OOPs

Q What is meant by OOPs?

→ It refers to Object Oriented Programming.
→ It is the programming paradigm that is defined using objects

Q What is need for OOPs?

→ It helps users to understand software easily
→ The readability, understandability and maintainbility of code increase
→ Written and managed easily.

Q. Major OOPs language
→ C++
→ Java
→ JS
→ Python
→ PhP

Q. Other programming pardigms other than OOPs?

→ Imperative Programming paradigm
→ Declearative " "

→ Imperative :-
It focus on how to execute program logic and defines control flow as statement that change a program state.

→ Declarative :-
It focus on what to execute and define Program logic but not detail control flow.

Q Structured Programming?

It refers to method of programming which consists of a completely structured control flow.

→ structure refers to a block, which contains a set of rules and definitive control flow, such as (if/then/else), (while and for)

* Features of OOPs?

→ Inheritance
→ Encapsulation
→ Polymorphism
→ Data Abstraction

* Advantages

→ It helpful in solving complex level of problems
→ Highly complex programs can be created, handled and maintained easily.
→ It promote code reuse, (reduce redundacy)
→ It helps to hide unnecessary details
→ Bottom-up approach based.

Q Why is OOPs so popular?

→ It is considered as a better style of programming. Not only it helps in writing a complex piece of code easily, but it allows also users to handle and maintane them easily.

Q What is a class?

→ It is blueprint, which contains some values known as member data and some set of rules, known as behaviors or functions

→ When an object is created, it automatically takes the data and functions that are defined in class.

→ Eg:- First, a car's template is created. Then multiple units of car are created on based of that template.

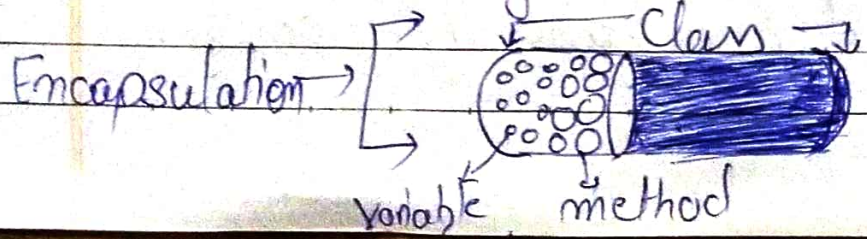
Q What is an object?

→ It is refers to the instance of class, which contains the instance of members and behaviors defind in class template.

→ In real world, It refers to an actual entity to which a user interacts, whereas class is just the blueprint for that object.

Q Encapsulation:-

→ It is the process of binding data members and methods of program together to do a specific job, ~~write~~ without revealing unnecessary details.



→ Two Different ways:

1. Data hiding: Encapsulation is the process of hiding unwanted information, such as restricting access to any member of an object.

2. Data binding: It is the process of binding the data members and methods together as a whole, as a class.

* Polymorphism:-

→ It composed of two words - "poly" means "many", and "morph" means "shapes".

→ It means to something that has many shapes

→ In OOPs, It refers to the process by which some data, code, method or object behave differently under different circumstances or contexts.

TYPES:

1. Compile time polymorphism:-

→ It is also known as static polymorphism, refers to type of polymorphism that happen at compile time. that is compiler decides what shapes or value has to be taken.

Compile time polymorphism:

Ex:

```
class CompileTimePolymorphism{
    public int add(int x, int y){
        return x+y;
    }

    public int add(int x, int y, int z){
        return x+y+z;
    }

    public int add(double x, int y){
        return (int) x+y;
    }

    public int add(int x, double y){
        return x+(int)y;
    }
}
class Test{
    public static void main(String[] args){
        CompileTimePolymorphism demo =
            new CompileTimePolymorphism();
        System.out.println(demo.add(2,3));
        System.out.println(demo.add(2,3,4));
        System.out.println(demo.add(2,3.4));
        System.out.println(demo.add(2.5,3));}}
```

Runtime Polymorphism:

→ It is also known as dynamic poly.
→ It refers to type of polymorphism that happen at run time.
→ It means it can't be decided by compiler.
→ The shape and value has to be taken depends upon execution.

