## Set A Solution

1. **Write a program to convert a given value based on user choice: kilometers to miles, pounds to kilograms, or liters to gallons, using switch case the formulas provided.**
   **Miles = Kilometers * 0.621371**
   **Kilograms = Pounds * 0.453592**
   **Gallons = Liters * 0.264172**

```
using System;

public class UnitConverter
{
    public static void Main(string[] args)
    {
        Console.WriteLine("Choose a conversion type:");
        Console.WriteLine("1. Kilometers to Miles");
        Console.WriteLine("2. Pounds to Kilograms");
        Console.WriteLine("3. Liters to Gallons");

        // Take user choice
        int choice = Convert.ToInt32(Console.ReadLine());

        // Variable to store the value to be converted
        double value, convertedValue;

        // Perform conversion based on user choice
        switch (choice)
        {
            case 1:
                Console.Write("Enter distance in kilometers: ");
                value = Convert.ToDouble(Console.ReadLine());
                convertedValue = value * 0.621371;
                Console.WriteLine($"{value} kilometers is equal to {convertedValue} miles.");
                break;

            case 2:
                Console.Write("Enter weight in pounds: ");
                value = Convert.ToDouble(Console.ReadLine());
                convertedValue = value * 0.453592;
                Console.WriteLine($"{value} pounds is equal to {convertedValue} kilograms.");
                break;

            case 3:
                Console.Write("Enter volume in liters: ");
                value = Convert.ToDouble(Console.ReadLine());
                convertedValue = value * 0.264172;
                Console.WriteLine($"{value} liters is equal to {convertedValue} gallons.");
                break;

            default:
```

```
            Console.WriteLine("Invalid choice! Please choose a valid conversion type.");
            break;
        }
    }
}
```

**2. Write a program that accepts two strings from the user and uses string methods to compare them (using Equals()), concatenate them (Concat()), and convert one string to uppercase (ToUpper()). Handle any exceptions if an invalid input is provided.**

```
public class StringOperations
{
    public static void Main(string[] args)
    {
        try
        {
            // Accepting two strings from the user
            Console.Write("Enter the first string: ");
            string firstString = Console.ReadLine();

            Console.Write("Enter the second string: ");
            string secondString = Console.ReadLine();

            // Compare strings using Equals()
            bool areStringsEqual = firstString.Equals(secondString);
            Console.WriteLine($"Are the two strings equal? {areStringsEqual}");

            // Concatenate strings using Concat()
            string concatenatedString = string.Concat(firstString, secondString);
            Console.WriteLine($"Concatenated String: {concatenatedString}");

            // Convert the first string to uppercase using ToUpper()
            string upperCaseString = firstString.ToUpper();
            Console.WriteLine($"First string in uppercase: {upperCaseString}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }
}
```

**3. Write a program using method overloading to create a class Calculator with methods Multiply(int a, int b), Multiply(double a, double b), and Multiply(float a, float b) to perform multiplication of two integers, two doubles, and two floats respectively.**

```
using System;

class Calculator
{
    // Method to multiply two integers
    public int Multiply(int a, int b)
    {
        return a * b;
    }

    // Method to multiply two doubles
    public double Multiply(double a, double b)
    {
        return a * b;
    }

    // Method to multiply two floats
    public float Multiply(float a, float b)
    {
        return a * b;
    }
}

class Program
{
    static void Main(string[] args)
    {
        Calculator calculator = new Calculator();

        // Multiplying integers
        int intResult = calculator.Multiply(4, 5);
        Console.WriteLine($"Multiplication of integers: 4 * 5 = {intResult}");

        // Multiplying doubles
        double doubleResult = calculator.Multiply(4.5, 5.5);
        Console.WriteLine($"Multiplication of doubles: 4.5 * 5.5 = {doubleResult}");

        // Multiplying floats
        float floatResult = calculator.Multiply(4.5f, 5.5f);
        Console.WriteLine($"Multiplication of floats: 4.5 * 5.5 = {floatResult}");
    }
}
```

**4. Write a program to create a class Employee with data members Name, ID, Position, and Salary. Create a function GetEmployeeDetails() to accept data and a DisplayEmployeeDetails() function to display the details of the employee.**

```
using System;
public class Employee
{
    // Data members
    public string Name { get; set; }
    public int ID { get; set; }
    public string Position { get; set; }
    public double Salary { get; set; }

    // Function to accept employee details
    public void GetEmployeeDetails()
    {
        Console.Write("Enter employee name: ");
        Name = Console.ReadLine();

        Console.Write("Enter employee ID: ");
        ID = Convert.ToInt32(Console.ReadLine());

        Console.Write("Enter employee position: ");
        Position = Console.ReadLine();

        Console.Write("Enter employee salary: ");
        Salary = Convert.ToDouble(Console.ReadLine());
    }
    // Function to display employee details
    public void DisplayEmployeeDetails()
    {
        Console.WriteLine("\nEmployee Details:");
        Console.WriteLine($"Name: {Name}");
        Console.WriteLine($"ID: {ID}");
        Console.WriteLine($"Position: {Position}");
        Console.WriteLine($"Salary: {Salary:C}");  // Displays salary in currency format
    }
}
public class Program
{
    public static void Main(string[] args)
    {
        // Create an Employee object
        Employee employee = new Employee();

        // Get employee details
        employee.GetEmployeeDetails();

        // Display employee details
        employee.DisplayEmployeeDetails();
    }
}
```

