C# Assignments on Classes & Objects

Assignment 1: Circle Class

Problem Statement: Create a Circle class with a property for Radius. Implement a getter to retrieve the radius and a setter to ensure that the radius cannot be negative.

Assignment 2: Employee Class

Problem Statement: Design an Employee class with properties for Name and Salary. Use getters and setters to manage access to the salary, ensuring it cannot be set to a negative value.

Assignment 3: Library Management System

Problem Statement: Design a library management system that manages books. Each book has a title, author, and ISBN number. The system should allow adding a book, removing a book, and displaying all books.

Assignment 4: Banking System

Problem Statement: Create a simple banking system that allows account creation and basic transactions. Each account has an account number, account holder name, and balance. Implement deposit and withdrawal methods. Use getters and setters to manage access to the balance, ensuring it cannot be set to a negative value.

Assignment 5: Student Management System

Problem Statement: Develop a student management system that stores student details. Each student has a name, ID, and a list of grades. Implement methods to add a grade and calculate the average grade.

Assignment 6: Inventory System

Problem Statement: Create an inventory management system that manages items in a store. Each item has a name, stock, and price. Implement methods to add, remove, and update items. Use getters and setters to manage access to the stock and price, ensuring it cannot be set to a negative value.

Assignment 7: E-commerce System

Problem Statement: Design an e-commerce system that manages products and orders. Each product has a name, price, and stock quantity. Implement methods to create an order that reduces stock quantity. Implement getters and setters to ensure that the price cannot be negative and the stock cannot be less than zero.

Assignment 8: Print Class

Problem Statement: Design a Print class that contains overloaded methods to print different types of data: a string, an integer, and an array of integers.

Assignment 9: Rectangle Class

Problem Statement: Create a Rectangle class with overloaded methods to calculate the area. Implement methods that take either width and height or just one side length (for a square).

Assignment 10: Time Class

Problem Statement: Create a Time class with overloaded methods to set the time. Implement methods to set the time using hours and minutes, and another method to set the time using seconds.

Static and Instance Blocks

Assignment 11: Initializing a Static Field

Problem Statement: Create a class Bank that has a static field for the interest rate and a non-static field for the account holder's balance. Write a static constructor to initialize the interest rate to a default value and a regular constructor for setting up the account balance.

Assignment 12: Counting Objects with Static and Instance Fields

Problem Statement: Create a class Car that counts how many instances of Car have been created using a static counter. Initialize this counter using a static constructor.

Assignment 13: Initializing Constants with Static Block

Problem Statement: Create a class MathOperations that initializes a static field representing the value of Pi. Write a static constructor to assign this value.

Assignment 14: Initializing Configuration with Static Constructor

Problem Statement: Create a class Configuration to load system-wide settings (e.g., application name) using a static constructor. Allow individual users to set preferences using an instance constructor.

Assignment15: Implementing and Understanding Copy Constructor

Problem Statement:

Write a C# program that implements a copy constructor. The program should:

1. Create a class with several fields.
2. Provide a constructor to initialize those fields.
3. Provide a copy constructor that allows the creation of a new object from an existing object.
4. Demonstrate how the copy constructor works by comparing objects created using it with objects created via direct assignment (which just copies references).

Assignment 16: Identifying the Need for Chained Constructors

Tasks:

1. Create a class named Car with the following:
    - Fields for make, model, year, and price.
    - Multiple constructors:
        - A constructor that initializes only the make.
        - A constructor that initializes make and model.
        - A constructor that initializes make, model, and year.
        - A constructor that initializes all fields: make, model, year, and price.
    - Use **constructor chaining** to avoid duplicating the logic for initializing fields.
2. In the Main() method:
    - Create several Car objects using different constructors.
    - Display the details of each car to verify that all fields are initialized correctly.

Assignment 17: Understanding the Need for Constructor Overloading

Problem Statement:

Write a C# program that models a **Product** class with overloaded constructors. The class should:

1. Provide flexibility in product initialization based on the availability of price and discount information.
2. Use constructor overloading to handle cases where only basic product information is available, as well as cases where detailed information (price and discount) is provided.

Tasks:

1. Create a class named Product with the following:
    - Fields for name, price, and discount.
    - Three constructors:
        - A constructor that initializes only the name.

        - A constructor that initializes name and price.
        - A constructor that initializes name, price, and discount.
2. Implement a method CalculateFinalPrice() that computes the final price after applying the discount (if applicable).
3. In the Main() method:
    - Create different Product objects using various constructors.
    - Display the details of each product, including the final price after any applicable discount.

Assignment 18: Exploring Different Ways to Initialize Objects

Problem Statement:

Write a C# program that demonstrates different ways to initialize an object of a class. The class should represent a **Product** with properties such as `Name`, `Price`, and `Category`. Implement the following methods of object initialization:

1. Constructor initialization.
2. Object initializer syntax.
3. Static factory method.
4. Anonymous types.
5. Reflection.
6. Default values in constructors.

Tasks:

1. Create a class named Product with the following:
    - Properties for Name, Price, and Category.
    - A constructor that initializes all three properties.
    - A static method to create a Product object.
    - Use reflection to dynamically create a Product object.
    - Implement a constructor that provides default values for the properties.
2. In the Main() method:
    - Create instances of the Product class using the different initialization techniques mentioned above.
    - Display the details of each product.

Assignment 19: Exploring Initialization Using Tuples and Records

Problem Statement:

Write a C# program that demonstrates object initialization using **tuples** and **records**. Create a simple model for **Student** with properties like `Name`, `Age`, and `Grade`. Use tuples and records to initialize and work with this model.

Tasks:

1. Create a Student class using the **record** keyword with properties Name, Age, and Grade.
2. Use tuples to store and retrieve student details.
3. Create a method that accepts a tuple as a parameter and returns a Student record.
4. Display the details of the students.

Assignment 20: Shopping Cart

Problem Statement:

You need to create a Shopping Cart class that holds a list of Product objects. The Product class will be a nested class. The system should allow users to add products to the cart and display the total price.

Tasks:

1. Create a ShoppingCart class that contains:
   - A list of Product objects.
   - Methods to add products and calculate the total price.
2. Create a nested Product class with properties for Name, Price, and Quantity.
3. Demonstrate adding products and displaying the total price in the Main() method.

Assignment 21: Banking System

Problem Statement:

You need to create a Bank class that contains a list of Customer objects. Each Customer can have multiple Account objects (nested class). Implement methods to add customers, add accounts, and display customer account details.

Tasks:

1. Create a Bank class with:
   - A list of Customer objects.
   - Methods to add customers and accounts, and to display customer details.
2. Create a nested Customer class with properties for Name and a list of accounts.
3. Create a nested Account class with properties for AccountNumber and Balance.
4. Demonstrate the functionality in the Main() method.

Assignment 22: University System

Problem Statement:

You need to create a University class that holds a list of Department objects. Each Department can have multiple Course objects (nested class). Implement methods to add departments, add courses, and display course information.

Tasks:

1. Create a University class with:
    - A list of Department objects.
    - Methods to add departments and courses, and to display course details.
2. Create a nested Department class with properties for Name and a list of courses.
3. Create a nested Course class with properties for CourseName, CourseCode, and Credits.
4. Demonstrate the functionality in the Main() method.