# Best Practices in Constructors

**Use this Keyword:**

* Avoid ambiguity when parameter names are the same as attribute names.
* Example: this.customerName = customerName;

**Keep Logic Simple:**

* Avoid heavy computations or database calls inside constructors.

**Provide Multiple Constructors:**

* Support various initialization scenarios by overloading constructors.

**Encapsulate Logic:**

* Use private methods (like **CalculatePrice()**) to keep constructors clean.

**Best Practices in Access Modifiers**

**Use the Least Privilege:**

* Start with the most restrictive modifier (private) and relax it as needed (protected or public).

**Encapsulation:**

* Always make attributes private and use getters/setters for controlled access.

**Protected Usage:**

* Use protected only when inheritance is required and controlled access is necessary.

**Avoid Overexposure:**

* Limit the use of public to methods or classes that are meant to be accessed by external code.

**Avoid Leaks:**

* Be cautious with exposing mutable objects like collections via getters. Return a copy or an unmodifiable view when possible.

**Level 1 Practice Programs**

1. **Book Class**
   * Create a Book class with attributes title, author, and price.
   * Provide both default and parameterized constructors.

| using System; using System.Collections.Generic; using System.Diagnostics; using System.Linq; using System.Text; using System.Threading.Tasks;  namespace ConstructorVariableModifier {  class Book  {  public string title ;  public string author ;  public double price ;  // Default Constructor  public Book()   {  this.title = "Unknown";  this.author = "Unknown";  this.price = 0.0;  }   // Parameterized Constructor  public Book(string title, string author, double price)   {  this.title = title;  this.author = author;  this.price = price;  }   // method to display result  public void DisplayDetails()  {  Console.WriteLine("Book Title: " + title);  Console.WriteLine("Author: " + author);  Console.WriteLine("Price: " + price);  }   public static void Main()  {  // create object of Book class  Book book1 = new Book();  Book book2 = new Book("The Last Kingdom" , "Rohit" , 100);   book1.DisplayDetails();  book2.DisplayDetails();  }  } } |
| --- |

1. **Circle Class**
   * Write a Circle class with a radius attribute.
   * Use constructor chaining to initialize radius with both default and user-provided values.

| using System; using System.Collections.Generic; using System.Diagnostics; using System.Linq; using System.Text; using System.Threading.Tasks;  namespace ConstructorVariableModifier {  class Circle  {  public double radius;   // Constructor chaining  public Circle() : this(1.0) { }   // Parameterized Constructor  public Circle(double radius)  {  this.radius = radius;  }   public void DisplayDetails()  {  Console.WriteLine("Circle Radius: " + this.radius);  }   public static void Main()  {  // create object of Circle class  Circle circle1 = new Circle();  Circle circle2 = new Circle(5);   circle1.DisplayDetails();  circle2.DisplayDetails();  }  } } |
| --- |

1. **Person Class (Copy Constructor)**
   * Create a Person class with a copy constructor that clones another person's attributes.

| using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks;  namespace ConstructorVariableModifier {  // Person Class (Copy Constructor)  class Person  {  public string name;  public int age;   // Parameterized Constructor  public Person(string name, int age)  {  this.name = name;  this.age = age;  }   // Copy Constructor  public Person(Person other)   {  this.name = other.name;  this.age = other.age;  }   public void DisplayDetails()  {  Console.WriteLine("Person name: " + this.name);  Console.WriteLine("Person age: " + this.age);  }   public static void Main()  {  // create object of Person class  Person person1 = new Person("Rohit" , 24);  Person person2 = new Person(person1);   person1.DisplayDetails();  person2.DisplayDetails();  }  } } |
| --- |

1. **Hotel Booking System**
   * Create a HotelBooking class with attributes guestName, roomType, and nights.
   * Use default, parameterized, and copy constructors to initialize bookings.