Ex:-

```
class AnyVehicle {
    public void move() {
        System.out.println("Any vehicle should move");
    }
}
class Bike extends AnyVehicle {
    public void move() {
        System.out.println("Bike can move too");
    }
}
class Test {
    public static void main(String[] args) {
        AnyVehicle vehicle = new Bike();

        vehicle = new AnyVehicle();

        vehicle.move();
    }
}
```

* What is meant by Inheritance?

→ The term "inheritance" means "receiving some quality or behavior of a parent to an offspring."

→ It is mechanism by which an object or class (referred to as a child) is created using the definition of another object or class (referred to as a parent).

→ Inheritance not only helps to ~~keep~~ ~~help~~ keep the implementation simpler but also helps to facilitate code reuse.

* Abstraction

→ It is method of hiding unnecessary details from the necessary ones.

→ Ex:- Consider a car. You only need to know how to run a car, and not how the wires are connected inside it.

Q. Is it always necessary to create objects from class?

No, An object is necessary to be created if the base class has non-static methods.

But if the class has static methods, then object don't need to be created.

Q What is a Constructor?

→ It is a method used to initialize the state of an object and it get revoked at the time of object creation.

Rules:-
* Constructor Name should be same as class Name
* Constructor must have no return type.

TYPES of Constructor:-

i. Default Constructor :- Doesn't take any argument.

```
class ABC
{
    int x;
    ABC()
    {
        x=0;
    }
}
```

ii. Parameterized Constructor :- Take some argument

```
class ABC
{
    int x;
    ABC(int y)
    {
        x=y;
    }
}
```

iii. Copy Constructor:

→ It is a member function that initializes an object using another object of same class.

→ i.e. clone an object and its values

```
class ABC
{
    int x;
    ABC(int y)
    {
        x = y;
    }

    ABC(ABC abc)
    {
        x = abc.x;
    }
}
```

Q. What is Destructor?

→ It free up the resources and memory occupied by an object.

→ It automatically called when an object is being destroyed.

* Limitation of Inheritance

i) It needs more time to be process, as it navigate through multiple classes of its implementation.

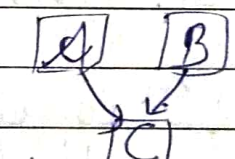
ii) The classes involved in Inheritance - the base class and child class, are very tightly coupled together. So if one needs to make some changes, they might need to do nested changes in both classes.

iii) It is complex for implementation. So if not correctly implemented, this might lead to unexpected error or incorrect o/p.

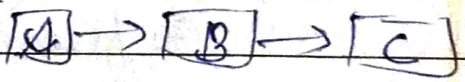
* TYPEs of Inheritance

i) Single Inheritance :- A ⟶ B

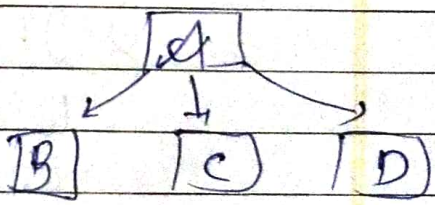
ii) Multiple Inheritance :-



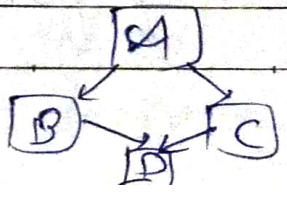
iii) Multi-level Inheritance :-



iv) Hierarchical Inheritance :-



v) Hybrid Inheritance :-



Q. What is Subclass?

→ It is an entity, which inherits from another class (i.e. child class, it calls)

Q. Define superclass?

→ which allows subclasses or child class to inherit from itself.

Q. What is inline function?

→ It is technique used by compilers and instructs to insert complete body of function wherever that function is used in program source code.

Q Friend function is a friend of class that is allowed to access to public, Private, Protected data in same class.

* Function Overloading

→ It is defined as a normal function but it has the ability to Perform different tasks.

→ It allows creation of several methods with same name which differ from each other by type of input and output of function.

* Operator overloading?

→ It is a function where different operators are applied and depends on arguments.

