OOP Interview Questions & Answers - VeloxCare First Round

1) Introduction About Your Self?

Prepare a short 1–2 minute intro with your education, skills, projects, and strengths.

2) What is Polymorphism?

Polymorphism means 'one name, many forms'. It allows the same function/method to behave differently depending on the object.

3) How to use Polymorphism & when to use Polymorphism?

Use it when you want reusable and maintainable code. Achieved through method overloading (compile-time) and overriding (runtime).

4) What is method overloading & method overriding?

Overloading: same method name, different parameters (compile-time). Overriding: same method name & parameters, but redefined in child class (runtime).

5) What is public, private & protected?

public: accessible everywhere. private: accessible only inside the class. protected: accessible inside class + subclasses.

6) Which Situation you can use private, Protected and public?

private: when you want to hide sensitive data. protected: when subclasses need access. public: when functionality should be available globally.

7) What is Abstraction?

Abstraction is hiding implementation details and showing only essential features.

8) Where to use it & how to use it?

Use when hiding logic and exposing functionality. Achieved with abstract classes & interfaces.

9) What is Interface & How to use it?

Interface defines a contract (methods without implementation). Classes must implement it.

10) Difference between interface & abstraction?

See detailed table below.

11) When to use Abstraction & Interface?

Abstract: when classes share base behavior. Interface: when unrelated classes need the same contract.

12) What is Encapsulation & How to use it?

Encapsulation is wrapping data & methods inside a class and restricting direct access using access modifiers.

13) How can it Hide the data?

By making fields private and exposing them with getters/setters (properties in C#).

14) Ensure that encapsulation can wrap the data or Hiding explain perfectly with example.

Example: class Student { private string name; public string Name { get { return name; } set { name = value; } } }

15) Give the proper Example of Encapsulation?

Bank Account → balance is private, deposit/withdraw methods control access.

16) Explain methodology of Stack & Queue.

Stack → LIFO (Last In, First Out). Queue → FIFO (First In, First Out).

17) Difference between Stack & Queue?

See detailed table below.

18) Real-world Example of stack and queue?

Stack: Undo in editors, browser back button. Queue: Ticket booking, printer tasks.

19) Which methodology stack is more useful than Queue?

When reverse order is needed, e.g., recursion, function calls.

20) What is Dependency Injection?

Design pattern where dependencies are provided externally instead of creating inside the class.

21) Example of Polymorphism – Why same method work differently in different objects?

Shape s1 = new Circle(); Shape s2 = new Rectangle(); s1.Draw(); → Circle draw, s2.Draw(); → Rectangle draw.

Detailed Differences

1) Method Overloading vs Method Overriding

Feature	Method Overloading	Method Overriding
Definition	Same method name, different parameters	Same method name & parameters, different implem
Type	Compile-time polymorphism	Runtime polymorphism
Return Type	Can be same or different	Must be same or covariant
Inheritance	Not required	Requires inheritance (parent → child)
Access Modifiers	Can be any	Cannot reduce visibility
Example (C#)	Add(int a, int b) / Add(double a, double b)	Parent: Draw(), Child: override Draw()

2) Interface vs Abstraction

Feature	Abstract Class	Interface
Definition	Can have abstract + concrete methods	Only method signatures (no implementation until
Implementation	Can have method body	Cannot have method body (before C# 8)
Multiple Inheritance	Only one abstract class allowed	Multiple interfaces allowed
Access Modifiers	Can use public, private, protected	Methods are public by default
Fields & Variables	Can have fields, constants, constructors	Cannot have instance fields, only constants
When to Use	When classes share base behavior	When unrelated classes must follow same contr
Example (C#)	abstract class Shape { public abstract void Draw(); }	interface IShape { void Draw(); }

3) Stack vs Queue

Feature	Stack (LIFO)	Queue (FIFO)
Definition	Last In, First Out	First In, First Out
Order	Reverses order of items	Preserves order of items
Insertion	Push() adds to top	Enqueue() adds to rear
Deletion	Pop() removes from top	Dequeue() removes from front
Use Cases	Function calls, Undo, Backtracking	Printer tasks, Ticket queues, OS scheduling
Example	Books stack – last added removed first	People queue – first person served first

4) Encapsulation vs Abstraction

Feature	Encapsulation	Abstraction
Definition	Wrapping data & methods in a class	Hiding implementation & showing only functiona

Access Control	Achieved using access modifiers	Achieved using abstract classes & interfaces
Purpose	Restrict direct access to data	Provide only necessary information
Example	private string password with getter/setter	abstract void Draw();
Focus	Data hiding	Design level hiding