**Best Programming Practices in C#**

**General Guidelines**

1. **Use static** for shared values and utility methods to reduce memory usage and avoid redundancy.
2. **Leverage this** to avoid ambiguity when initializing attributes.
3. **Declare readonly variables** for identifiers or constants that should remain unchanged.
4. **Use is operator** for safe type-checking and to prevent runtime errors during typecasting.

**Sample Program 1: Bank Account System**

Create a BankAccount class with the following features:

* **static**:
  + A static variable bankName shared across all accounts.
  + A static method GetTotalAccounts() to display the total number of accounts.
* **this**:
  + Use this to resolve ambiguity in the constructor when initializing AccountHolderName and AccountNumber.
* **readonly**:
  + Use a readonly variable AccountNumber to ensure it cannot be changed once assigned.
* **is operator**:
  + Check if an account object is an instance of the BankAccount class before displaying its details.

| using System; using System.Collections;  namespace BankManagementSystem {  class BankAccount  {  private static ArrayList allAccounts = new ArrayList();  private static string bankName;  private static int totalAccounts = 0;  private readonly long accountNumber;  protected string accountHolderName;  private double balance;  private static int accountCounter = 1000;    static BankAccount()  {  bankName = "INDIAN BANK";  }   public static int GenerateAccountNumber()  {  Random random = new Random();  accountCounter++;  return accountCounter\* 1000 + random.Next(0, 1000);   }    public BankAccount(string accountHolderName, double balance)  {  this.accountNumber = GenerateAccountNumber();  this.accountHolderName = accountHolderName;  this.balance = balance;  totalAccounts++;  }   public static int GetTotalAccounts()  {  return totalAccounts;  }   public double GetBalance()  {  return this.balance;  }   public void Deposit(double amount)  {  if (amount > 0)  {  this.balance += amount;  Console.WriteLine($"{amount} deposited successfully");  }  else  {  Console.WriteLine("Enter valid deposit amount");  }  }   public void Withdraw(double amount)  {  if (amount > 0 && this.balance - amount >= 1000)  {  this.balance -= amount;  Console.WriteLine($"{amount} withdrawn successfully");  }  else  {  Console.WriteLine("Insufficient balance! Minimum balance of 1000 should be maintained");  }  }   public void Display()  {  Console.WriteLine("==================================");  Console.WriteLine($"Bank Name: {bankName}");  Console.WriteLine($"Account Number: {this.accountNumber}");  Console.WriteLine($"Account Holder: {this.accountHolderName}");  Console.WriteLine($"Balance: {this.balance}");  Console.WriteLine("==================================");  }   public static void AddCustomer(BankAccount other)  {  allAccounts.Add(other);  }    public static BankAccount GetAccountByNumber(int accountNumber)  {  foreach (BankAccount acc in allAccounts)  {  if (acc.accountNumber == accountNumber)  {  return acc;   }  }  return null;  }   public void BankManagement()  {  while (true)  {  Console.WriteLine("Choose an action:");  Console.WriteLine("1 - Check Balance");  Console.WriteLine("2 - Deposit");  Console.WriteLine("3 - Withdraw");  Console.WriteLine("4 - Display Account Details");  Console.WriteLine("5 - Exit");   int input = Convert.ToInt32(Console.ReadLine());    switch (input)  {  case 1:  Console.WriteLine($"Current Balance: {GetBalance()}");  break;  case 2:  Console.Write("Enter deposit amount: ");  double depositAmount = Convert.ToDouble(Console.ReadLine());    if (depositAmount>0)  {  Deposit(depositAmount);  }  else  {  Console.WriteLine("Invalid amount entered.");  }  break;  case 3:  Console.Write("Enter withdrawal amount: ");  double withdrawAmount = Convert.ToDouble(Console.ReadLine());  if (withdrawAmount > 0 && this.balance - withdrawAmount > 1000)  {  Withdraw(withdrawAmount);  }  else  {  Console.WriteLine("Invalid Trasaction! Minimum Rs 1000 should be maintained");  }  break;  case 4:  Display();  break;  case 5:  Console.WriteLine("Thank you for banking with us!");  return;  default:  Console.WriteLine("Invalid Input. Please try again.");  break;  }  }  }   public static void Main()  {  while (true)  {  Console.WriteLine("1 - Create New Account");  Console.WriteLine("2 - Show All Accounts");  Console.WriteLine("3 - Search Account by Number");  Console.WriteLine("4 - Exit");  Console.Write("Choose an option: ");   int input = Convert.ToInt32(Console.ReadLine());    switch (input)  {  case 1:   Console.Write("Account Holder Name: ");  string accountHolderName = Console.ReadLine();   double balance;  do  {  Console.Write("Initial Balance (minimum 1000): ");  balance = Convert.ToDouble(Console.ReadLine());  }  while (balance <1000);   BankAccount newAccount = new BankAccount(accountHolderName, balance);  AddCustomer(newAccount);  newAccount.BankManagement();  break;   case 2:  Console.WriteLine("All Accounts:");  foreach (BankAccount acc in allAccounts)  {  acc.Display();  }  break;    case 3:  Console.Write("Enter Account Number: ");  int searchAccountNumber = Convert.ToInt32(Console.ReadLine());  BankAccount account = GetAccountByNumber(searchAccountNumber);    if (account != null)  {  account.Display();  }  else  {  Console.WriteLine("Account not found.");  }  break;   case 4:  Console.WriteLine("Exiting program. Thank you!");  return;   }  }  }  } } |
| --- |

