Building an Automobile Management Application using Windows Forms and ADO.NET

Introduction

Imagine you're an employee of a car retailer named **Automobile Store**. Your manager has asked you to develop a Windows Forms application for automobile management (CarID, CarName, Manufacturer, Price, and ReleasedYear). The application has to support adding, viewing, modifying, and removing products—a standardized usage action verbs better known as Create, Read, Update, Delete (CRUD).

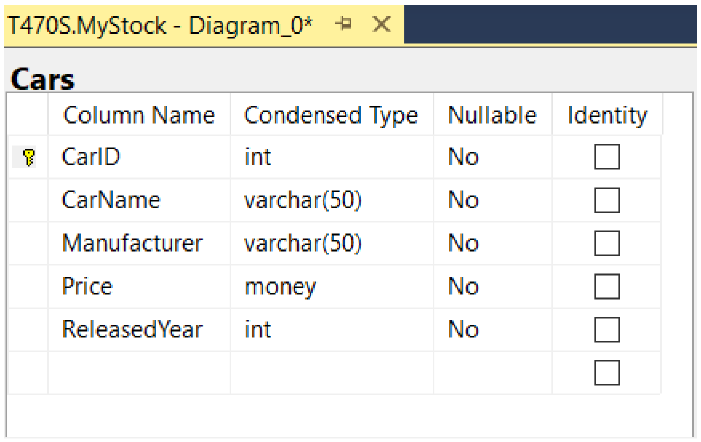
This lab explores creating an application using Windows Forms with .NET Core, and C#. An **SQL Server Database** will be created to persist the car's data that will be used for reading and managing automobile data by **ADO.NET**

Lab Objectives

In this lab, you will:

- Use the Visual Studio.NET to create Windows Forms and Class Library (.dll) project.
- Create a SQL Server database named MyStock that has a Cars table.
- Develop a DataProvider class to perform CRUD actions using ADO.NET
- Apply passing data in WinForms application
- Apply Repository pattern and Singleton pattern in a project
- Add CRUD action methods to WinForms application
- Run the project and test the WinForms actions.

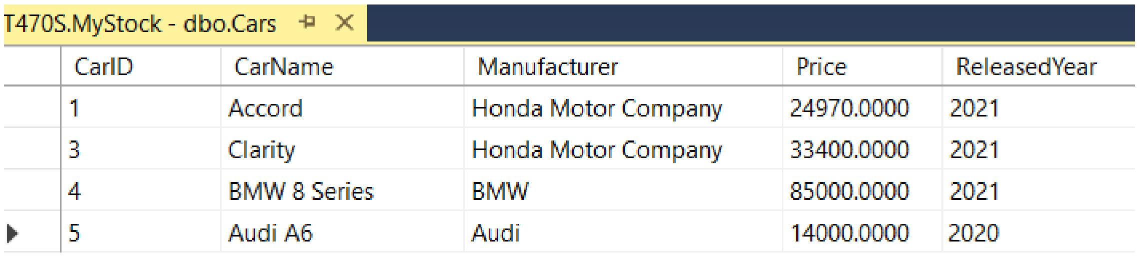
MyStock Database



T470S.MyStock - Diagram_0*

Cars

	Column Name	Condensed Type	Nullable	Identity
🔑	CarID	int	No	☐
	CarName	varchar(50)	No	☐
	Manufacturer	varchar(50)	No	☐
	Price	money	No	☐
	ReleasedYear	int	No	☐
				☐



T470S.MyStock - dbo.Cars

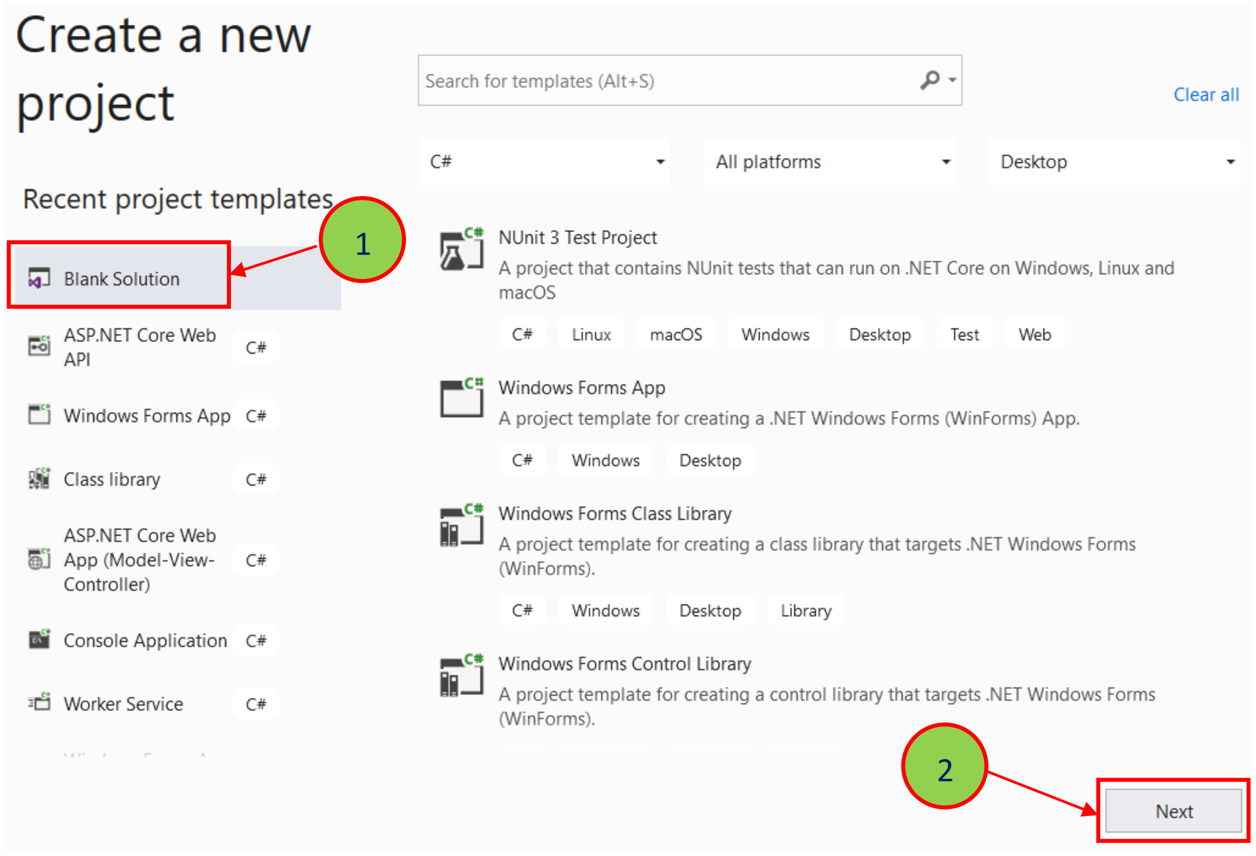
	CarID	CarName	Manufacturer	Price	ReleasedYear
	1	Accord	Honda Motor Company	24970.0000	2021
	3	Clarity	Honda Motor Company	33400.0000	2021
	4	BMW 8 Series	BMW	85000.0000	2021
▶	5	Audi A6	Audi	14000.0000	2020

Activity 01: Build a solution by Visual Studio.NET

Create a Blank Solution named **AutomobileSolution** then add new a Class Library Project named **AutomobileLibrary** and a Windows Forms project named **AutomobileWinApp**

Step 01. Create a Blank solution.

- Open the Visual Studio .NET application and performs steps as follows:



Create a new project

Search for templates (Alt+S)