| using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks;  namespace ConstructorVariableModifier {  class HotelBooking  {  public string guestName;  public string roomType;  public int nights;   // Default Constructor  public HotelBooking()   {  this.guestName = "Guest";  this.roomType = "Standard";  this.nights = 1;  }   // Parameterized Constructor  public HotelBooking(string guestName, string roomType, int nights)   {  this.guestName = guestName;  this.roomType = roomType;  this.nights = nights;  }   // Copy Constructor  public HotelBooking(HotelBooking other)   {  this.guestName = other.guestName;  this.roomType = other.roomType;  this.nights = other.nights;  }   public void DisplayDetails()  {  Console.WriteLine("Guest Name: " + this.guestName);  Console.WriteLine("Room Type: " + this.roomType);  Console.WriteLine("number of nights: " + this.nights);  }   public static void Main()  {  // create object of HotelBooking class  HotelBooking hb1 = new HotelBooking();  HotelBooking hb2 = new HotelBooking("Rohit" , "AC Room" , 3);  HotelBooking hb3 = new HotelBooking(hb2);   hb1.DisplayDetails();  hb2.DisplayDetails();  hb3.DisplayDetails();  }  } } |
| --- |

1. **Library Book System**
   * Create a Book class with attributes title, author, price, and availability.
   * Implement a method BorrowBook() to borrow a book.

| using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks;  namespace ConstructorVariableModifier {  class LibraryBook  {  public string title;  public string author;  public double price;  public bool availability;   public LibraryBook(string title, string author, double price, bool availability)  {  this.title = title;  this.author = author;  this.price = price;  this.availability = availability;  }   public void BorrowBook()  {  if (availability)  {  availability = false;  Console.WriteLine($"{this.title} has been borrowed");  }  else  {  Console.WriteLine($"{this.title} is not available");  }  }   public void DisplayDetails()  {  Console.WriteLine("Book title: " + this.title);  Console.WriteLine("Book author: " + this.author);  Console.WriteLine("Book price: " + this.price);  Console.WriteLine("Book availability: " + this.availability);  }   public static void Main()  {  // create object of LibraryBook class  LibraryBook lb = new LibraryBook("The Last Kingdom", "Rohit", 100 , true);   lb.DisplayDetails();  }  }  } |
| --- |

1. **Car Rental System**
   * Create a CarRental class with attributes customerName, carModel, and rentalDays.
   * Add constructors to initialize the rental details and calculate total cost.

| using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks;  namespace ConstructorVariableModifier {  class CarRental  {  public string customerName;  public string carModel;  public int rentalDays;  public double costPerDay = 5000.0;    public CarRental(string customerName, string carModel, int rentalDays)  {  this.customerName = customerName;  this.carModel = carModel;  this.rentalDays = rentalDays;  }   public double CalculateTotalCost()  {  return this.rentalDays \* this.costPerDay;  }    public static void Main()  {  // create object of CarRental class  CarRental cr = new CarRental("Rohit" , "FERRARI - SF90XX", 5);   Console.WriteLine($"Total rent is {cr.CalculateTotalCost()}");  }  }  } |
| --- |

**1. Instance vs. Class Variables and Methods**

**Problem 1: Product Inventory**

* **Create a Product class with:**
  + **Instance Variables: productName, price.**
  + **Class Variable: totalProducts (shared among all products).**
* **Implement the following methods:**
  + **An instance method DisplayProductDetails() to display the details of a product.**
  + **A class method DisplayTotalProducts() to show the total number of products created.**

| **using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks;  namespace ConstructorVariableModifier {  class Product  {  public string productName;  public double price;  public static int totalProducts = 0;   public Product(string productName, double price)  {  this.productName = productName;  this.price = price;  Product.totalProducts++;  }   public void DisplayProductDetails()  {  Console.WriteLine($"Product: {this.productName}, Price: {this.price}");  }   public static void DisplayTotalProducts()  {  Console.WriteLine($"Total Products: {Product.totalProducts}");  }   public static void Main()  {  Product product = new Product("Shirt" , 1100);   product.DisplayProductDetails();  Product.DisplayTotalProducts();  }  } }** |
| --- |