**Sample Program 2: Library Management System**

Create a Book class to manage library books with the following features:

* **static**:
  + A static variable LibraryName shared across all books.
  + A static method DisplayLibraryName() to print the library name.
* **this**:
  + Use this to initialize Title, Author, and ISBN in the constructor.
* **readonly**:
  + Use a readonly variable ISBN to ensure the unique identifier of a book cannot be changed.
* **is operator**:
  + Verify if an object is an instance of the Book class before displaying its details.

| using System;  class Book {  // Static variable   private static string libraryName = "GLA Library";   // Readonly variable   private readonly string isbn;    // Instance variables   private string title;  private string author;   // Constructor   public Book(string title, string author, string isbn)  {   this.title = title;  this.author = author;  this.isbn = isbn;  }   // Static method to display library name  public static void DisplayLibraryName()  {   Console.WriteLine("Library Name :" + libraryName);  }   // Method to display book details  public void DisplayBookDetails()  {  if (this is Book)  {  Console.WriteLine("Title: " + this.title);  Console.WriteLine("Author: " + this.author);  Console.WriteLine("ISBN: " + this.isbn);  }  else  {  Console.WriteLine("Invalid book");  }  }    public static void Main()  {  // Display library name  Book.DisplayLibraryName();    // Creating book instances  Book book = new Book("The Last Kingdom", "Rohit", "123456789");    // Display book details  book.DisplayBookDetails();  } } |
| --- |

**Sample Program 3: Employee Management System**

Design an Employee class with the following features:

* **static**:
  + A static variable CompanyName shared by all employees.
  + A static method DisplayTotalEmployees() to show the total number of employees.
* **this**:
  + Use this to initialize Name, Id, and Designation in the constructor.
* **readonly**:
  + Use a readonly variable Id for the employee ID, which cannot be modified after assignment.
* **is operator**:
  + Check if a given object is an instance of the Employee class before printing the employee details.