**5. Write a program to create a base class Person with attributes Name and Age, then derive two classes Employee and Manager, where Employee adds EmployeeID and Salary, and Manager adds Department, overriding the DisplayDetails() method to show all details.**

```
using System;

class Person
{
    public string Name { get; set; }
    public int Age { get; set; }

    public virtual void DisplayDetails()
    {
        Console.WriteLine($"Name: {Name}");
        Console.WriteLine($"Age: {Age}");
    }
}

class Employee : Person
{
    public int EmployeeID { get; set; }
    public double Salary { get; set; }

    public override void DisplayDetails()
    {
        base.DisplayDetails();
        Console.WriteLine($"Employee ID: {EmployeeID}");
        Console.WriteLine($"Salary: {Salary:C}");
    }
}

class Manager : Employee
{
    public string Department { get; set; }

    public override void DisplayDetails()
    {
        base.DisplayDetails();
        Console.WriteLine($"Department: {Department}");
    }
}

class Program
{
    static void Main(string[] args)
    {
        // Create an Employee object
        Employee employee = new Employee
        {
            Name = "Alice",
            Age = 30,
            EmployeeID = 12345,
```

```
            Salary = 50000
        };

        // Create a Manager object
        Manager manager = new Manager
        {
            Name = "Bob",
            Age = 40,
            EmployeeID = 67890,
            Salary = 75000,
            Department = "IT"
        };

        // Display details of Employee
        Console.WriteLine("Employee Details:");
        employee.DisplayDetails();

        Console.WriteLine("\nManager Details:");
        // Display details of Manager
        manager.DisplayDetails();
    }
}
```

6. **Design a program with a** Vehicle **class having attributes** Model**,** Brand**, and** Speed**, and derive** Car **and** Bike **classes with additional attributes** FuelType **and** WheelType**, implementing constructors and overriding** DisplayDetails **to display respective details**

```
using System;

class Vehicle
{
    public string Model { get; set; }
    public string Brand { get; set; }
    public int Speed { get; set; }

    public virtual void DisplayDetails()
    {
        Console.WriteLine($"Model: {Model}");
        Console.WriteLine($"Brand: {Brand}");
        Console.WriteLine($"Speed: {Speed} km/h");
    }
}

class Car : Vehicle
{
    public string FuelType { get; set; }

    public Car(string model, string brand, int speed, string fuelType)
    {
        Model = model;
        Brand = brand;
        Speed = speed;
```

```
        FuelType = fuelType;
    }

    public override void DisplayDetails()
    {
        Console.WriteLine($"Car Details:");
        Console.WriteLine($"Model: {Model}");
        Console.WriteLine($"Brand: {Brand}");
        Console.WriteLine($"Speed: {Speed} km/h");
        Console.WriteLine($"Fuel Type: {FuelType}");
    }
}

class Bike : Vehicle
{
    public string WheelType { get; set; }

    public Bike(string model, string brand, int speed, string wheelType)
    {
        Model = model;
        Brand = brand;
        Speed = speed;
        WheelType = wheelType;
    }
    public override void DisplayDetails()
    {
        Console.WriteLine($"Bike Details:");
        Console.WriteLine($"Model: {Model}");
        Console.WriteLine($"Brand: {Brand}");
        Console.WriteLine($"Speed: {Speed} km/h");
        Console.WriteLine($"Wheel Type: {WheelType}");
    }
}
class Program
{
    static void Main(string[] args)
    {
        // Create a Car object
        Car car = new Car("Model S", "Tesla", 200, "Electric");

        // Create a Bike object
        Bike bike = new Bike("Ninja ZX-10R", "Kawasaki", 300, "Sport");

        // Display details of Car
        car.DisplayDetails();

        Console.WriteLine();

        // Display details of Bike
        bike.DisplayDetails();
    }
}
```

**7. Write a program to create an abstract class Employee with abstract methods CalculateSalary() and DisplayEmployeeDetails(). Create a derived class FullTimeEmployee that implements these methods and calculates the salary based on hours worked and hourly rate. Demonstrate the functionality by creating an instance of FullTimeEmployee and displaying the details.**

```
using System;

abstract class Employee
{
    public string Name { get; set; }
    public int EmployeeID { get; set; }

    // Abstract methods to be implemented by derived classes
    public abstract void CalculateSalary(int hoursWorked, double hourlyRate);
    public abstract void DisplayEmployeeDetails();
}

class FullTimeEmployee : Employee
{
    public double Salary { get; private set; }

    // Implementation of CalculateSalary
    public override void CalculateSalary(int hoursWorked, double hourlyRate)
    {
        Salary = hoursWorked * hourlyRate;
    }

    // Implementation of DisplayEmployeeDetails
    public override void DisplayEmployeeDetails()
    {
        Console.WriteLine($"Employee ID: {EmployeeID}");
        Console.WriteLine($"Name: {Name}");
        Console.WriteLine($"Salary: {Salary:C}");
    }
}

class Program
{
    static void Main(string[] args)
    {
        // Create an instance of FullTimeEmployee
        FullTimeEmployee employee = new FullTimeEmployee
        {
            EmployeeID = 101,
            Name = "John Doe"
        };

        // Calculate salary
        int hoursWorked = 160; // Example: 160 hours in a month
        double hourlyRate = 50; // Example: $50 per hour
        employee.CalculateSalary(hoursWorked, hourlyRate);
```

```
        // Display employee details
        employee.DisplayEmployeeDetails();
    }
}
```