**Problem 2: Online Course Management**

* **Design a Course class with:**
  + **Instance Variables: courseName, duration, fee.**
  + **Class Variable: instituteName (common for all courses).**
* **Implement the following methods:**
  + **An instance method DisplayCourseDetails() to display course details.**
  + **A class method UpdateInstituteName() to modify the institute name for all courses.**

| **using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks;  namespace ConstructorVariableModifier {  class Course  {  public string courseName;  public int duration;  public double fee;  public static string instituteName = "GL University";   public Course(string courseName, int duration, double fee)  {  this.courseName = courseName;  this.duration = duration;  this.fee = fee;  }   public void DisplayCourseDetails()  {  Console.WriteLine($"Course: {this.courseName}, Duration: {this.duration} weeks, Fee: {this.fee}, Institute: {Course.instituteName}");  }   public static void UpdateInstituteName(string newName)  {  Course.instituteName = newName;  }   public static void Main()  {  Course course = new Course("MCA", 3, 300000);   course.DisplayCourseDetails();  Course.UpdateInstituteName("GLA University");  Console.WriteLine($"Modified Institute Name is {Course.instituteName}");   }  } }** |
| --- |

**Problem 3: Vehicle Registration**

* **Create a Vehicle class to manage vehicle details:**
  + **Instance Variables: ownerName, vehicleType.**
  + **Class Variable: registrationFee (fixed for all vehicles).**
* **Implement the following methods:**
  + **An instance method DisplayVehicleDetails() to display owner and vehicle details.**
  + **A class method UpdateRegistrationFee() to change the registration fee.**

| **using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks;  namespace ConstructorVariableModifier {  class Vehicle  {  public string ownerName { get; set; }  public string vehicleType { get; set; }  public static double registrationFee = 500.0;   public Vehicle(string ownerName, string vehicleType)  {  this.ownerName = ownerName;  this.vehicleType = vehicleType;  }   public void DisplayVehicleDetails()  {  Console.WriteLine($"Owner: {this.ownerName}, Vehicle Type: {this.vehicleType}, Registration Fee: {Vehicle.registrationFee}");  }   public static void UpdateRegistrationFee(double newFee)  {  Vehicle.registrationFee = newFee;  }   public static void Main()  {  Vehicle vehicle = new Vehicle("Rohit" , "Two wheeler");   vehicle.DisplayVehicleDetails();  Vehicle.registrationFee = 1000;  Console.WriteLine($"Modified Registration fee is {Vehicle.registrationFee}");    }  } }** |
| --- |

**2. Access Modifiers**

**Problem 1: University Management System**

* **Create a Student class with:**
  + **rollNumber (public)**
  + **name (protected)**
  + **CGPA (private)**
* **Implement methods to:**
  + **Access and modify CGPA using public methods.**
  + **Create a subclass PostgraduateStudent to demonstrate the use of protected members.**

| **using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks;  namespace ConstructorVariableModifier {  class Student  {  public int rollNumber;  protected string name;  private double CGPA;   public Student(int rollNumber, string name, double cgpa)  {  this.rollNumber = rollNumber;  this.name = name;  this.CGPA = cgpa;  }   public double GetCGPA()  {  return this.CGPA;  }  public void SetCGPA(double cgpa)  {  this.CGPA = cgpa;  }  }   class PostgraduateStudent : Student  {  public string researchTopic;   public PostgraduateStudent(int rollNumber, string name, double cgpa, string researchTopic)  : base(rollNumber, name, cgpa)  {  this.researchTopic = researchTopic;  }   public static void Main()  {  PostgraduateStudent pgStudent = new PostgraduateStudent(174, "Rohit", 9.5, "Scheduling");  Console.WriteLine($"Roll number: {pgStudent.rollNumber}");  Console.WriteLine($"name: {pgStudent.name}");  Console.WriteLine($"CGPA: {pgStudent.GetCGPA()}");  Console.WriteLine($"Research topic: {pgStudent.researchTopic}");  }  } }** |
| --- |

