# National Textile University

## Department of Computer Science

## Final Lab: Object Oriented Programming- COC2071

## Instructor: Abdul Qadeer Bilal

**Submission = Visual Studio Project Zip Files (2 zip files for 2 Questions**

### QUESTION 01: (04 Marks)

**Note:** Question 01 is a scenario that involves classes, inheritance, method overloading, and constructor overloading.

**Scnario Question: Car Dealership System**

You are tasked with designing a class hierarchy for a car dealership system. The system should support various types of vehicles, including cars and motorcycles. Each vehicle has common properties like `VehicleId`, `Make`, `Model`, `Year`, `Price`, and `InStock`. Additionally, there are specific details for each vehicle type.

**1. Vehicle Class:**

- Create a base class called `Vehicle` with common properties and a parameterized constructor to initialize these properties.

- Implement a method called `DisplayVehicleInfo()` that displays information about the vehicle.

**2. Car Class (Inherits from Vehicle):**

- Create a derived class called `Car` that inherits from the `Vehicle` class.

- Add properties specific to cars, such as `NumDoors` and `TrunkCapacity`.

- Implement a parameterized constructor for the `Car` class that initializes both common and specific properties.

- Override the `DisplayVehicleInfo()` method to include car-specific information.

**3. Motorcycle Class (Inherits from Vehicle):**

- Create another derived class called `Motorcycle` that inherits from the `Vehicle` class.

- Add properties specific to motorcycles, such as `HasFairing` and `FuelTankCapacity`.

- Implement a parameterized constructor for the `Motorcycle` class that initializes both common and specific properties.

- Override the `DisplayVehicleInfo()` method to include motorcycle-specific information.

**4. Method Overloading:**

- Implement method overloading in the `Vehicle` class for the `DisplayVehicleInfo()` method. Create different overloads to display basic information, detailed information, and information specific to either cars or motorcycles.

**5. Constructor Overloading:**

- Implement constructor overloading in the `Vehicle`, `Car`, and `Motorcycle` classes to provide flexibility when creating instances of these classes. For example, create constructors with different parameter sets to allow for varying levels of detail during instantiation.

**Main Program:**

The ID of the Vahicle would be static and auto Generated. In main create objects of appropriate classes and perform execution of all steps.

Your goal is to create a class hierarchy that is well-organized and allows for easy extension in the future. Demonstrate the use of method overloading and constructor overloading to enhance the flexibility and usability of your classes.

## QUESTION 02: (06 Marks)

**Case Study: Inventory Management System**

You are tasked with developing an Inventory Management System for a small business that deals with various products. The system should allow the addition, removal, and display of products in the inventory. Each product has a unique identifier, a name, a price, and a quantity.

Requirements:

**1. Product Class:**

- Create a class named `Product` with the following attributes:

  - `ProductId` (string)

  - `ProductName` (string)

  - `Price` (decimal)

  - `Quantity` (int)

**2. Abstract Class:**

- Create an abstract class named `InventoryItem` with the following abstract methods:

  - `AddToInventory()` - Should add a product to the inventory.

  - `SellFromInventory()` - Should sell a specified quantity of a product from the inventory.

  - `DisplayInventory()` - Should display the current state of the inventory.

**3. Interfaces:**

- Create two interfaces named `IStorable` and `ISellable` with the following methods:

  - `Store()` in `IStorable` - Should store the product in the inventory.

  - `Sell()` in `ISellable` - Should sell the product.

**4. Inventory Manager Class:**

- Create a class named `InventoryManager` that inherits from `InventoryItem` and implements the `IStorable` and `ISellable` interfaces.

- Implement the abstract methods from the `InventoryItem` class to manage the inventory.

- Implement the `Store()` and `Sell()` methods from the interfaces to handle storing and selling products.

**5. Console Application:**

- Create a simple console application to test your inventory management system.

- Allow the user to interactively add products to the inventory, sell products, and display the current state of the inventory.

-Atleast add 2 products to the inventory from the Main program and then sell out of them and demonstrate current inventory

- Make the ID of the Products as Statis and it should be auto Generated.

Your solution should demonstrate proper use of classes, abstract classes, and interfaces. Consider edge cases such as trying to sell more products than available in the inventory.

Code Outline:

```
// Product class representing a generic product
public class Product
{
    // Properties

    // Constructor

    // Method to auto-generate ProductId
}

// Abstract class representing an inventory item
public abstract class InventoryItem
{
    //Code
}

// Interfaces for storing and selling products
public interface IStorable
{
    //Code
}

public interface ISellable
```

```
{
    //Code
}

// InventoryManager class implementing InventoryItem, IStorable, and ISellable
public class InventoryManager : InventoryItem, IStorable, ISellable
{
//Code
}

class Program
{
    static void Main()
    {
        //Code
    }
}
```

**Good Luck**