Clear all

C# | All platforms | Desktop

Recent project templates

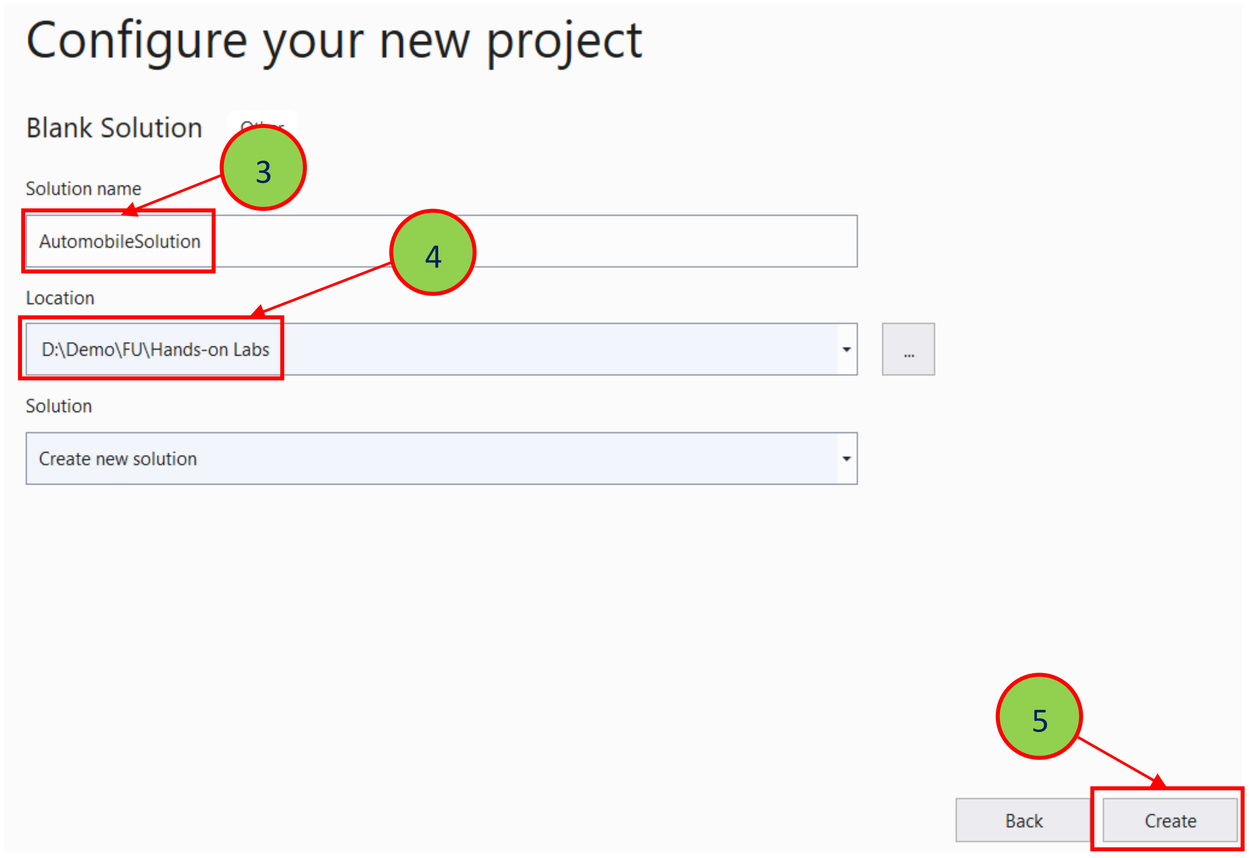
- Blank Solution
- ASP.NET Core Web API C#
- Windows Forms App C#
- Class library C#
- ASP.NET Core Web App (Model-View-Controller) C#
- Console Application C#
- Worker Service C#

NUnit 3 Test Project
A project that contains NUnit tests that can run on .NET Core on Windows, Linux and macOS
C# Linux macOS Windows Desktop Test Web

Windows Forms App
A project template for creating a .NET Windows Forms (WinForms) App.
C# Windows Desktop

Windows Forms Class Library
A project template for creating a class library that targets .NET Windows Forms (WinForms).
C# Windows Desktop Library

Windows Forms Control Library
A project template for creating a control library that targets .NET Windows Forms (WinForms).



Configure your new project

Blank Solution

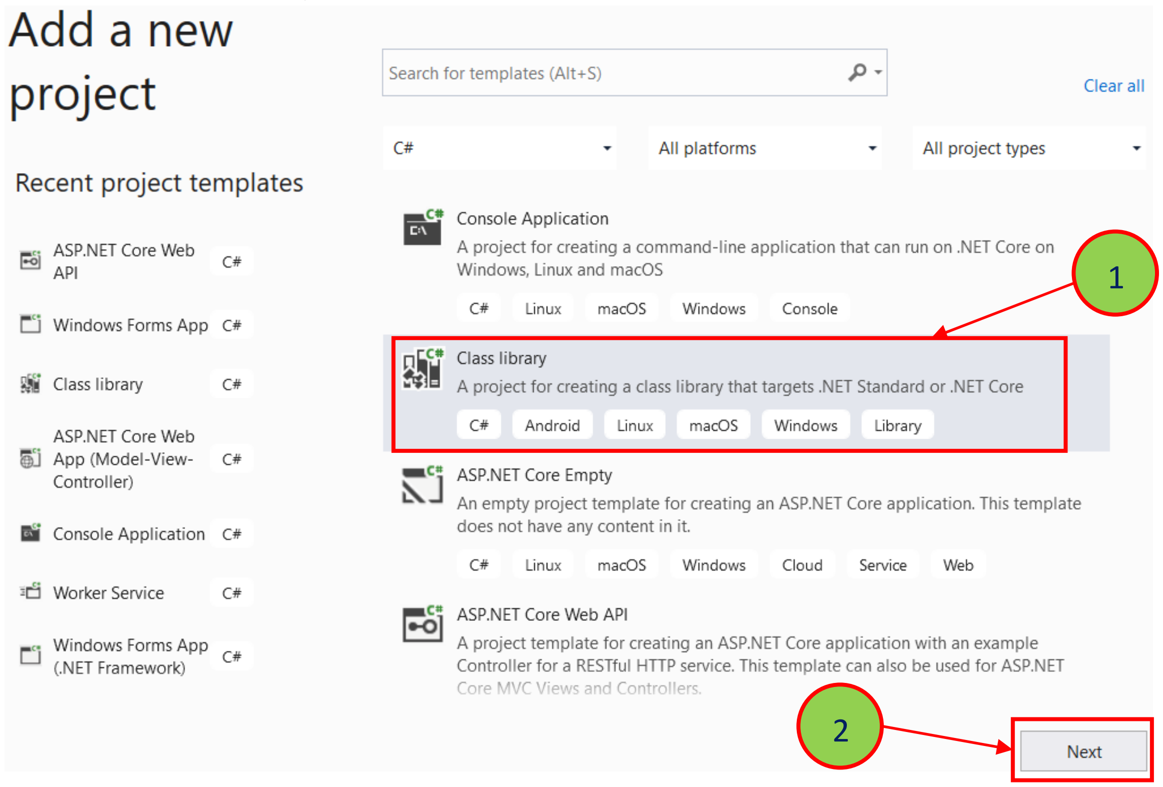
Solution name: AutomobileSolution

Location: D:\Demo\FU\Hands-on Labs

Solution: Create new solution

Step 02. Create a Class Library project.

- From the File menu | Add | New Project, on the Add New Project dialog, select "Class Library" and performs steps as follows:



Add a new project

Search for templates (Alt+S)

Clear all

C# · All platforms · All project types

Recent project templates

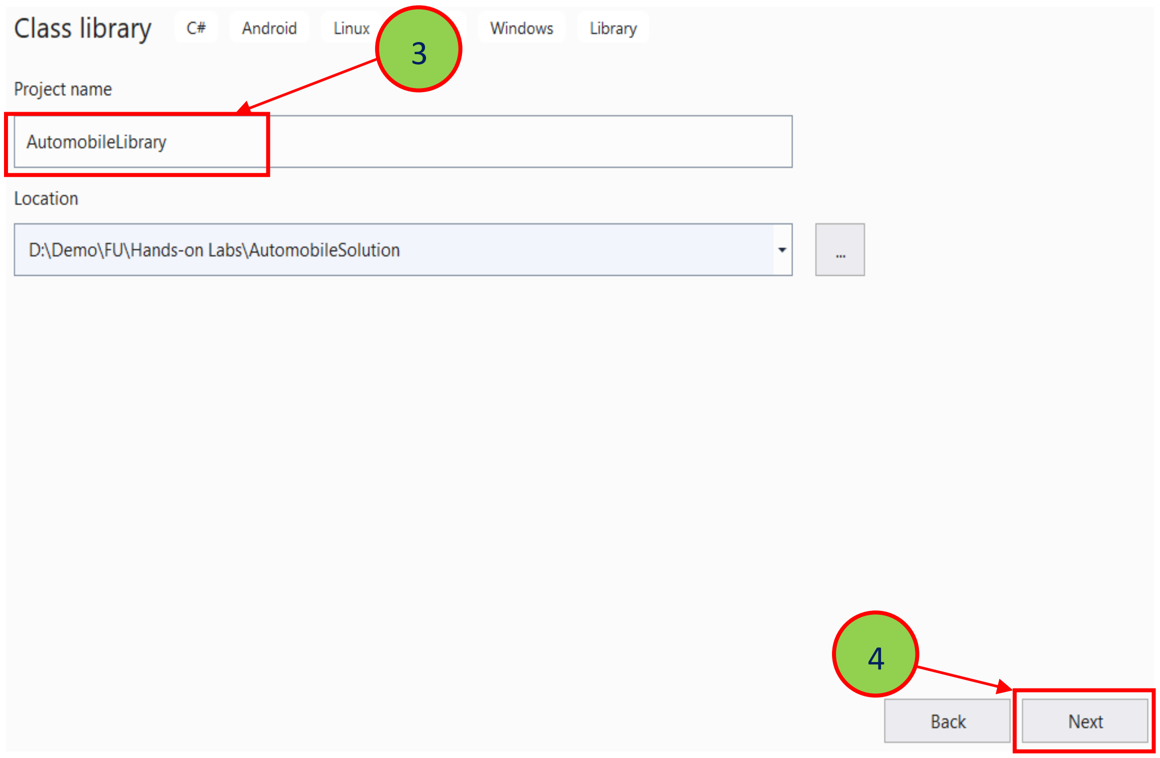
ASP.NET Core Web API	C#
Windows Forms App	C#
Class library	C#
ASP.NET Core Web App (Model-View-Controller)	C#
Console Application	C#
Worker Service	C#
Windows Forms App (.NET Framework)	C#