| using System;  class Employee {  // Static variable   private static string companyName = "Capgemini";    // Static variable   private static int employeCount = 0;   // Readonly variable   private readonly int id;    // Instance variables   private string name;  private string designation;   // Constructor   public Employee(int id, string name, string designation)  {  this.id = id;  this.name = name;  this.designation = designation;  employeCount++;   }    // Method to display company name  public void DisplayCompanyname()  {  Console.WriteLine("Company Name :" + companyName);  }   // Method to display employee details  public void DisplayEmployeeDetails()  {  if (this is Employee)  {  Console.WriteLine("Name: " + this.name);  Console.WriteLine("ID: " + this.id);  Console.WriteLine("Designation: " + this.designation);  }  else  {  Console.WriteLine("Invalid employee");  }  }    public static void Main()  {  // Creating employee instances  Employee emp = new Employee(101, "Rohit", "Software Engineer");    // Display company name  emp.DisplayCompanyname();    // Display employee details  emp.DisplayEmployeeDetails();  } } |
| --- |

**Sample Program 4: Shopping Cart System**

Create a Product class to manage shopping cart items with the following features:

* **static**:
  + A static variable Discount shared by all products.
  + A static method UpdateDiscount() to modify the discount percentage.
* **this**:
  + Use this to initialize ProductName, Price, and Quantity in the constructor.
* **readonly**:
  + Use a readonly variable ProductID to ensure each product has a unique identifier that cannot be changed.
* **is operator**:
  + Validate whether an object is an instance of the Product class before processing its details.

| using System;  class Product {  // Static variable   private static double discount = 10;    // Readonly variable   private readonly int productID;    // Instance variables   private string productName;  private double price;  private int quantity;   // Constructor   public Product(int productId, string productName, double price, int quantity)  {  this.productID = productId;  this.productName = productName;  this.price = price;  this.quantity = quantity;  }   // Static method   public static void UpdateDiscount(double newDiscount)  {  discount = newDiscount;  Console.WriteLine("discount updated to: " + discount + "%");  }   // Method to display product details   public void DisplayProductDetails()  {  if (this is Product)  {  double finalPrice = price - (price \* discount / 100);   Console.WriteLine("Product ID: " + this.productID);  Console.WriteLine("Name: " + this.productName);  Console.WriteLine("price (before discount): " + this.price);  Console.WriteLine("price (after discount): " + finalPrice);  Console.WriteLine("quantity: " + this.quantity);  }  else  {  Console.WriteLine("Invalid product");  }  }    public static void Main()  {  // Updating the discount percentage  Product.UpdateDiscount(15.0);    // Creating product instances  Product product = new Product(101, "Laptop", 1000, 5);    // Displaying product details  product.DisplayProductDetails();  } } |
| --- |

**Sample Program 5: University Student Management**

Create a Student class to manage student data with the following features:

* **static**:
  + A static variable UniversityName shared across all students.
  + A static method DisplayTotalStudents() to show the number of students enrolled.
* **this**:
  + Use this in the constructor to initialize Name, RollNumber, and Grade.
* **readonly**:
  + Use a readonly variable RollNumber for each student that cannot be changed.
* **is operator**:
  + Check if a given object is an instance of the Student class before performing operations like displaying or updating grades.

| using System;  class Student {  // Static variable   private static string universityName = "GLA University";  private static int StudentCount = 0;   // Readonly variable  private readonly int rollNumber;    // Instance variables   private string name;  private char grade;   // Constructor   public Student(string name, int rollNumber, char grade)  {  this.name = name;  this.rollNumber = rollNumber;  this.grade = grade;  StudentCount++;   }   // Static method to display total students  public static void DisplayTotalStudents()  {  Console.WriteLine("Total Students Enrolled: " + StudentCount);  }   // Method to update student grade  public void UpdateGrade(char newGrade)  {  if (this is Student)  {  this.grade = newGrade;  Console.WriteLine("grade updated successfully for Roll Number: " + this.rollNumber);  }  else  {  Console.WriteLine("Invalid student");  }  }   // Method to display student details  public void DisplayStudentDetails()  {  if (this is Student)  {  Console.WriteLine("name: " + this.name);  Console.WriteLine("Roll Number: " + this.rollNumber);  Console.WriteLine("grade: " + this.grade);  }  else  {  Console.WriteLine("Invalid student");  }  }    public static void Main()  {  // Display university name  Console.WriteLine("University: " + "GLA University");    // Creating student instances  Student student = new Student("Rohit", 174 , 'O');    // Displaying student details  student.DisplayStudentDetails();    // Updating student grade  student.UpdateGrade('A');    // Displaying total students   Student.DisplayTotalStudents();  } } |
| --- |

