OOP Interview Questions

What is Object-Oriented Programming (OOP)?
Answer: OOP is a programming paradigm that uses objects and classes to design and structure programs. It emphasizes the encapsulation of code and data, inheritance, polymorphism, and abstraction.

Can you explain the four basic principles of OOP?
Answer: The four basic principles of OOP are

- **Encapsulation.**
  hide the state (attributes / properties) from explicit / direct public access / external access. this helps keep the integrity / protect it from corruption.
- **Abstraction.**
  reveal only the relevant behavior / methods to the user , and hide the implementation (internal processes).
- **Inheritance.**
  to inherit properties/ methods from a general / parent calss.
- **Polymorphism.**
  the ability of a subclass / child to extend / morph the functionality of the parent class.

What is a Class?
Answer: A class is a blueprint or template for creating objects. It defines a set of properties and methods that the created objects share.

What is an Object?
Answer: An object is an instance of a class. It contains data and behavior as defined by the class.

What is Inheritance?
Answer: Inheritance is a mechanism where a new class, called a subclass, is derived from an existing class, called a superclass. The subclass inherits attributes and behaviors (methods) from the superclass.

Can you explain Polymorphism?

Answer: Polymorphism allows objects of different classes to be treated as objects of a common superclass; it is the ability to present the same interface for differing underlying data types.

What is Encapsulation?
Answer: Encapsulation is the concept of wrapping data and the methods that manipulate that data into a single unit, restricting direct access to some of an object's components.

What is Abstraction?
Answer: Abstraction is the process of hiding the complex reality while exposing only the necessary parts. It helps in reducing programming complexity and effort.

How do you achieve encapsulation in OOP?
Answer: Encapsulation is achieved by making the class variables private and providing public setter and getter methods to modify and view the variables' values.

What is the difference between an Abstract Class and an Interface?
Answer: An abstract class cannot be instantiated and may contain implementation; an interface is a pure abstraction that cannot have any implementation at all. Classes can implement multiple interfaces but usually extend only one abstract class.

What is the 'this' keyword and what does it represent?
Answer: The 'this' keyword represents the current instance of the class and is used to access class variables and methods.

What are design patterns and why are they important?
Answer: Design patterns are standard solutions to common problems in software design. They are important because they provide tested, proven development paradigms, improving code readability, reusability, and maintainability.

How does composition differ from inheritance?
Answer: Composition involves constructing complex objects from simpler ones, whereas inheritance derives new classes from existing classes, creating a hierarchy.

Can you explain the concept of 'method overloading' and 'method overriding'?

Answer: Method overloading is the ability to define several methods with the same name but different signatures. Method overriding is redefining a superclass's method in a subclass with the same signature.

What is a destructor and how is it used in OOP?

Answer: A destructor is a special class function that is called when an object goes out of scope or is explicitly destroyed; it is used to clean up resources that the object may have acquired during its lifetime.

How can OOP lead to tighter code security?

Answer: OOP provides tighter code security through encapsulation, as it restricts access to the internal state of objects, and only exposes a limited interface.

What is a constructor and how is it used in OOP?

Answer: A constructor is a special type of subroutine called to create an object. It prepares the new object for use, often accepting parameters that the constructor uses to set member variables.

Why are getter and setter methods used?

Answer: They are used to access and update the value of a private variable. This is part of encapsulation, which allows for validation and control over how the data is accessed or modified.

Can you explain the principle of 'separation of concerns'?

Answer: This principle involves separating a computer program into distinct sections, such that each section addresses a separate concern or part of the problem.