Console Application
A project for creating a command-line application that can run on .NET Core on Windows, Linux and macOS
C# Linux macOS Windows Console

Class library
A project for creating a class library that targets .NET Standard or .NET Core
C# Android Linux macOS Windows Library

ASP.NET Core Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
C# Linux macOS Windows Cloud Service Web

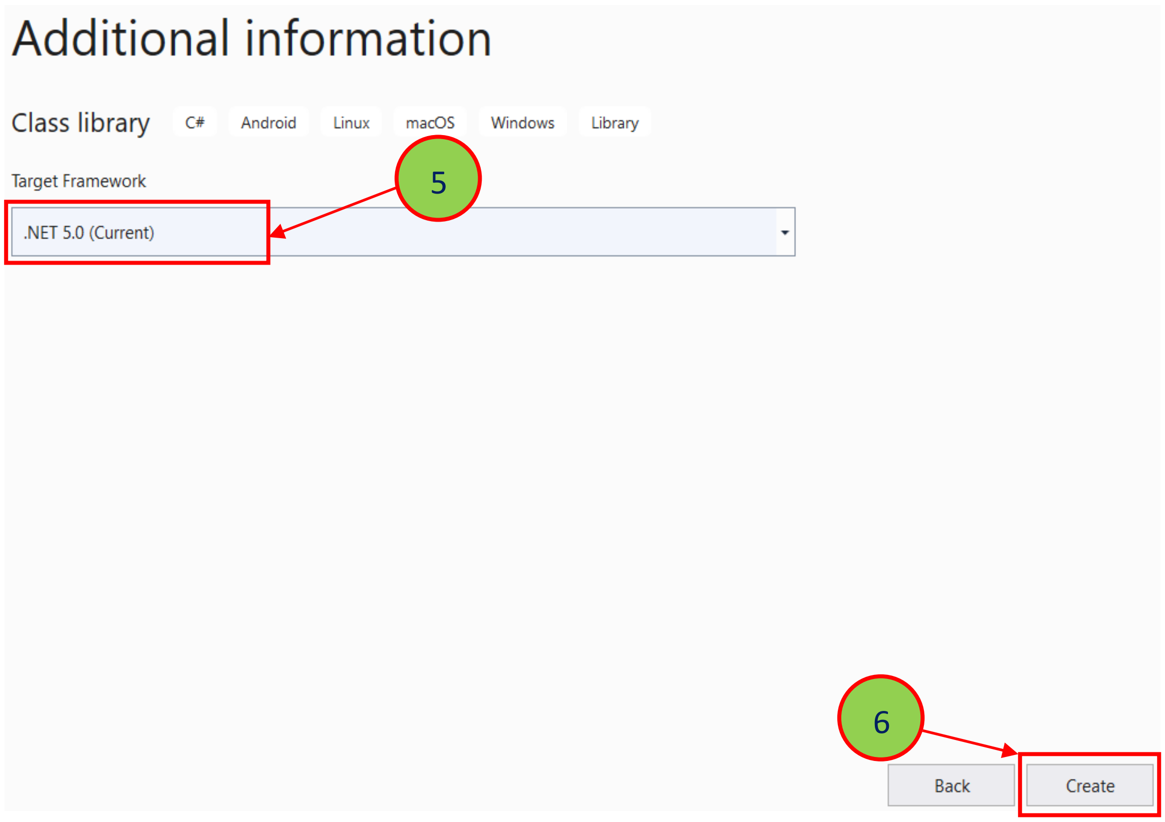
ASP.NET Core Web API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.



Class library · C# · Android · Linux · Windows · Library

Project name: AutomobileLibrary

Location: D:\Demo\FU\Hands-on Labs\AutomobileSolution



Additional information

Class library C# Android Linux macOS Windows Library

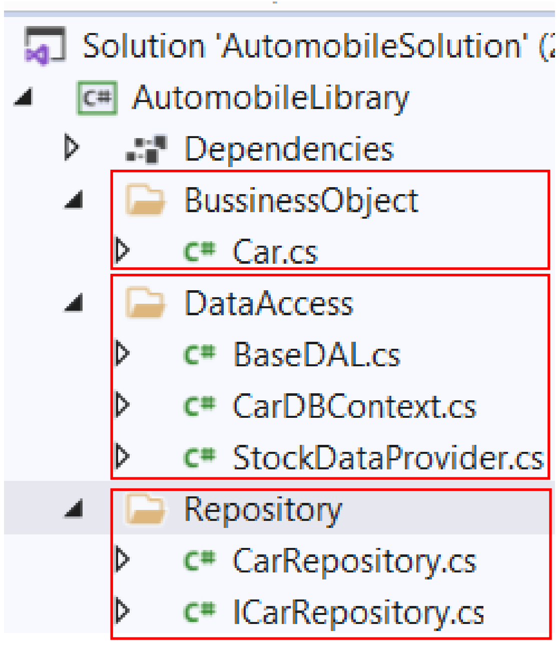
Target Framework

.NET 5.0 (Current)

Step 03. Repeat **Step 02** to create a Windows Form project.

Activity 02: Write codes for the AutomobileLibrary project

Step 01. Create folders and add classes to the project as follows:

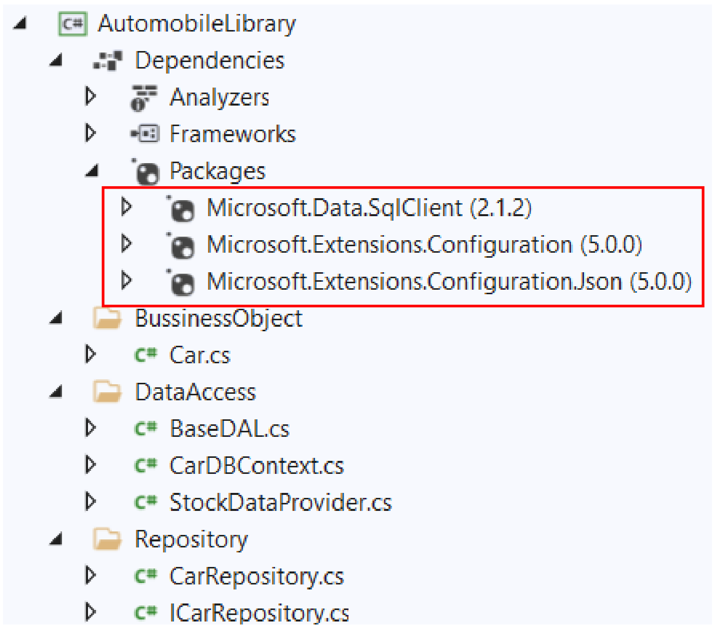


```
Solution 'AutomobileSolution' (
  AutomobileLibrary
    Dependencies
    BussinessObject
      Car.cs
    DataAccess
      BaseDAL.cs
      CarDBContext.cs
      StockDataProvider.cs
    Repository
      CarRepository.cs
      ICarRepository.cs
```

Step 02. Write codes for **Car.cs** as follows:

```
namespace AutomobileLibrary.BussinessObject
{
    public class Car
    {
        public int CarID { get; set; }
        public string CarName { get; set; }
        public string Manufacturer { get; set; }
        public decimal Price { get; set; }
        public int ReleaseYear { get; set; }
    }
}                                                    `
```

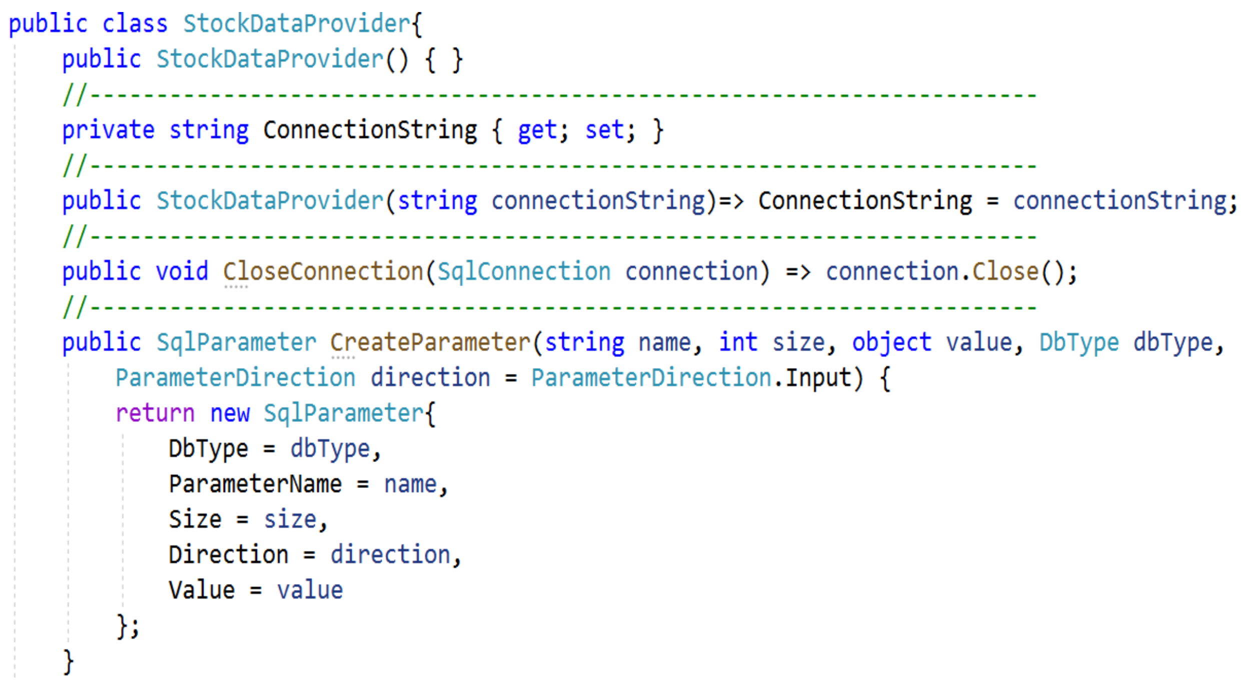
Step 03. Install the following packages from Nuget:



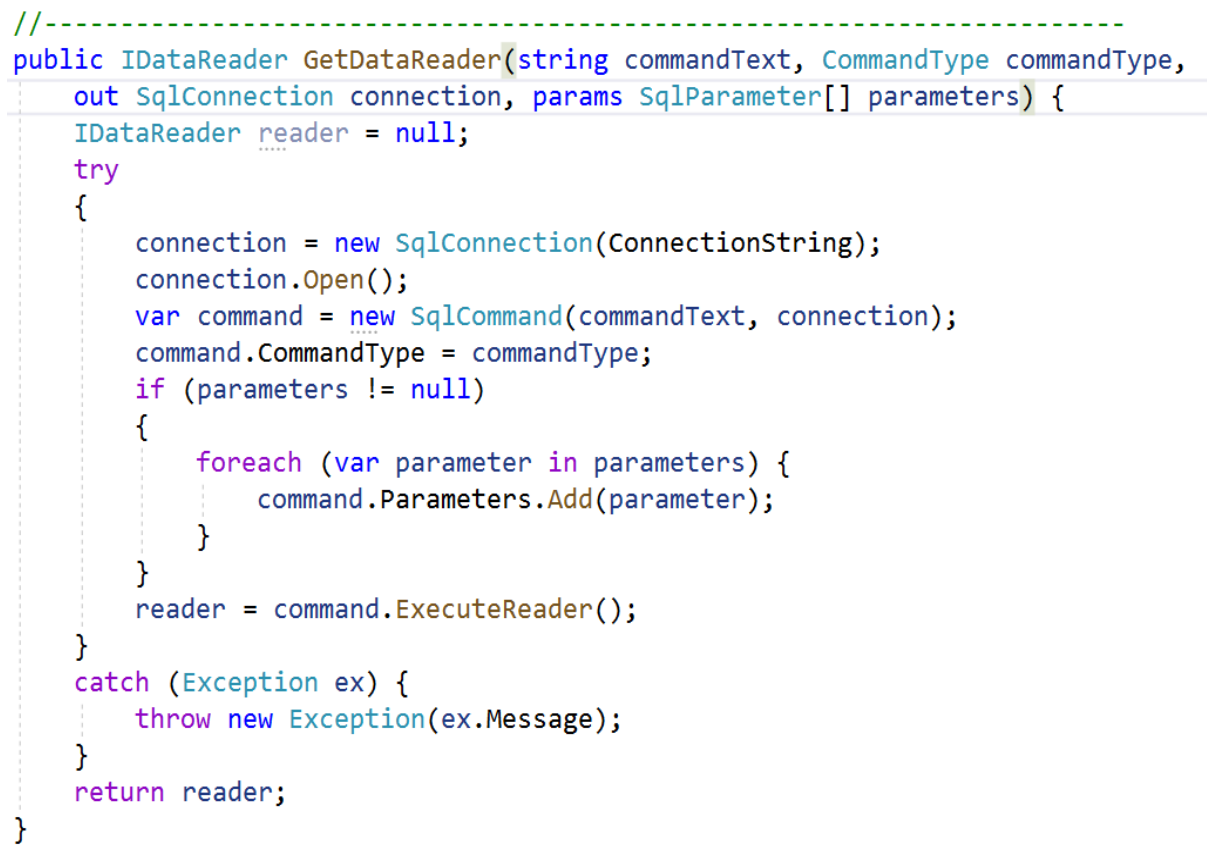
```
◢ AutomobileLibrary
    ◢ Dependencies
        ▷ Analyzers
        ▷ Frameworks
        ◢ Packages
            ▷ Microsoft.Data.SqlClient (2.1.2)
            ▷ Microsoft.Extensions.Configuration (5.0.0)
            ▷ Microsoft.Extensions.Configuration.Json (5.0.0)
    ◢ BussinessObject
        ▷ Car.cs
    ◢ DataAccess
        ▷ BaseDAL.cs
        ▷ CarDBContext.cs
        ▷ StockDataProvider.cs
    ◢ Repository
        ▷ CarRepository.cs
        ▷ ICarRepository.cs
```

Step 04. Write codes for **StockDataProvider.cs** as follows:

```
using System;
using System.Data;
using Microsoft.Data.SqlClient;

namespace AutomobileLibrary.DataAccess{
```



```
public class StockDataProvider{
    public StockDataProvider() { }
    //-----------------------------------------------------------------------------
    private string ConnectionString { get; set; }
    //-----------------------------------------------------------------------------
    public StockDataProvider(string connectionString)=> ConnectionString = connectionString;
    //-----------------------------------------------------------------------------
    public void CloseConnection(SqlConnection connection) => connection.Close();
    //-----------------------------------------------------------------------------
    public SqlParameter CreateParameter(string name, int size, object value, DbType dbType,
        ParameterDirection direction = ParameterDirection.Input) {
        return new SqlParameter{
            DbType = dbType,
            ParameterName = name,
            Size = size,
            Direction = direction,
            Value = value
        };
    }
```



```
//---------------------------------------------------------------------------------------
public IDataReader GetDataReader(string commandText, CommandType commandType,
    out SqlConnection connection, params SqlParameter[] parameters) {
    IDataReader reader = null;
    try
    {
        connection = new SqlConnection(ConnectionString);
        connection.Open();
        var command = new SqlCommand(commandText, connection);
        command.CommandType = commandType;
        if (parameters != null)
        {
            foreach (var parameter in parameters) {
                command.Parameters.Add(parameter);
            }
        }
        reader = command.ExecuteReader();
    }
    catch (Exception ex) {
        throw new Exception(ex.Message);
    }
    return reader;
}
```

```
//----------------------------------------------------------------------
public void Delete(string commandText, CommandType commandType,
    params SqlParameter[] parameters)  {
    try
    {
        using var connection = new SqlConnection(ConnectionString);
        connection.Open();
        using var command = new SqlCommand(commandText, connection);
        command.CommandType = commandType;
        if (parameters != null) {
            foreach (var parameter in parameters){
                command.Parameters.Add(parameter);
            }
        }
        command.ExecuteNonQuery();
    }
    catch (Exception ex) {
        throw new Exception("Data Provider:Delete Method", ex.InnerException);
    }
}

//----------------------------------------------------------------------
public void Insert(string commandText, CommandType commandType,
    params SqlParameter[] parameters) {
    try
    {
        using var connection = new SqlConnection(ConnectionString);
        connection.Open();
        using var command = new SqlCommand(commandText, connection);
        command.CommandType = commandType;
        if (parameters != null) {
            foreach (var parameter in parameters) {
                command.Parameters.Add(parameter);
            }
        }
        command.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message);
    }
}
```

```
    //-------------------------------------------------------------------
    public void Update(string commandText, CommandType commandType,
        params SqlParameter[] parameters) {
        try
        {
            using var connection = new SqlConnection(ConnectionString);
            connection.Open();
            using var command = new SqlCommand(commandText, connection);
            command.CommandType = commandType;
            if (parameters != null) {
                foreach (var parameter in parameters){
                    command.Parameters.Add(parameter);
                }
            }
            command.ExecuteNonQuery();
        }
        catch (Exception ex) {
            throw new Exception(ex.Message);
        }
    }
  }//end class
}//end namespace
```

<u>Step 05</u>. Write codes for **BaseDAL.cs** as follows:

```
using System.Data.SqlClient;
using System.IO;
using Microsoft.Data.SqlClient;
using Microsoft.Extensions.Configuration;

namespace AutomobileLibrary.DataAccess{
    public class BaseDAL {
        public StockDataProvider dataProvider { get; set; } = null;
        public SqlConnection connection = null;
        //-------------------------------------------------------------------
        public BaseDAL() {
            var connectionString = GetConnectionString();
            dataProvider = new StockDataProvider(connectionString);
        }
        //-------------------------------------------------------------------
        //------------------------------------------------------------------
        public string GetConnectionString(){
            string connectionString;
            IConfiguration config = new ConfigurationBuilder()
                                        .SetBasePath(Directory.GetCurrentDirectory())
                                        .AddJsonFile("appsettings.json", true, true)
                                        .Build();
            connectionString = config["ConnectionString:MyStockDB"];
            return connectionString;
        }
        //------------------------------------------------------------------
        public void CloseConnection()=>dataProvider.CloseConnection(connection);
    }//end class
}//end namespace
```