→ Operator, -, * can be used to pass through the function and it has their own precedence to execute.

Q. Abstract class?

→ It is a class which can't be ~~instantiated~~. implemented
→ Creation of object is not possible with abstract class, but it can be inherited.

* Interface / ~~Pure~~ Pure abstract class

→ It refers to special type of class, which contains methods, but not their definition.

→ Only declaration of methods is allowed.
→ To use interface, we can't create objects, instead, we need to implement that interface and define the methods of their implementation.

Difference b/w Interface & Abstract class

→ An interface is implemented, the subclass must define all its method, and provide its implementation. Whereas when an abstract ~~method~~ class is inherited, the subclass doesn't need to provide the definition of abstract method, until subclass is being used.

* Acces Modifers:

→ It used to control or specify the accensibility of entities like class, methods.

→ It also plays a vary vital role in achieving Encapsulation.

• Private • Protected • Public • friend

* TYPEs of Arguments

i) Call by value :- value pasced will get modified only inside the function. and it returns the same value whatever it is pasced ~~in~~ into function.

ii) Call by Reference :- value pasced will get modified in both inside and outside function and it return the same or different value.

Q What is super keyword?

→ It is used to invoke overridden method which overrides one of its super class methods. This keyword allow to acces overridden methods and also to acces hidden members of superclass.

Revise

* Overriding :- (Runtime Poly.)

→ It is allows an entity has same name, but its implementation changes during execution. Ex:- Method overridi

* Overloading :- (Compiletime Poly.)

→ It allows an entity has multiple implementation with same name. Ex:- Method overloading and Operator overloa

Q what is an exception?

→ It can be considered as a special event, which is raised during the execution of a program is mainly due to a position in program, where the user wants to do something for which program is not specified. Like undesirable i/p.

Q What is meant by exception handling?

Exception is an event that occurs during the execution of a program. Exceptions can be any type - Runtime exception, Error exceptions. Those exceptions are handle properly through exception handling mechanism like try, catch and throw keywords

Q Difference b/w class and an object?

OBJECT	CLASS
• An object is an instance of a class. objects hold any information	• class don't have any information
• Object doesn't have sub-objects	• class can have sub-classes.

Q What is difference b/w structure and a class?

Structure	class
→ Structure default access type is public.	→ class access type is private.
→ It is used for grouping data.	→ It can be used for grouping data and methods
→ They are exclusively used for data and it doesn't require strict validation.	→ They are used to encapsulates and inherit data which require strict validation.

Q Explain RESTful APIs and how you fetch data in React.JS?

Q→ How would you handle an API failure or t le in react

→ A Representational state Transfer API is a web service that allows communication b/w client and server using standard HTTPs methods.

We use fetch() function or Axios to make API calls

→ API failure handle by .catch() or try -- catc.
→ " " " " set a time limit for API calls
→ " " " " automatically retry failed requests.

* Difference between useState and useEffect hook in react JS.

useState is hook that allows to add state to funcitional component in reacts

UseState returns an array with two values:-
i) Current states value
ii) function update states

UseEffect is hook that allows to perform side effects in functional components.

→ It takes two arguments:-
→ It is contains side effects code.
→ The effects run only when one of these dependencies
→ It is used for cleanup like removing event listeners..

What is Virtual function?

→ Virtual is a keyword in C++.
→ A virtual function is redefined in derived class.

→ when a virtual function is defined in base class, then pointer to base class is created. Now, on the basis of type of object assigned, the respective class function is called.

Eg:

```
#include<bits/stdc++.h>
class A
{
public:
    virtual void show(){
        cout<<"Base class";
    }
};
class B: public A{
public:
    void show(){
        cout<<"Derived class";
    }
};
int main(){
    A* bptr = B aa;
    bptr = &aa;
    bptr -> show();
    return 0;
}
```

Q What is Virtual function?

→ Virtual is a keyword in C++.
→ A virtual function is redefined in derived class.

→ when a virtual function is defined in base class, then pointer to base class is created. Now, on the basis of type of object assigned, the reseeptive class function is called.

