.

ii) C++ objects:

Objects are the instances of class.

Eg:

The cars can be SUV, Van, Sedan, electric, etc.

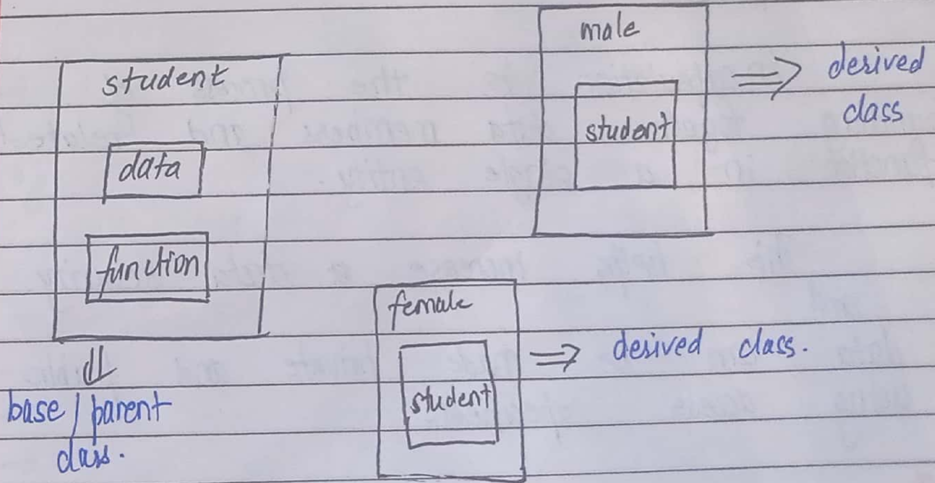
These SUV, Van, Sedan are called objects.

Objects has own data and function to manipulate data.

(iii) Inheritance:

Inheritance in C++ helps us to create a new class ie, derived class from a ~~b~~ base class. without modifying the existing class.

It helps in code reuseability and derived classes can have ~~attit~~ addition features too.



Date. No.

Inside male & female class, there is student class.

(iv): Abstraction:

Abstraction means only displaying the necessary information to the user and hiding complex details of program implementation and execution.

Unnecessary details can be hidden from the user.

<*>: Abstraction means ~~&~~ showing ~~Relative~~ relevant information while data hiding means restricting data. ~~Acces~~ access to outside class.

(v): Encapsulation:

Encapsulation is the process of bundling together data members and related function in a single entity.

This helps increase ~~#~~ data security and data can be made private and public using access specifiers.

Date. No.

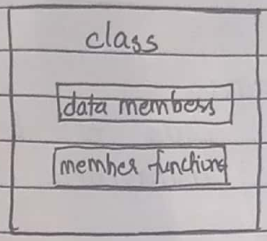


fig: C++ encapsulation.

(vi): Polymorphism:

Polymorphism is the ability to use a common function and operator in multiple ways.

It is obtained through function ~~overiding~~ overriding and overloading.

This helps us to use the same functions in different tasks.

Eg: A sum function can be used to add. numbers in different cases

→ int sum (int x, int y)
→ ~~int~~ float sum (int x, float y)
→ int sum (int x, int y, int z)
→ float sum (float x, float y).

Date. No.

(X): Benefits of OOP over POP:

i) Easier to maintain:
OOP allows code to be broken down into smaller, more manageable pieces.

ii) Code reuseability:
Objects and class can be used across program minimizing code duplication.

(iii): Readability:
OOP has easier to read and understand syntax than POP.

(iv): Data security:
Encapsulation and data hiding makes it difficult for external sources to modify program data enhancing data security.

(x): Drawbacks of OOP from POP:

(i): Overhead:
OOP programs need additional overheads like need to create and manage objects making execution slower than POP.

Date. No.

(ii): Complexity:
OOP has concepts of object and classes making difficulty in concepts for beginners.

(iii): Overuse of Inheritance:
Overuse of inheritance creates complex class hierarchies which are difficult to maintain and understand.

(iv): Memory management:
OOP requires more memory ~~manag~~ management than POP as there's increased chances of memory leaks and bugs.

(v): Low-level coding:
Writing low-level ~~#~~ codes using OOP is difficult.