Step 06. Write codes for **CarDBContext.cs** as follows:

```
//add namespaces
using System.Data;
using AutomobileLibrary.BussinessObject;
using Microsoft.Data.SqlClient;
namespace AutomobileLibrary.DataAccess{
   public class CarDBContext : BaseDAL {
       //----------------------------------------------
       //Using Singleton Pattern
       private static CarDBContext instance = null;
       private static readonly object instanceLock = new object();
       private CarDBContext() { }
       public static CarDBContext Instance{
           get{
               lock (instanceLock){
                   if (instance == null){
                       instance = new CarDBContext();
                   }
                   return instance;
               }
           }
       }
```

```
//----------------------------------------------
public IEnumerable<Car> GetCarList() {
    IDataReader dataReader = null;
    string SQLSelect = "Select CarID, CarName, Manufacturer, Price, ReleasedYear from Cars";
    var cars = new List<Car>();
    try {
        dataReader = dataProvider.GetDataReader(SQLSelect, CommandType.Text, out connection);
        while (dataReader.Read()) {
            cars.Add(new Car{
                CarID = dataReader.GetInt32(0),
                CarName = dataReader.GetString(1),
                Manufacturer = dataReader.GetString(2),
                Price = dataReader.GetDecimal(3),
                ReleaseYear = dataReader.GetInt32(4)
            });
        }
    }
    catch (Exception ex){
        throw new Exception(ex.Message);
    }
    finally{
        dataReader.Close();
        CloseConnection();
    }
    return cars;
}

//----------------------------------------------
public Car GetCarByID(int carID){
    Car car = null;
    IDataReader dataReader = null;
    string SQLSelect = "Select CarID, CarName, Manufacturer, Price, ReleasedYear " +
        "  from Cars where CarID = @CarID";
    try{
        var param = dataProvider.CreateParameter("@CarID", 4, carID, DbType.Int32);
        dataReader = dataProvider.GetDataReader(SQLSelect, CommandType.Text, out connection, param);
        if (dataReader.Read()){
            car = new Car{  CarID = dataReader.GetInt32(0),
                CarName = dataReader.GetString(1), Manufacturer = dataReader.GetString(2),
                Price = dataReader.GetDecimal(3),
                ReleaseYear = dataReader.GetInt32(4)
            };
        }
    }
    catch (Exception ex){
        throw new Exception(ex.Message);
    }
    finally{
        dataReader.Close();
        CloseConnection();
    }
    return car;
}
```

```
//---------------------------------------------------------------
public void AddNew(Car car){
    try{
        Car pro = GetCarByID(car.CarID);
        if (pro == null){
            string SQLInsert = "Insert Cars values(@CarID,@CarName,@Manufacturer,@Price,@ReleasedYear)";
            var parameters = new List<SqlParameter>();
            parameters.Add(dataProvider.CreateParameter("@CarID", 4, car.CarID, DbType.Int32));
            parameters.Add(dataProvider.CreateParameter("@CarName", 50, car.CarName, DbType.String));
            parameters.Add(dataProvider.CreateParameter("@Manufacturer", 50, car.Manufacturer, DbType.String));
            parameters.Add(dataProvider.CreateParameter("@Price", 50, car.Price, DbType.Decimal));
            parameters.Add(dataProvider.CreateParameter("@ReleasedYear", 4, car.ReleaseYear, DbType.Int32));
            dataProvider.Insert(SQLInsert, CommandType.Text, parameters.ToArray());
        }
        else{
            throw new Exception("The car is already exist.");
        }
    }
    catch (Exception ex){
        throw new Exception(ex.Message);
    }
    finally{
        CloseConnection();
    }
}

//---------------------------------------------------------------
public void Update(Car car){
    try{
        Car c = GetCarByID(car.CarID);
        if (c != null){
            string SQLUpdate = "Update Cars set CarName = @CarName,Manufacturer = @Manufacturer," +
                "Price = @Price,ReleasedYear=@ReleasedYear where CarID=@CarID";
            var parameters = new List<SqlParameter>();
            parameters.Add(dataProvider.CreateParameter("@CarID", 4, car.CarID, DbType.Int32));
            parameters.Add(dataProvider.CreateParameter("@CarName", 50, car.CarName, DbType.String));
            parameters.Add(dataProvider.CreateParameter("@Manufacturer", 50, car.Manufacturer, DbType.String));
            parameters.Add(dataProvider.CreateParameter("@Price", 50, car.Price, DbType.Decimal));
            parameters.Add(dataProvider.CreateParameter("@ReleasedYear", 4, car.ReleaseYear, DbType.Int32));
            dataProvider.Update(SQLUpdate, CommandType.Text, parameters.ToArray());
        }
        else {
            throw new Exception("The car does not already exist.");
        }
    }
    catch (Exception ex){
        throw new Exception(ex.Message);
    }
    finally{
        CloseConnection();
    }
}
```

```
        //---------------------------------------------------------------
        public void Remove(int carID) {
            try{
                Car car = GetCarByID(carID);
                if (car != null){
                    string SQLDelete = "Delete Cars where CarID = @CarID";
                    var param = dataProvider.CreateParameter("@CarID", 4, carID, DbType.Int32);
                    dataProvider.Delete(SQLDelete, CommandType.Text, param);
                }
                else{
                    throw new Exception("The car does not already exist.");
                }
            }
            catch (Exception ex){
                throw new Exception(ex.Message);
            }
            finally{
                CloseConnection();
            }
        }//end Remove
    }//end class
}//end namespace
```

<u>Step 07</u>. Write codes for **ICarRepository.cs** as follows:

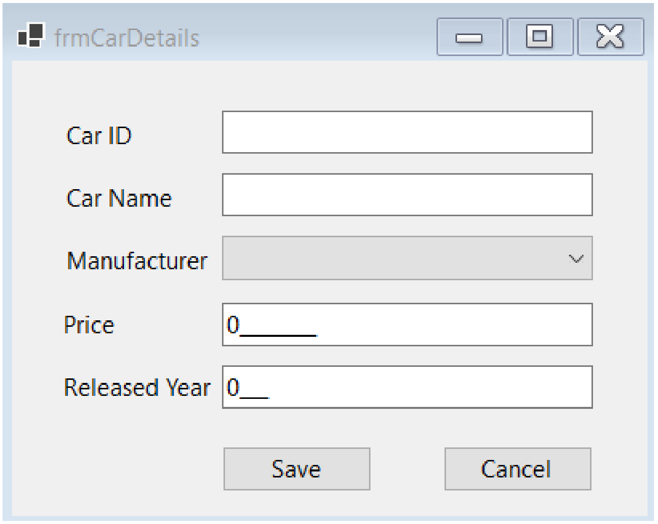
```
using System.Collections;
using AutomobileLibrary.BussinessObject;

namespace AutomobileLibrary.Repository
{
    public interface ICarRepository
    {
        IEnumerable<Car> GetCars();
        Car GetCarByID(int carId);
        void InsertCar(Car car);
        void DeleteCar(int carId);
        void UpdateCar(Car car);
    }
}
```

<u>Step 08</u>. Write codes for **CarRepository.cs** as follows:

```
using AutomobileLibrary.BussinessObject;
using AutomobileLibrary.DataAccess;

namespace AutomobileLibrary.Repository
{
    public class CarRepository : ICarRepository
    {
        public Car GetCarByID(int carId) => CarDBContext.Instance.GetCarByID(carId);

        public IEnumerable<Car> GetCars() => CarDBContext.Instance.GetCarList;

        public void InsertCar(Car car) => CarDBContext.Instance.AddNew(car);

        public void DeleteCar(int carId) => CarDBContext.Instance.Remove(carId);

        public void UpdateCar(Car car) => CarDBContext.Instance.Update(car);
    }
}
```

Activity 03: Design UI and write codes for WinForms project

Step 01. Right-click on the **AutomobileWinApp** project and add a new form named **frmCarDetails.cs** with UI as follows:



No.	Object Type	Object name	Properties / Events
1	Label	lbCarID	Text: Car ID
2	Label	lbCarName	Text: Car Name
3	Label	lbManufacturer	Text: Manufacturer
4	Label	lbPrice	Text: Price
5	Label	lbReleaseYear	Text: ReleaseYear
6	TextBox	txtCarID	
7	TextBox	txtCarName	
8	MaskedTextBox	txtPrice	Mask: 000000000 Text: 0
9	MaskedTextBox	txtReleaseYear	Mask: 0000 Text: 0
10	ComboBox	cboManufacturer	Items: Audi BMW Ford Honda Hyundai Kia Suzuki Toyota
11	Button	btnSave	Text: Save

			DialogResult: OK Event Handler: Click
12	Button	btnCancel	Text: Cancel DialogResult: Cancel Event Handler: Click
13	Form	frmCarDetails	StartPosition: CenterScreen Text: frmCarDetails Event Handler: Load