**Sample Program 6: Vehicle Registration System**

Create a Vehicle class with the following features:

* **static**:
  + A static variable RegistrationFee common for all vehicles.
  + A static method UpdateRegistrationFee() to modify the fee.
* **this**:
  + Use this to initialize OwnerName, VehicleType, and RegistrationNumber in the constructor.
* **readonly**:
  + Use a readonly variable RegistrationNumber to uniquely identify each vehicle.
* **is operator**:
  + Check if an object belongs to the Vehicle class before displaying its registration details.

| using System;  class Vehicle {  // Static variable   public static double registrationFee = 5000;    // Readonly variable  public readonly string registrationNumber;    // Instance variables  public string ownerName;  public string vehicleType;   // Constructor   public Vehicle(string ownerName, string vehicleType, string registrationNumber)  {  this.ownerName = ownerName;  this.vehicleType = vehicleType;  this.registrationNumber = registrationNumber;  }    // Static method to update registration fee  public static void UpdateregistrationFee(double newFee)  {  registrationFee = newFee;  }   // Method to display vehicle details  public void DisplayDetails()  {  if (this is Vehicle)   {  Console.WriteLine($"Owner: {ownerName}");  Console.WriteLine($"Type: {vehicleType}");  Console.WriteLine($"Registration Number: {registrationNumber}");  Console.WriteLine($"Registration Fee: {registrationFee}");  }  }   public static void Main()  {  // Creating vehicle instances  Vehicle car = new Vehicle("Rohit", "Car", "ABC1234");  Vehicle bike = new Vehicle("Rahul", "Bike", "XYZ5678");   car.DisplayDetails();  bike.DisplayDetails();    // Update Registration Fee  Vehicle.UpdateregistrationFee(5500);    Console.WriteLine("After Updating Registration Fee:");  car.DisplayDetails();  bike.DisplayDetails();  } } |
| --- |

**Sample Program 7: Hospital Management System**

Create a Patient class with the following features:

* **static**:
  + A static variable HospitalName shared among all patients.
  + A static method GetTotalPatients() to count the total patients admitted.
* **this**:
  + Use this to initialize Name, Age, and Ailment in the constructor.
* **readonly**:
  + Use a readonly variable PatientID to uniquely identify each patient.
* **is operator**:
  + Check if an object is an instance of the Patient class before displaying its details.

| using System;  class Patient {  // Static variable   public static string hospitalName = "City Hospital";  private static int totalPatients = 0;    // Readonly variable  public readonly int patientID;    // Instance variables  public string name;  public int age;  public string ailment;   // Constructor   public Patient(string name, int age, string ailment)  {  this.name = name;  this.age = age;  this.ailment = ailment;  this.patientID = ++totalPatients;  }   // Static method to get total patients count  public static int GetTotalPatients()  {  return totalPatients;  }   // Method to display patient details  public void DisplayDetails()  {  if (this is Patient)   {  Console.WriteLine($"Hospital: {hospitalName}");  Console.WriteLine($"Patient ID: {patientID}");  Console.WriteLine($"name: {name}");  Console.WriteLine($"age: {age}");  Console.WriteLine($"ailment: {ailment}");  }  }    public static void Main()  {  // Creating patient instances  Patient patient1 = new Patient("Rohit", 24, "Flu");  Patient patient2 = new Patient("Ramkumar", 45, "Joint Pain");    // Display details  patient1.DisplayDetails();  patient2.DisplayDetails();    // Get total patients count  Console.WriteLine($"Total Patients Admitted: {Patient.GetTotalPatients()}");  } } |
| --- |