Classes and Objects

x) Data encapsulation:
It is the process of combining both data members / attributes and the functions that operate on the data into a single unit.

x) Class: A description of a number of similar objects.

x) Data hiding:
Restricting data ~~user~~ access to the data to keep it safe from accidental alteration.
It is done using access specifiers.

x) Objects: Objects are the instances of class.

x) Access Specifier:
Access specifiers / modifiers define how members of a class can be accessed.

In C++, there are 3 types of access specifiers.

i) public: members are accessible from outside the class.
ii) private: members are accessible from inside the class only
iii) protected: members are inaccessible outside class but can be accessed by derived class / sub-class.

x) Syntax:

```
class class_name {
  access specifier:
      data_members;
      functions;
  access specifies:
      data_members;
      functions;
};
```

Eg:
```
class point {
  private:
      double x;
      double y;
  public:
      void set(int x, int y);
      double distance();
};
```

x) Declaring objects:
class_name object_name;
Eg: point p1;

x) Accessing objects:
object_name.membersname
Eg: p1.x

→ we may also have to use getter and setter.

X) C++ Separate header and Implementation files

In coding, we divide our code into many files to ~~prevent~~ ~~use~~ provide code clarity along with code reusability, reducing compilation time and implementation hiding.

Header file: It contains class declarations in separate file with .h extension.

Source file / Implementation file: It contains class function definitions in separate file with .cpp extension.

Client code: It contains the main() function in file with .cpp extension.

X) Abstraction:

Abstraction is the process of providing only essential information to outside world by hiding background details.

This process separates our code into interface and implementation.

Interface is independent component ~~and~~ not changed by implementation component.

Eg: Interface can be header file with class declarations and implementation can be source files containing class function definitions.

X) UML class diagram:

A class diagram gives an overview of a software application by presenting the classes and relations between them.

Point	→ class name
- x - y	→ Attributes
+ setPoints(double x, y:double) + distance(anotherPoint: Point)	→ Member functions

- : private
+ : public
# : protected

Here, -, +, # are access specifiers

X) Private member function:

A private member function cannot be accessed from outside the class.

There is no difference while defining a private member function.

A private member function can be called from another member function of the same class but can't be called from outside the class.

Date. No.

x) const members functions:

A const member function ~~prevents~~ guarantees it will not modify the object or call any non-const member functions.

It contains const in their declaration.

Syntax: datatype function_name(arguments) const;

x) Static data members:/member functions

Static data members and static member functions are shared by all objects of the class

Static members belong to the class itself and exist if no ~~class~~ objects have been instantiated.

Static members ~~variat~~ have lifetime throughout the program.

→ Accessing static members:

class-name :: static_member_variable

Static member function dont have 'this' pointer