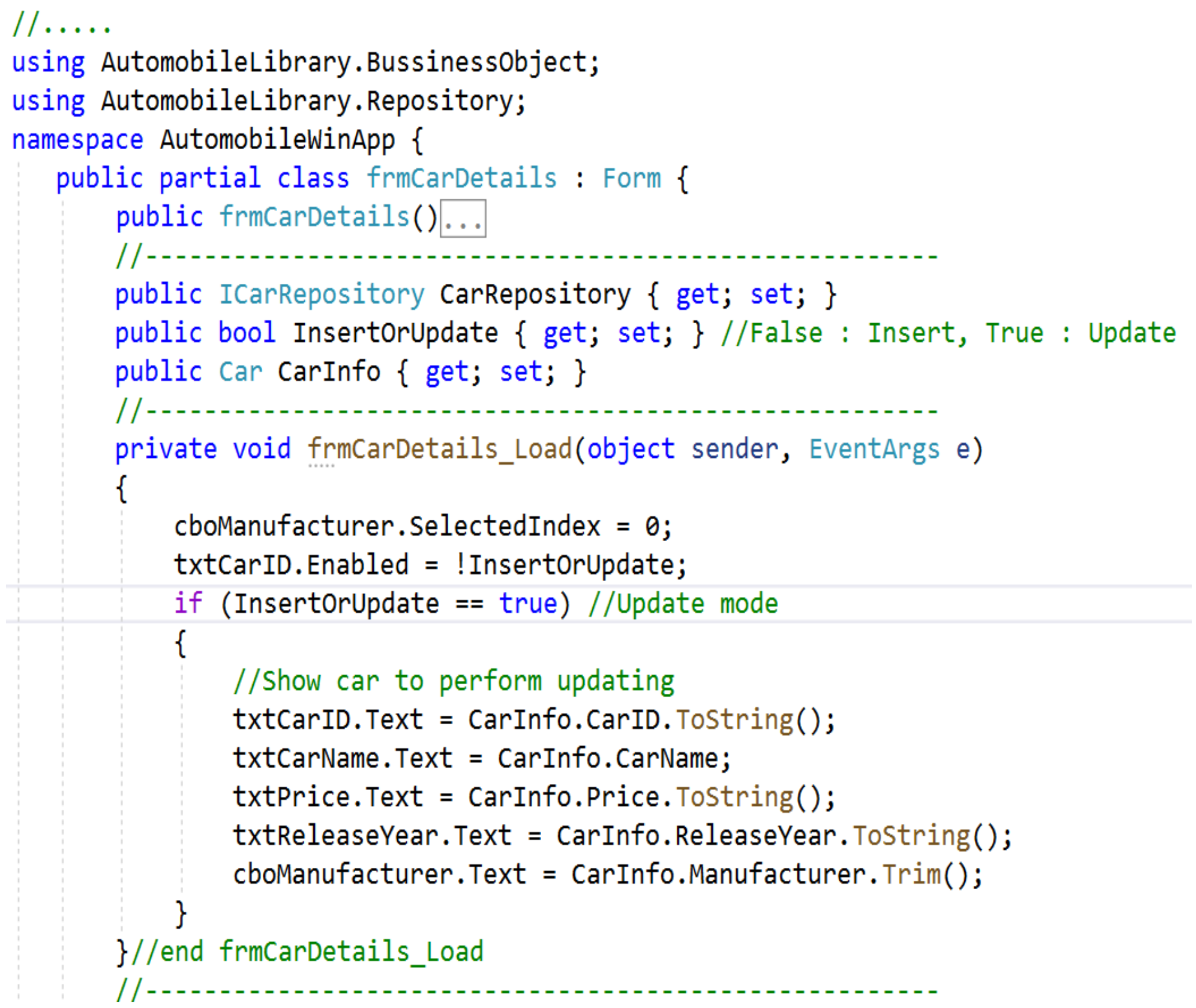
Step 02. Right-click on the project | Add | New Item, select **JavaScript JSON Configuration File** then rename to **appsettings.json** , click Add and write contents as follows:

```
{
  "ConnectionString": {
    "MyStockDB": "Server=(local);uid=sa;pwd=123;database=MyStock"
  }
}
```

Step 03. Right-click on the project, select **Edit Project File,** and write config information as follows then press Crtl+S to save:

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>WinExe</OutputType>
    <TargetFramework>net5.0-windows</TargetFramework>
    <UseWindowsForms>true</UseWindowsForms>
  </PropertyGroup>

  <ItemGroup>
  <ProjectReference Include="..\AutomobileLibrary\AutomobileLibrary.csproj" />
  </ItemGroup>

  <ItemGroup>
    <None Update="appsettings.json">
      <CopyToOutputDirectory>Always</CopyToOutputDirectory>
    </None>
  </ItemGroup>

</Project>
```

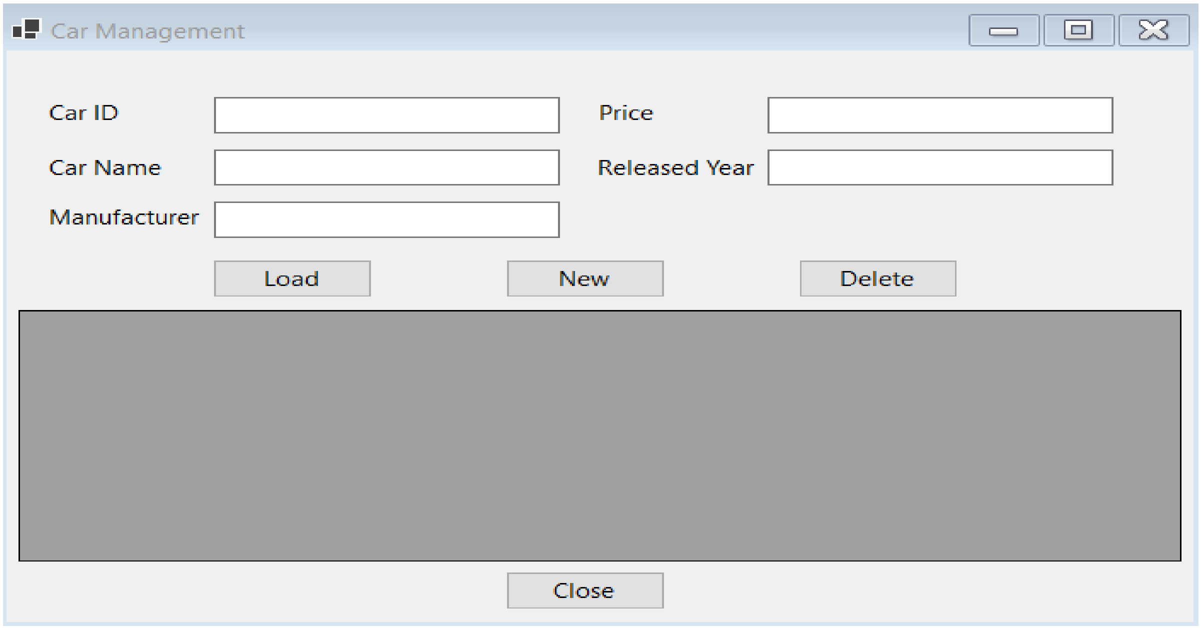
<u>Step 04</u>. Write codes for **frmCarDetails.cs**:



```
//.....
using AutomobileLibrary.BussinessObject;
using AutomobileLibrary.Repository;
namespace AutomobileWinApp {
   public partial class frmCarDetails : Form {
       public frmCarDetails()...
       //-----------------------------------------------------
       public ICarRepository CarRepository { get; set; }
       public bool InsertOrUpdate { get; set; } //False : Insert, True : Update
       public Car CarInfo { get; set; }
       //-----------------------------------------------------
       private void frmCarDetails_Load(object sender, EventArgs e)
       {
           cboManufacturer.SelectedIndex = 0;
           txtCarID.Enabled = !InsertOrUpdate;
           if (InsertOrUpdate == true) //Update mode
           {
               //Show car to perform updating
               txtCarID.Text = CarInfo.CarID.ToString();
               txtCarName.Text = CarInfo.CarName;
               txtPrice.Text = CarInfo.Price.ToString();
               txtReleaseYear.Text = CarInfo.ReleaseYear.ToString();
               cboManufacturer.Text = CarInfo.Manufacturer.Trim();
           }
       }//end frmCarDetails_Load
       //-----------------------------------------------------
```