**Problem 2: Book Library System**

* **Design a Book class with:**
  + **ISBN (public)**
  + **title (protected)**
  + **author (private)**
* **Implement methods to:**
  + **Set and get the author name.**
  + **Create a subclass EBook to access ISBN and title and demonstrate access modifiers.**

| **using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks; using ConstructorVariableModifier;  namespace ConstructorVariableModifier {  class Book  {  public string ISBN;  protected string title;  private string author;   public Book(string isbn, string title, string author)  {  this.ISBN = isbn;  this.title = title;  this.author = author;  }   public string GetAuthor()  {  return this.author;  }  public void SetAuthor(string author)  {  this.author = author;  }  }   class EBook : Book  {  public string fileFormat;   public EBook(string isbn, string title, string author, string fileFormat)  : base(isbn, title, author)  {  this.fileFormat = fileFormat;  }   public static void Main()  {  EBook ebook = new EBook(" 978-3-16-148410-0", "The Last Kingdom", "Rohit", "pdf");  Console.WriteLine($"ISBN number: {ebook.ISBN}");  Console.WriteLine($"title: {ebook.title}");  Console.WriteLine($"author: {ebook.GetAuthor()}");  Console.WriteLine($"file format: {ebook.fileFormat}");  }  } }** |
| --- |

**Problem 3: Bank Account Management**

* **Create a BankAccount class with:**
  + **accountNumber (public)**
  + **accountHolder (protected)**
  + **balance (private)**
* **Implement methods to:**
  + **Access and modify balance using public methods.**
  + **Create a subclass SavingsAccount to demonstrate access to accountNumber and accountHolder.**

| **using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks;  namespace ConstructorVariableModifier {  class BankAccount  {  public int accountNumber;  protected string accountHolder;  private double balance;   public BankAccount(int accountNumber, string accountHolder, double balance)  {  this.accountNumber = accountNumber;  this.accountHolder = accountHolder;  this.balance = balance;  }   public double GetBalance()  {  return this.balance;  }  public void SetBalance(double balance)  {  this.balance = balance;  }  }   class SavingsAccount : BankAccount  {  public double interestRate;   public SavingsAccount(int accountNumber, string accountHolder, double balance, double interestRate)  : base(accountNumber, accountHolder, balance)  {  this.interestRate = interestRate;  }   public static void Main()   {  SavingsAccount savingsAccount = new SavingsAccount(12348901 , "Rohit" , 50000 , 4.5);  Console.WriteLine($"Account Number: {savingsAccount.accountNumber}");  Console.WriteLine($"Account Holder: {savingsAccount.accountHolder}");  Console.WriteLine($"Account Balance: Rs {savingsAccount.GetBalance()}");  Console.WriteLine($"Interest Rate: {savingsAccount.interestRate}%");  }  } }** |
| --- |

**Problem 4: Employee Records**

* **Develop an Employee class with:**
  + **employeeID (public)**
  + **department (protected)**
  + **salary (private)**
* **Implement methods to:**
  + **Modify salary using a public method.**
  + **Create a subclass Manager to access employeeID and department.**

| using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks;  namespace ConstructorVariableModifier {  class Employee  {  public int employeeID;  protected string department;  private double salary;   public Employee(int employeeID, string department, double salary)  {  this.employeeID = employeeID;  this.department = department;  this.salary = salary;  }   public double GetSalary()  {  return this.salary ;  }   public void SetSalary(double salary)  {  this.salary = salary;  }  }  // subclass Manager  class Manager : Employee  {  public string teamName;   public Manager(int employeeID, string department, double salary, string teamName)  : base(employeeID, department, salary)  {  this.teamName = teamName;  }   public static void Main()  {  Manager manager = new Manager(123, "Finance", 50000, "Team Alpha");   Console.WriteLine($"Employee ID: {manager.employeeID}");  Console.WriteLine($"Department: {manager.department}");  Console.WriteLine($"Salary: {manager.GetSalary()}");  Console.WriteLine($"Team Name: {manager.teamName}");  }  } } |
| --- |