OOP: The Backbone of Modern Programming!

Ever wondered why OOP is the foundation of most enterprise applications? Let's break it down!

① What is OOP?

Object-Oriented Programming (OOP) is a programming paradigm based on the concept of "**objects**", which encapsulate data and behavior.

🧠 **Analogy:**

Think of OOP like a car 🚗:

- The engine is private (encapsulation)
- The accelerator controls speed (abstraction)
- Different car brands share features (inheritance)
- Swappable car parts (polymorphism)

② OOP Pillars Overview

🛠️ Four Pillars of OOP

- **Encapsulation** - Hides data, exposing only necessary parts.
- **Abstraction** - Hides complexity, showing only essential details.
- **Inheritance** - Allows code reuse via parent-child relationships.
- **Polymorphism** - Enables multiple forms of the same method.

③ Encapsulation

Encapsulation restricts direct access to an object's data and only exposes safe methods to interact with it.

```
public class BankAccount
{
    private decimal balance; // Private field

    public void Deposit(decimal amount)
    {
        if (amount > 0)
            balance += amount;
    }

    public decimal GetBalance()
    {
        return balance; // Controlled access
    }
}
```

④ Abstraction

Abstraction hides complex details and exposes only necessary features.

```
1  public abstract class Vehicle
2  {
3      public abstract void Start(); // Abstract method
4  }
5
6  public class Car : Vehicle
7  {
8      public override void Start()
9      {
10         Console.WriteLine("Car starting with key...");
11     }
12 }
```

⑤ Inheritance

Inheritance allows one class to acquire properties and methods of another class.

```
1 public class Animal
2 {
3     public void Eat() => Console.WriteLine("Eating...");
4 }
5
6 public class Dog : Animal
7 {
8     public void Bark() => Console.WriteLine("Barking...");
9 }
```

6 Polymorphism

Polymorphism allows one method to have multiple implementations.

```
1 public class Shape
2 {
3     public virtual void Draw() => Console.WriteLine("Drawing a shape");
4 }
5
6 public class Circle : Shape
7 {
8     public override void Draw() => Console.WriteLine("Drawing a circle");
9 }
```

7 OOP vs Procedural Programming

Feature	OOP (C#)	Procedural (C)
Approach	Objects & Classes	Functions & Procedures
Code Reusability	High (via inheritance & polymorphism)	Low (code repetition)
Data Security	High (Encapsulation)	Low (Global variables)
Scalability	Highly Scalable	Less Scalable

When to use OOP?

- Large, complex applications
- Enterprise-level software (e.g., Banking, E-commerce)
- When maintainability & reusability are key

Manish Jaiswal

OOP Interview

8 Interview Questions on OOP

Top Interview Questions

- What are the four pillars of OOP?
- How does encapsulation improve security?
- Difference between abstraction and encapsulation?
- When would you use composition over inheritance?
- Explain method overriding vs method overloading.

Let me know if this helped!