```
        private void btnSave_Click(object sender, EventArgs e){
            try
            {
                var car = new Car {
                    CarID = int.Parse(txtCarID.Text),
                    CarName = txtCarName.Text,
                    Manufacturer = cboManufacturer.Text,
                    Price = decimal.Parse(txtPrice.Text),
                    ReleaseYear = int.Parse(txtReleaseYear.Text)
                };
                if(InsertOrUpdate == false){
                    CarRepository.InsertCar(car);
                }
                else{
                    CarRepository.UpdateCar(car);
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message,InsertOrUpdate==false?"Add a new car": "Update a car");
            }
        }//end btnSave_Click
         //-------------------------------------------------------
        private void btnCancel_Click(object sender, EventArgs e) => Close();
    }// end Form
}
```

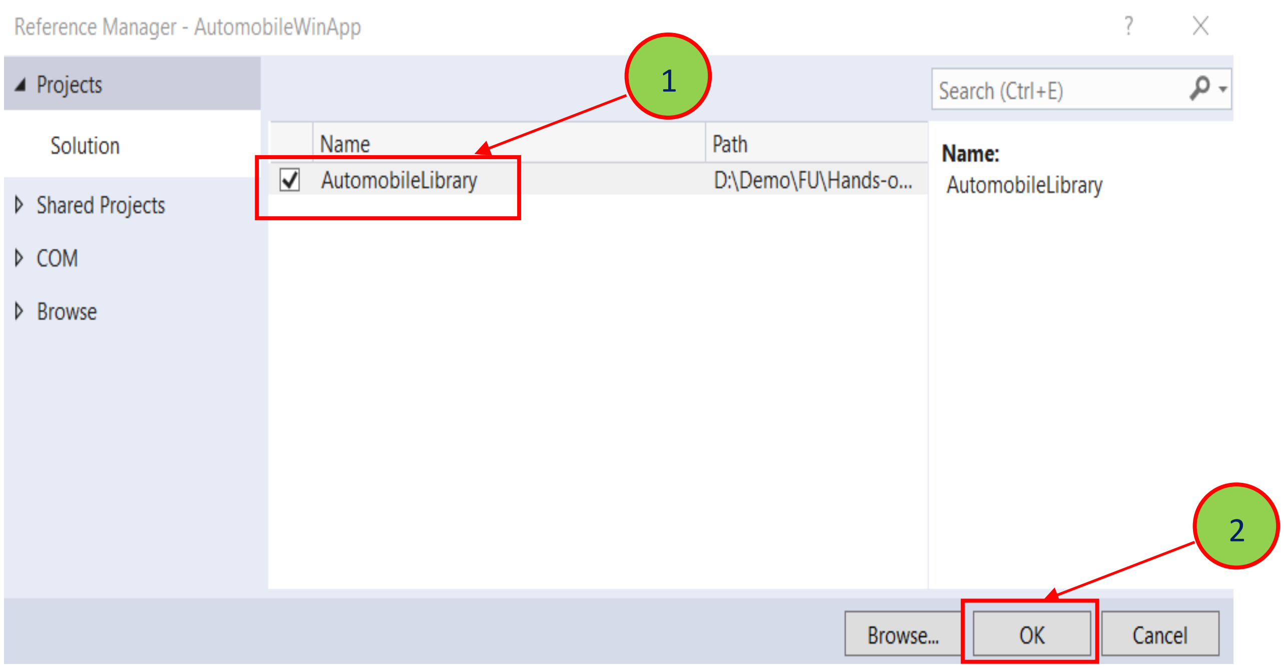
Step 05. Design UI for **frmCarManagement.cs** form:



No.	Object Type	Object name	Properties / Events
1	Label	lbCarID	Text: Car ID
2	Label	lbCarName	Text: Car Name
3	Label	lbManufacturer	Text: Manufacturer
4	Label	lbPrice	Text: Price
5	Label	lbReleaseYear	Text: ReleaseYear
6	TextBox	txtCarID	
7	TextBox	txtCarName	
8	TextBox	txtPrice	
9	TextBox	txtReleaseYear	
10	TextBox	txtManufacturer	
11	Button	btnLoad	Text: Load Event Handler: Click
12	Button	btnNew	Text: New Event Handler: Click
13	Button	btnDelete	Text: Delete Event Handler: Click
14	DataGridView	dgvCarList	ReadOnly: True SelectionMode:FullRowSelect
15	Form	frmCarManagement	StartPosition: CenterScreen Text: Car Management Event Handler: Load

Activity 04: Reference to AutomobileLibrary project and write code for WinForms project

Step 01. Right-click on **AutomobileWinApp** project, select Add | Project Reference, and perform as the below figure:



<u>Step 02</u>. Write codes for **frmCarManagement.cs**

```
using AutomobileLibrary.Repository;
using AutomobileLibrary.BussinessObject;
namespace AutomobileWinApp{
    public partial class frmCarManagement : Form{
        ICarRepository carRepository = new CarRepository();
        //Create a data source
        BindingSource source;
        //--------------------------------------------------------
        public frmCarManagement()...
        //--------------------------------------------------------
        private void frmCarManagement_Load(object sender, EventArgs e){
            btnDelete.Enabled = false;
            //Register this event to open the frmCarDetails form that performs updating
            dgvCarList.CellDoubleClick += DgvCarList_CellDoubleClick;
        }
        //--------------------------------------------------------
        private void DgvCarList_CellDoubleClick(object sender, DataGridViewCellEventArgs e){
            frmCarDetails frmCarDetails = new frmCarDetails{
                Text = "Update car",
                InsertOrUpdate = true,
                CarInfo = GetCarObject(),
                CarRepository = carRepository
            };
            if (frmCarDetails.ShowDialog() == DialogResult.OK){
                LoadCarList();
                //Set focus car updated
                source.Position = source.Count - 1;
            }
        }
```

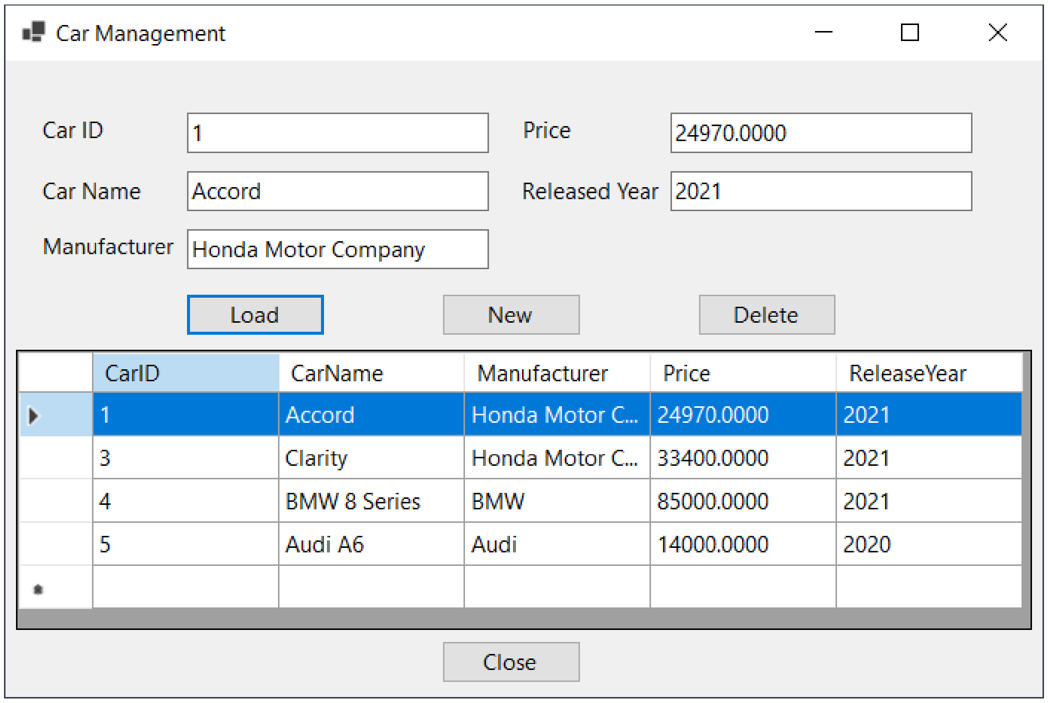
```
//Clear text on TextBoxes
private void ClearText(){
    txtCarID.Text = string.Empty;
    txtCarName.Text = string.Empty;
    txtManufacturer.Text = string.Empty;
    txtPrice.Text = string.Empty;
    txtReleaseYear.Text = string.Empty;
}
//--------------------------------------------------------
private Car GetCarObject()
{
    Car car = null;
    try
    {
        car = new Car
        {
            CarID = int.Parse(txtCarID.Text),
            CarName = txtCarName.Text,
            Manufacturer = txtManufacturer.Text,
            Price = decimal.Parse(txtPrice.Text),
            ReleaseYear = int.Parse(txtReleaseYear.Text)
        };
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message,"Get car");
    }
    return car;
}//end GetCarObject
//--------------------------------------------------------
```

```
public void LoadCarList(){
    var cars = carRepository.GetCars();
    try {
        //The BindingSource component is designed to simplify
        //the process of binding controls to an underlying data source
        source = new BindingSource();
        source.DataSource = cars;

        txtCarID.DataBindings.Clear();
        txtCarName.DataBindings.Clear();
        txtManufacturer.DataBindings.Clear();
        txtPrice.DataBindings.Clear();
        txtReleaseYear.DataBindings.Clear();

        txtCarID.DataBindings.Add("Text", source, "CarID");
        txtCarName.DataBindings.Add("Text", source, "CarName");
        txtManufacturer.DataBindings.Add("Text", source, "Manufacturer");
        txtPrice.DataBindings.Add("Text", source, "Price");
        txtReleaseYear.DataBindings.Add("Text", source, "ReleaseYear");

        dgvCarList.DataSource = null;
        dgvCarList.DataSource = source;
        if (cars.Count() == 0){
            ClearText();
            btnDelete.Enabled = false;
        }
        else{
            btnDelete.Enabled = true;
        }
    }
    catch (Exception ex) {
        MessageBox.Show(ex.Message,"Load car list");
    }
}//end LoadCarList
```