Eg:

```
#include<bits/stdc++.h>
class A
{
public:
    virtual void show(){
        cout<<"Base class";
    }
};
class B: public A{
public:
    void show(){
        cout<<"Derived class";
    }
};
int main(){
    A* bptr = B aa;
    bptr = &aa;
    bptr->show();
    return 0;
}
```

Q. What is Exception handling?

* Exception are some unpredictable condition when our program terminate suddenly with some error/issues. (generally TLE)

→ Exception handling provides us the facility to handling the exception so that our program keep running.

-> we use try and catch() for exception handling

* Overloading (compile-time poly.)

→ Same function name but different parameters.

* Overriding (Run time poly.)

→ A function in the child class redefines the parent class function with same name and parameters

* Interface

→ It is a blue print for a class that contain only abstract method

* Encapsulation

→ Bundling data and methods that operate on data into a single unit and restricting direct access to some details.

Ex:- bank Balance

*

* Abstraction

→ Hiding the complex implementation and only shows essential functionalities to user.

Ex:- Car

* Inheritance

→ It allow a child class to inherit properties and methods from a parent class, avoiding code duplication.

Ex:- A bird has wings and beak but parrot and egle inherit properties while having uniques

Single, Multiple, Multilevel, hybrid, Hierachy

* Polymorphism

→ It allows one function or method to behave differently based on context.

i) Compile time (Overloading) - Same name different Parameter

ii) Run time (Overriding) - Derived class redefine

Work Experience

Q How do you handle and process real-time stock price in ur projects

→ fetching data from a finical API
→ Processing data using python
→ Updating UI in real-time using React.js and websocket

Q Chanllanges:-

Data inconsistency due to delayed stock updates

Overcome:-

→ Implemented Data Validation: checked missing data
→ Used caching to reduce API calls and improve response time
→ Optimised API Calls:- used batch requests instead of frequent single requests

Q. Explain how you fetched and displayed real-time stock prices?

→ Used websockets to receive real-time-updates
→ stored stock data in React state
→ Updated UI using useEffect()

Q How do you handle slow API response in react app?

→ Async/Await + loading State to show loading indic.
→ Retry mechanism if request fails.
→ Cache response using local storage
→ Use a Background fatch to pre-fatch stock data.

Q what did you learn from ur J.P. Morgan exp:?

→ Python improved, Data handling, Real skills
→ learned how to debug real-time stock data issues.
→

Q Constructors and destructors in C++?

A constructor initialises an object when it's created.
A destructor is called when an object goes out of scope, used to ~~reuse~~ release resources.

Q. Overloading: Same function name, different parameters in same class.

Overriding: Same function and parameter in base and derived class.

Q Pointer:-
It hold the memory address of another variable.
A reference is an alias to an existing variable and must be initialized.

Q. Tell me about forecasting

It is a weather app built using React JS It fetches real-time weather using an API and displays forecasts with a clean and responsive UI. I used useEffect for API calls and useState for state management.

Q. Challenges in Netflix clone?

while building the Netflix clone, I faced layout issues on small screens. I resolved this using responsive design with flexbox and media Queries.

Q. UseEffect

It's a hook that run side effects in components like fetching data or updating Dom.

Q What is virtual DOM in React?

It is lightweight JS objects that react use to improve UI rendering efficiency by minimizing direct manipulation of real Dom.

Q technologies in Portfolio website?

I used HTML, CSS, JS to creat a responsive portfolio that work on all screen size. I implemented smoth scroll, section, animation and a contact form using basic JS.

Q Handle API integration in forecasting?

I used fetch API in react and called the open weather API in useEffect hook to get weather data component load.

Q Improvement

→ Add geolocation support
→ show hourly forecasts
→ Cache API response
→ Add dark toggle

Q Netflix different from real one

It's static UI clone. It mimics the homepage design with scrollable movie sections and layout responsive, but lacks backend functionality like user login and video streaming.

- Abstract class vs Inheritance

An abstract class is used when we want to enforce that derived classes must implement certain function.

- Git & Github

Git is distributed version control system that helps track changes in source code, while Github is a platform built on top of Git where developer can host and collaborate on projects.