

How Abstraction Makes Your Code 10x Cleaner!



① What is Abstraction?

- Hides unnecessary details
 - Shows only relevant information
 - Improves maintainability & scalability
-  **Example:** When you pay online, you just select a method (Credit Card, PayPal, UPI).
● You don't see how the payment is processed behind the scenes!



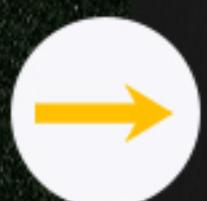
② Real World Example

Abstraction in Code:

A common interface allows different payment providers to be integrated seamlessly.



```
1 public abstract class PaymentProcessor
2 {
3     public abstract void ProcessPayment(decimal amount);
4 }
5
6 public class PayPalProcessor : PaymentProcessor
7 {
8     public override void ProcessPayment(decimal amount)
9     {
10         Console.WriteLine($"Processing ${amount} via PayPal.");
11     }
12 }
```



③ How Abstraction Simplifies Systems

- **Users interact with a simple interface** → No need to understand internal complexities
- **Easily switch implementations** → Add new payment methods (Stripe, Razorpay) without modifying existing code
- **Encapsulates business logic** → Prevents direct access to sensitive processing details



④ Abstraction vs Encapsulation (Common Confusion!)

🚀 **Abstraction:** Hides implementation details from the user.

🔒 **Encapsulation:** Hides implementation details within the class



⑤ Example

 **Abstraction** – Users see a "Pay Now" button but don't know how payments are processed

 **Encapsulation** – The ProcessPayment() method secures card details inside private fields



⑧ Common Questions ?

1) Is abstraction only useful in large applications?

 No! Even in small projects, it makes your code cleaner & scalable.

2) An abstract class vs an interface?

 Abstract Class: Can have implemented methods & fields.

Interface: Only defines method signatures —no implementation.

3) Does abstraction reduce performance?

 No! It improves code organization without affecting speed.



Let me know if this helped!

