Real-World Scenarios for Function Overriding in C#

# 🔁 What is Function Overriding?

Function overriding in C# happens when a derived class provides a new implementation of a virtual method defined in its base class. It enables polymorphism, allowing the derived class to customize or extend base class behavior.

# 📦 1. E-commerce System - Payment Processing

Scenario: Different payment types override the payment process logic.

public class Payment  
{  
 public virtual void ProcessPayment()  
 {  
 Console.WriteLine("Processing generic payment...");  
 }  
}  
  
public class CreditCardPayment : Payment  
{  
 public override void ProcessPayment()  
 {  
 Console.WriteLine("Processing credit card payment.");  
 }  
}  
  
public class PayPalPayment : Payment  
{  
 public override void ProcessPayment()  
 {  
 Console.WriteLine("Processing PayPal payment.");  
 }  
}

# 🏥 2. Hospital Management - Appointments

Scenario: In-person and virtual appointments schedule differently.

public class Appointment  
{  
 public virtual void Schedule()  
 {  
 Console.WriteLine("Scheduling a generic appointment...");  
 }  
}  
  
public class InPersonAppointment : Appointment  
{  
 public override void Schedule()  
 {  
 Console.WriteLine("Scheduling an in-person appointment with a room.");  
 }  
}  
  
public class VirtualAppointment : Appointment  
{  
 public override void Schedule()  
 {  
 Console.WriteLine("Scheduling a virtual appointment with a video link.");  
 }  
}

# 🏦 3. Banking System - Account Types

Scenario: Savings and current accounts calculate interest differently.

public class BankAccount  
{  
 public virtual void CalculateInterest()  
 {  
 Console.WriteLine("Calculating interest in base account.");  
 }  
}  
  
public class SavingsAccount : BankAccount  
{  
 public override void CalculateInterest()  
 {  
 Console.WriteLine("Calculating interest for savings account at 5%.");  
 }  
}  
  
public class CurrentAccount : BankAccount  
{  
 public override void CalculateInterest()  
 {  
 Console.WriteLine("No interest for current account.");  
 }  
}

# 🚗 4. Vehicle System - Vehicle Start

Scenario: Different vehicles have different starting mechanisms.

public class Vehicle  
{  
 public virtual void Start()  
 {  
 Console.WriteLine("Starting a vehicle.");  
 }  
}  
  
public class Car : Vehicle  
{  
 public override void Start()  
 {  
 Console.WriteLine("Starting a car with a key.");  
 }  
}  
  
public class ElectricScooter : Vehicle  
{  
 public override void Start()  
 {  
 Console.WriteLine("Starting an electric scooter with a button.");  
 }  
}

# 🧠 Summary

Scenario | Base Class | Overridden Method | Purpose  
-----------------------|-------------------|------------------------|-------------------------------------------  
E-commerce Payment | Payment | ProcessPayment() | Different payment types  
Hospital Appointments | Appointment | Schedule() | In-person vs virtual  
Banking | BankAccount | CalculateInterest() | Interest logic differs by account type  
Vehicle System | Vehicle | Start() | Cars and scooters start differently