```
        //-----------------------------------------------------------
        private void btnLoad_Click(object sender, EventArgs e)
        {
            LoadCarList();
        }//end btnLoad_Click
         //-----------------------------------------------------------
        private void btnClose_Click(object sender, EventArgs e) => Close();
        //-----------------------------------------------------------
        private void btnNew_Click(object sender, EventArgs e) {
            frmCarDetails frmCarDetails = new frmCarDetails {
                Text = "Add car",
                InsertOrUpdate = false,
                CarRepository = carRepository
            };
            if(frmCarDetails.ShowDialog() == DialogResult.OK) {
                LoadCarList();
                //Set focus car inserted
                source.Position = source.Count - 1;
            }
        }
        //-----------------------------------------------------------

        private void btnDelete_Click(object sender, EventArgs e)
        {
            try
            {
                var car = GetCarObject();
                carRepository.DeleteCar(car.CarID);
                LoadCarList();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Delete a car");
            }
        }//end btnDelete_Click
    }//end Form
}
```

Activity 06: Press Ctrl+F5 to run the WinForms project and test all actions

Step 01. Click **Load** button and display the result as the below figure.

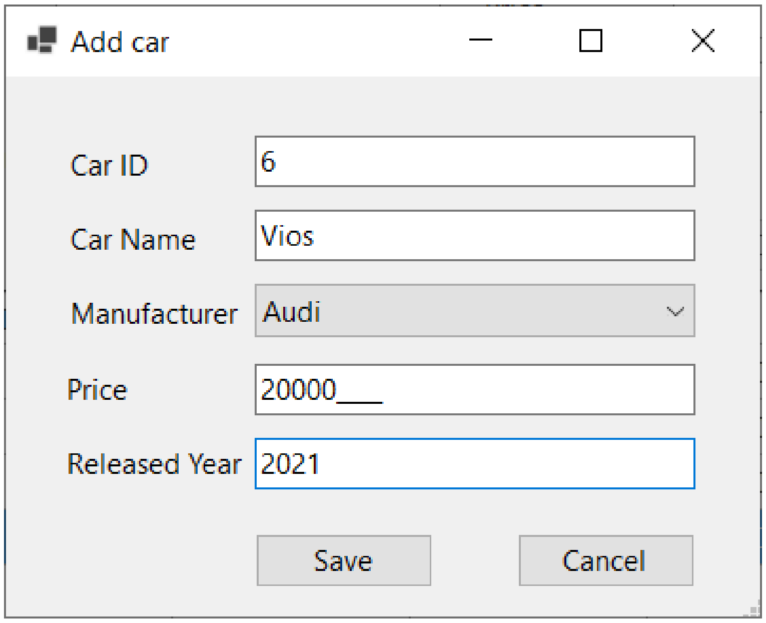


Car Management

Car ID: 1
Price: 24970.0000
Car Name: Accord
Released Year: 2021
Manufacturer: Honda Motor Company

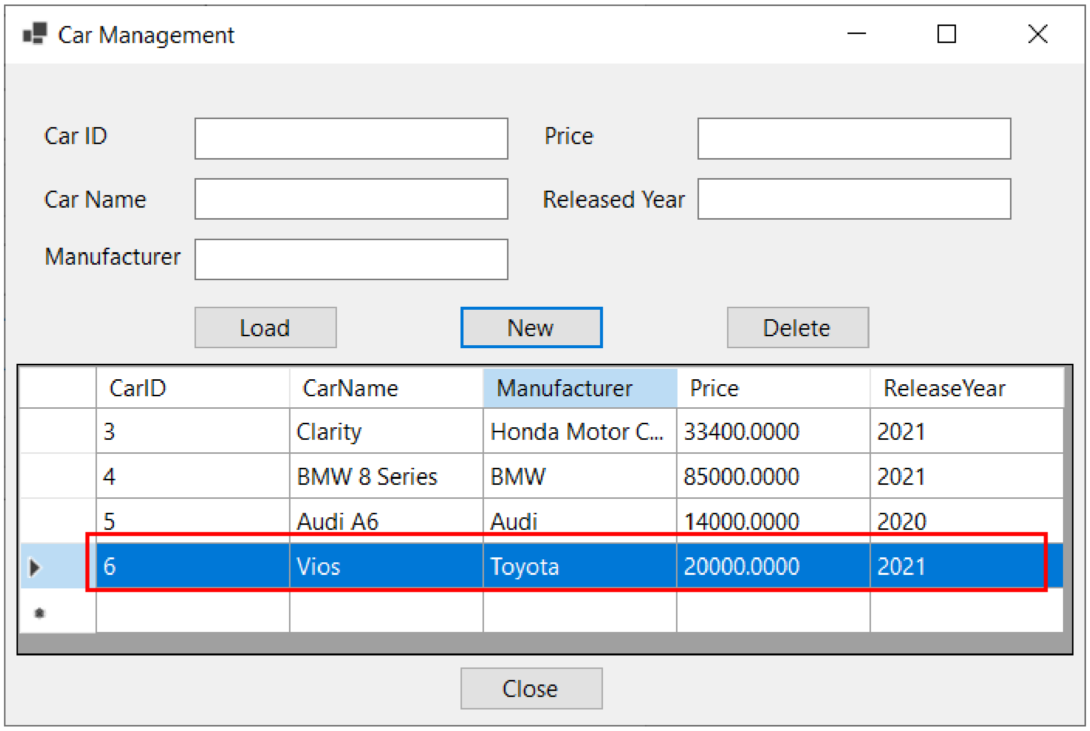
CarID	CarName	Manufacturer	Price	ReleaseYear
1	Accord	Honda Motor C...	24970.0000	2021
3	Clarity	Honda Motor C...	33400.0000	2021
4	BMW 8 Series	BMW	85000.0000	2021
5	Audi A6	Audi	14000.0000	2020

<u>Step 02</u>. Click **New** button and display the result as the below figure, enter the values on TextBoxes then click **Save** to add a new car.



Add car

Car ID: 6
Car Name: Vios
Manufacturer: Audi
Price: 20000___
Released Year: 2021

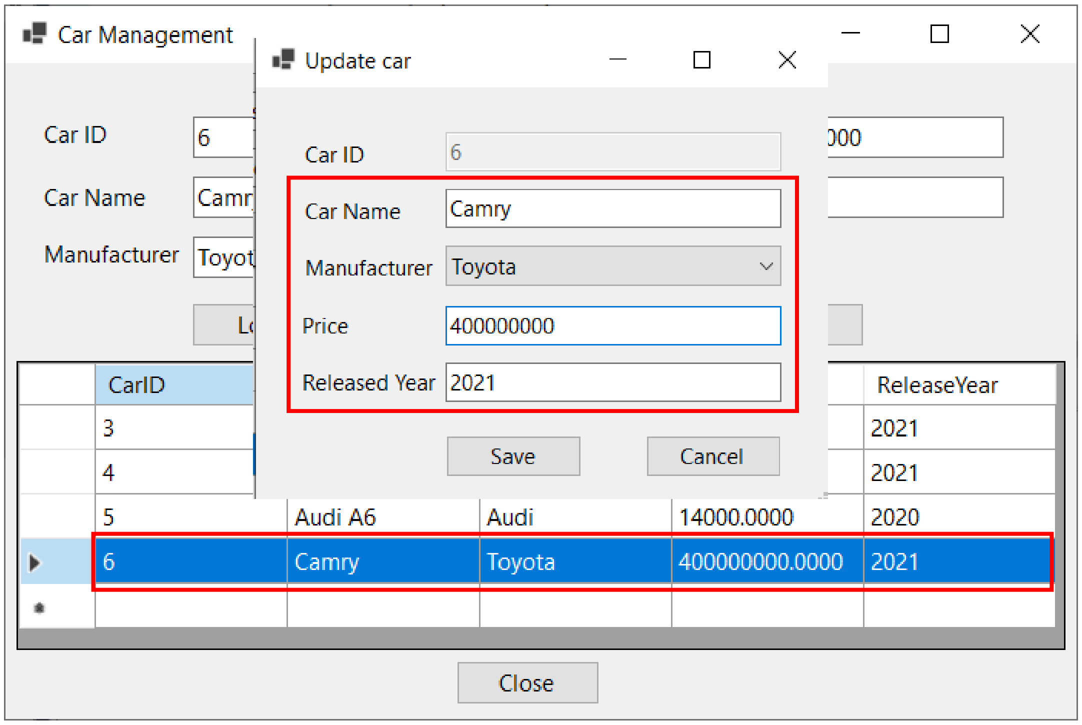


Car ID		Price	
Car Name		Released Year	
Manufacturer			

CarID	CarName	Manufacturer	Price	ReleaseYear
3	Clarity	Honda Motor C...	33400.0000	2021
4	BMW 8 Series	BMW	85000.0000	2021
5	Audi A6	Audi	14000.0000	2020
6	Vios	Toyota	20000.0000	2021

Step 03. Select a row on the DataGridView then click **Delete** to remove a Car.

Step 04. Double-click a row on the DataGridView to update a Car on the popup form, edit values then click **Save** to update.



Car ID	6
Car Name	Camry
Manufacturer	Toyota
Price	400000000
Released Year	2021

CarID	CarName	Manufacturer	Price	ReleaseYear
3				2021
4				2021
5	Audi A6	Audi	14000.0000	2020
6	Camry	Toyota	400000000.0000	2021